Kenneth G. Paterson (Ed.)

LNCS 6632

Advances in Cryptology – EUROCRYPT 2011

30th Annual International Conference
on the Theory and Applications of Cryptographic Techniques
Tallinn, Estonia, May 2011, Proceedings

Springer

Lecture Notes in Computer Science 6632

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

Kenneth G. Paterson (Ed.)

Advances in Cryptology – EUROCRYPT 2011

30th Annual International Conference
on the Theory and Applications of Cryptographic Techniques
Tallinn, Estonia, May 15-19, 2011
Proceedings

Springer

Volume Editor

Kenneth G. Paterson
Information Security Group (ISG)
Royal Holloway
University of London
Egham, Surrey TW20 0EX, UK
E-mail: kenny.paterson@rhul.ac.uk

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-20464-7 e-ISBN 978-3-642-20465-4
DOI 10.1007/978-3-642-20465-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011924899

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

These are the proceedings of Eurocrypt 2011, the 30th in the series of European Conferences on the Theory and Applications of Cryptographic Techniques. The conference was organized under the auspices of the International Association for Cryptologic Research and was held in Tallinn, Estonia, during May 15–19, 2011.

The aim of this series of conferences is to bring together leading researchers and practitioners from academia and industry in the field of cryptography. The conference program is intended to reflect the best of cryptographic research, in its widest sense. This year, a deliberate attempt was made to broaden the technical scope of the conference without making any compromise to its quality. The main mechanism for achieving this was to select Program Committee members from as broad a range of sub-areas of the field as possible, with the intention of sending a clear signal to potential authors from the field as a whole. I trust that readers of this volume find plenty to interest them here, and agree that the quality of the papers is as high as ever.

The program consisted of 2 invited talks and 31 contributed papers. The invited speakers were Ronald Cramer (CWI, Amsterdam and Mathematical Institute, Leiden) and Phong Nguyen (INRIA and ENS). I would like to thank them for accepting my invitation, for supplying informative abstracts for these proceedings, and for delivering excellent talks. It was a privilege to have such luminaries of our field as invited speakers.

The contributed papers were selected from 167 submissions. Each paper was reviewed by at least three people, with the submissions involving Program Committee members being subjected to at least five reviews each. There was significant online discussion about many of the papers, and a full-day Program Committee meeting was held at Royal Holloway on January 12, 2011 to finalize the program. The Program Committee decided to make a best paper award this year, and the award went to Eike Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain and Daniele Venturi for their paper "Efficient Authentication from Hard Learning Problems".

I would like to thank all the people who helped with the conference program and organization, particularly the General Chair, Helger Lipmaa. My heartfelt thanks go to the Program Committee and their sub-reviewers, as listed on the following pages, for their thoroughness during the review process. We had a tough assignment with many submissions and tight deadlines, and the committee members acted with utmost professionalism and attention to detail throughout. My particular thanks are due to Henri Gilbert, the previous Program Chair, who shared many insights with me, and to David Pointcheval, the next Program Chair, who kindly agreed to join the committee at short notice and who acted as a very effective "sweeper."

The submission and review process was greatly simplified by the ichair software developed by Thomas Baignères and Matthieu Finiasz. My thanks to them for producing this software and helping me with some technical queries during the review process. I will be sending them some Estonian delicacies by way of thanks; I highly recommend their software to all future Program Chairs. Thanks are also due to Tristan Findley and Jon Hart at Royal Holloway for maintaining the submission server and for their IT support during the Program Committee meeting.

I am grateful to the authors of all submitted papers for supporting the conference. The authors of accepted papers are thanked again for revising their papers according to the suggestions of the reviewers and for returning latex source files in good time. The revised versions were not checked by the Program Committee so authors bear full responsibility for their contents. I thank the staff at Springer for their help with producing the proceedings.

EuroCrypt 2011 was supported by the European Regional Development Fund (ERDF) through the Estonian Centre of Excellence in Computer Science, EXCS. I would also like to thank Guardtime, Qualcomm and Swedbank, the other sponsors of EuroCrypt 2011, for their generous support.

Finally, I would like to thank my partner Liz and my daughter Cara for their forbearance during a particularly hectic period.

February 2011 Kenny Paterson

Organization

Program Chair

Kenny Paterson	Royal Holloway, University of London, UK

General Chairs

Helger Lipmaa	Cybernetica AS, Estonia and Tallinn University, Estonia

Sponsors

The European Regional Development Fund (ERDF)
Guardtime
Qualcomm
Swedbank

Program Committee

Michel Abdalla	ENS Paris, France
Dan Bernstein	University of Chicago, USA
Xavier Boyen	University of Liege, Belgium
Christian Cachin	IBM Research, Switzerland
Orr Dunkelman	Weizmann Institute of Science, Israel
Pierrick Gaudry	LORIA, France
Henri Gilbert (2010 PC chair)	ANSSI, France
Jonathan Katz	University of Maryland, USA
Yoshi Kohno	University of Washington, USA
Benoit Libert	UCL, Belgium
Anna Lysyanskaya	Brown University, USA
Vadim Lyubashevsky	ENS Paris, France
Alex May	Ruhr University Bochum, Germany
Alfred Menezes	University of Waterloo, Canada
Tatsuaki Okamoto	NTT Labs, Japan
Rafail Ostrovsky	UCLA, USA
Elisabeth Oswald	University of Bristol, UK
Christof Paar	Ruhr University Bochum, Germany
Kenny Paterson (PC chair)	Royal Holloway, University of London, UK
Chris Peikert	Georgia Tech, USA
David Pointcheval (2012 PC chair)	ENS Paris, France

Renato Renner	ETH Zürich, Switzerland
Vincent Rijmen	K.U. Leuven, Belgium and TU Graz, Austria
Berry Schoenmakers	TU Eindhoven, The Netherlands
Mike Scott	DCU, Ireland
Hovav Shacham	UCSD, USA
Thomas Shrimpton	Portland State University, USA
Martijn Stam	EPFL, Switzerland
Doug Stinson	University of Waterloo, Canada
Frederik Vercauteren	K.U. Leuven, Belgium

Sub-reviewers

Johan Aaberg
Masayuki Abe
Divesh Aggarwal
Carlos Aguilar Melchor
Elena Andreeva
Benny Applebaum
Abhishek Banerjee
Aurelie Bauer
Georg Becker
Gaetan Bisson
Andrey Bogdanov
Joppe Bos
Zvika Brakerski
Christina Brzuska
Jan Camenisch
David Cash
Dario Catalano
Nishanth Chandran
Melissa Chase
Sanjit Chatterjee
Céline Chevalier
Sherman Chow
Jeremy Clark
Baudoin Collard
Daniel Dadush
Jean Paul Degabriele
Alex Dent
Claus Diem
Marten van Dijk
Dejan Dukaric
Frederic Dupuis
Thomas Eisenbarth
Nicolas Estibals
Junfeng Fan
Pooya Farshim
Sebastian Faust
Serge Fehr
Matthieu Finiasz
Dario Fiore
Marc Fischlin
Thomas Fuhr
Philippe Gaborit
Steven Galbraith
Sanjam Garg
Praveen Gauravaram
Ran Gelles
Clint Givens
Dov Gordon
Robert Granger
Jens Groth
Esther Haenggi
Brett Hemenway
Jens Hermans
Mathias Herrmann
Florian Hess
Stefan Heyse
Dennis Hofheinz
Susan Hohenberger
S.J.A. de Hoogh
Sebastiaan Indesteege
Abhishek Jain
Antoine Joux
Yael Tauman Kalai
Koray Karabina
Timo Kasper
Aniket Kate
Eike Kiltz
Mehmet S. Kiraz
Mikkel Krigrd
A. Kumarasubramanian
Mario Lamberger
Tanja Lange
Gregor Leander
Anja Lehmann
Arjen K. Lenstra
Peter van Liesdonk
Richard Lindner
Mark Manulis
Bart Mennink
Alexander Meurer
Petros Mol
Amir Moradi
Sean Murphy
Toru Nakanishi
Gregory Neven
Phong Nguyen
Jesper Buus Nielsen
Svetla Nikova
Ryo Nishimaki
Mehrdad Nojoumian
Femi Olumofin
Adam O'Neill
Onur Özen
Carles Padro
Rafael Pass
Ludovic Perret
Thomas Peters
Duong Hieu Phan
Krzysztof Pietrzak

Pandu Rangan
Oded Regev
Leo Reyzin
Alfredo Rial
Thomas Ristenpart
Matthieu Rivain
Louis Salvail
Rüdiger Schack
Christian Schaffner
Martin Schläffer
Aaron Segal
Yannick Seurin
Hakan Seyaliog
Aydin Sezgin
abhi shelat
Francesco Sica
Joe Silverman
Nigel Smart
F.-X. Standaert
Damien Stehlé
Anton Stolbunov
Björn Tackmann
Katsuyuki Takashima
Stefano Tessaro
Enrico Thomae
Deniz Toz
Joana Treger
Dominique Unruh
Vinod Vaikuntanathan
Kerem Varici
Damien Vergnaud
Marion Videau
Jorge L. Villar
Ivan Visconti
Akshay Wadia
Bogdan Warinschi
Brent Waters
Gaven Watson
Severin Winkler
Christopher Wolf
Stefan Wolf
Jiang Wu
Jürg Wullschleger
Keita Xagawa
Go Yamamoto
Kan Yasuda
Ralf Zimmermann

Table of Contents

Invited Talks

Lattice-Based Cryptography

Implementation and Side Channels

Homomorphic Cryptography

Signature Schemes

Information-Theoretic Cryptography

Symmetric Key Cryptography

Attacks and Algorithms

Secure Computation

Composability

Key Dependent Message Security

Public Key Encryption

The Arithmetic Codex: Theory and Applications

Ronald Cramer

CWI, Amsterdam & Mathematical Institute, Leiden University, The Netherlands
http://www.cwi.nl/~cramer

Abstract. We define the notion of an *arithmetic codex* (or *codex*, for short), and as a special case, *arithmetic secret sharing*. This notion encompasses as well as generalizes, in a single mathematical framework, all known types of specialized secret sharing schemes from the area of secure multi-party computation, i.e., the so-called *(strongly) multiplicative linear secret sharing schemes*.

These schemes were first studied as an abstract primitive by Cramer, Damgård, and Maurer in the late 1990s. They showed that the "*Fundamental Theorem of Information-Theoretically Secure Multi-Party Computation*," the landmark 1988 result by Ben-Or, Goldwasser, and Wigderson and, independently at the same time by Chaum, Crépeau, Damgård, admits a proof that uses this primitive as a blackbox: it is possible to bootstrap, in a blackbox fashion, from this primitive a set of atomic sub-protocols upon which general secure computation can be based. They also showed when and how multiplicative schemes (but not strongly multiplicative ones) reduce to ordinary ones and gave applications to security against non-threshold adversaries.

In 2006, Chen and Cramer showed an "asymptotically good" version of the Fundamental Theorem, where the size of the network is unbounded and where an adversary corrupts a constant fraction of the network, yet the information rate of the secret sharing primitive is constant. Their result relies on a careful choice of algebraic geometric codes, in combination with the earlier work of Cramer, Damgård, and Maurer.

In 2007 this asymptotic result turned out to have a surprising application in *two-party cryptography*, through the work of Ishai, Kushilevitz, Ostrovsky and Sahai ("*Multi-Party Computation in the Head*"). This first application was to zero knowledge for circuit satisfiability, but soon after other applications to secure two-party computation and information theory (correlation extractors) followed.

Our notion of arithmetic secret sharing is not merely a unification for its own sake. First, it casts these schemes in terms of a dedicated "representation" of K-algebras, thereby bringing the relevant mathematical structure to the surface. Second, it identifies novel types of special secret sharing schemes. And, third, there are novel cryptographic applications.

Besides presenting some elementary examples and giving an overview of the basic theory and the main applications, we discuss a construction of arithmetic secret sharing schemes based on a novel algebraic-geometric paradigm that we also introduce. This talk is mainly based on several recent joint works with Nacho Cascudo (CWI) and Chaoping Xing (NTU). But in part it is also based on recent joint work with Ivan Damgård (Aarhus University) and Valerio Pastro (Aarhus University).

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, p. 1, 2011.

Lattice Reduction Algorithms: Theory and Practice

Phong Q. Nguyen

INRIA and ENS, Département d'informatique, 45 rue d'Ulm, 75005 Paris, France
http://www.di.ens.fr/~pnguyen/

Abstract. Lattice reduction algorithms have surprisingly many applications in mathematics and computer science, notably in cryptology. On the one hand, lattice reduction algorithms are widely used in public-key cryptanalysis, for instance to attack special settings of RSA and DSA/ECDSA. On the other hand, there are more and more cryptographic schemes whose security require that certain lattice problems are hard. In this talk, we survey lattice reduction algorithms, present their performances, and discuss the differences between theory and practice.

Intuitively, a *lattice* is an infinite arrangement of points in $\mathbb{R}^m$ spaced with sufficient regularity that one can shift any point onto any other point by some symmetry of the arrangement. The simplest non-trivial lattice is the hypercubic lattice $\mathbb{Z}^n$ formed by all points with integral coordinates. The branch of number theory dealing with lattices (and especially their connection with convex sets) is known as *geometry of numbers* [24,41,12,5], and its origins go back to two historical problems: higher-dimensional generalizations of Euclid's gcd algorithm and sphere packings.

More formally, a lattice L is a discrete subgroup of $\mathbb{R}^m$, or equivalently, the set of all integer combinations of n linearly independent vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ in $\mathbb{R}^n$:

$$L = \{a_1\mathbf{b}_1 + \cdots + a_n\mathbf{b}_n, a_i \in \mathbb{Z}\}.$$

Such a set $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ is called a *basis* of the lattice. The goal of *lattice reduction* is to find reduced bases, that is bases consisting of reasonably short and nearly orthogonal vectors. This is related to the reduction theory of quadratic forms developed by Lagrange [19], Gauss [11] and Hermite [14]. Lattice reduction algorithms have proved invaluable in many fields of computer science and mathematics (see the book [30]), notably public-key cryptanalysis where they have been used to break knapsack cryptosystems [32] and special cases of RSA and DSA, among others (see [26,21] and references therein).

Reduced bases allow to solve the following important lattice problems, either exactly or approximately:

- The most basic computational problem involving lattices is the *shortest vector problem* (SVP), which asks to find a nonzero lattice vector of smallest norm, given a lattice basis as input. SVP can be viewed as a geometric generalization of gcd computations: Euclid's algorithm actually computes the

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 2–6, 2011.

smallest (in absolute value) non-zero linear combination of two integers, since $\gcd(a,b)\mathbb{Z} = a\mathbb{Z}+b\mathbb{Z}$, which means that we are replacing the integers a and b by an arbitrary number of vectors $\mathbf{b}_1, \ldots, \mathbf{b}_n$ with integer coordinates. Since SVP is NP-hard under randomized reductions [3] (see [17,34] for surveys on the hardness of lattice problems), one is also interested in approximating SVP, *i.e.* to output a nonzero lattice vector of norm not much larger than the smallest norm.

– The inhomogeneous version of SVP is called the *closest vector problem* (CVP); here we are given an arbitrary target vector in addition to the lattice basis and asked to find the lattice point closest to that vector. A popular particular case of CVP is *Bounded Distance Decoding* (BDD), where the target vector is known to be somewhat close to the lattice.

The first SVP algorithm was Lagrange's reduction algorithm [19], which solves SVP exactly in dimension two, in quadratic time. In arbitrary dimension, there are two types of SVP algorithms:

1. **Exact algorithms.** These algorithms provably find a shortest vector, but they are expensive, with a running time at least exponential in the dimension. Intuitively, these algorithms perform an exhaustive search of all extremely short lattice vectors, whose number is exponential in the dimension (in the worst case): in fact, there are lattices for which the number of shortest lattice vectors is already exponential. Exact algorithms can be split in two categories:
   (a) Polynomial-space exact algorithms. They are based on *enumeration* which dates back to the early 1980s with work by Pohst [33], Kannan [16], and Fincke-Pohst [6]. In its simplest form, enumeration is simply an exhaustive search for the best integer combination of the basis vectors. The best deterministic enumeration algorithm is Kannan's algorithm [16], with super-exponential worst-case complexity, namely $n^{n/(2e)+o(n)}$ polynomial-time operations (see [13]), where n denotes the lattice dimension. The enumeration algorithms used in practice (such as that of Schnorr-Euchner [37]) have a weaker preprocessing than Kannan's algorithm [16], and their worst-case complexity is $2^{O(n^2)}$ polynomial-time operations. But it is possible to obtain substantial speedups using *pruning* techniques: pruning was introduced by Schnorr-Euchner [37] and Schnorr-Hörner [38] in the 90s, and recently revisited by Gama, Nguyen and Regev [10], where it was shown that one can reach a $2^{n/2}$ heuristic speedup over basic enumeration.
   (b) Exponential-space exact algorithms. These algorithms have a better asymptotic running time, but they all require exponential space $2^{\Theta(n)}$. The first algorithm of this kind is the randomized sieve algorithm of Ajtai, Kumar and Sivakumar (AKS) [4], with exponential worst-case complexity of $2^{O(n)}$ polynomial-time operations. Micciancio and Voulgaris [22] recently presented an alternative deterministic algorithm, which solves both CVP and SVP within $2^{2n+o(n)}$ polynomial-time operations. Interestingly, there are several heuristic variants [31,23,43] of AKS with running time $2^{O(n)}$, where the $O()$ constant is much less than that of the best provable

algorithms known. For instance, the recent algorithm of Wang *et al.* [43] has time complexity $2^{0.3836n}$ polynomial-time operations.

2. **Approximation algorithms.** These algorithms are much faster than exact algorithms, but they only output short lattice vectors, not necessarily the shortest one: they typically output a whole reduced basis, and are therefore lattice reduction algorithms. The first algorithm of this kind is the celebrated algorithm of Lenstra, Lenstra and Lovász (LLL) [20,30], which can approximate SVP to within a factor $O((2/\sqrt{3})^n)$ in polynomial time: it can be viewed as an algorithmic version of Hermite's inequality. Since the appearance of LLL, research in this area has focused on two topics:
   (a) **Faster LLL.** Here, one is interested in obtaining reduced bases of similar quality than LLL, possibly slightly worse, but with a smaller running time. This is achieved by a divide-and-conquer strategy (such as in [39,18]) or by using floating-point arithmetic (such as in [36,29,25]). The most popular implementations of LLL are typically heuristic floating-point variants, such as that of Schnorr-Euchner [37]: see the survey [42] on floating-point LLL.
   (b) **Stronger LLL.** Here, one is interested in obtaining better approximation factors than LLL, at the expense of the running time. Intuitively, LLL repeatedly uses two-dimensional reduction to find short lattice vectors in dimension n. Blockwise reduction algorithms [35,7,8] obtain better approximation factors by replacing this two-dimensional reduction subroutine by a higher-dimensional one, using exact SVP algorithms in low dimension. The best polynomial-time blockwise algorithm known [8] achieves a subexponential approximation factor $2^{O((n \log\log n)/\log n)}$: it is an algorithmic version of Mordell's inequality. In practice, a popular choice is the BKZ algorithm of Schnorr-Euchner [37] implemented in the NTL library [40], which is a heuristic variant of Schnorr's blockwise algorithm [35]. The article [9] provides an experimental assessment of BKZ.

Both categories are in fact complementary: all exact algorithms known first apply an approximation algorithm (typically at least LLL) as a preprocessing, while all blockwise algorithms call many times an exact algorithm in low dimension as a subroutine. Most of the SVP algorithms we mentioned can be adapted to CVP (see for instance [1]). The provable SVP algorithms are surveyed in [27]. The heuristic algorithms which we mentioned are such that their running time may no longer be proved, and/or there may not be any guarantee on the output (should the algorithm ever terminate). Heuristic algorithms can typically outperform provable algorithms in practice, for reasons still not well understood.

Finally, it is folklore that lattice reduction algorithms behave better than their proved worst-case theoretical bounds. In the 80s, the early success of lattice reduction algorithms in cryptanalysis led to the belief that the strongest lattice reduction algorithms behaved as perfect oracles, at least in small dimension. But this belief showed its limits in the 90s with NP-hardness results and the development of lattice-based cryptography, following Ajtai's worst-case/average-case

reduction [2] and the NTRU cryptosystem [15]. The articles [28,9] clarify what can be expected in practice, based on experimental results. Such assessments are important to better understand the gap between theory and practice, but also to evaluate the concrete security of lattice-based cryptography.

References

1. Agrell, E., Eriksson, T., Vardy, A., Zeger, K.: Closest point search in lattices. IEEE Trans. on Info. Theory 48(8), 2201–2214 (2002)
2. Ajtai, M.: Generating hard instances of lattice problems. In: Proc. STOC 1996, pp. 99–108. ACM, New York (1996)
3. Ajtai, M.: The shortest vector problem in L_2 is NP-hard for randomized reductions. In: Proc. of 30th STOC. ACM, New York (1998)
4. Ajtai, M., Kumar, R., Sivakumar, D.: A sieve algorithm for the shortest lattice vector problem. In: Proc. 33rd STOC, pp. 601–610 (2001)
5. Cassels, J.: An Introduction to the Geometry of Numbers (1997)
6. Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Mathematics of Computation 44(170), 463–471 (1985)
7. Gama, N., Howgrave-Graham, N., Koy, H., Nguyên, P.Q.: Rankin's constant and blockwise lattice reduction. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 112–130. Springer, Heidelberg (2006)
8. Gama, N., Nguyen, P.Q.: Finding short lattice vectors within Mordell's inequality. In: Proc. 40th ACM Symp. on Theory of Computing (STOC) (2008)
9. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
10. Gama, N., Nguyen, P.Q., Regev, O.: Lattice enumeration using extreme pruning. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 257–278. Springer, Heidelberg (2010)
11. Gauss, C.: Disquisitiones Arithmeticæ, Leipzig (1801)
12. Gruber, M., Lekkerkerker, C.G.: Geometry of Numbers. North-Holland, Amsterdam (1987)
13. Hanrot, G., Stehlé, D.: Improved analysis of Kannan's shortest lattice vector algorithm (extended abstract). In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 170–186. Springer, Heidelberg (2007)
14. Hermite, C.: Extraits de lettres de M. Hermite à M. Jacobi sur différents objets de la théorie des nombres. J. Reine Angew. Math. 40, 261–315 (1850)
15. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS III 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
16. Kannan, R.: Improved algorithms for integer programming and related lattice problems. In: Proc. 15th ACM Symp. on Theory of Computing (STOC), pp. 193–206 (1983)
17. Khot, S.: Inapproximability results for computational problems on lattices. In: [30] (2010)
18. Koy, H., Schnorr, C.-P.: Segment LLL-reduction of lattice bases. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 67–80. Springer, Heidelberg (2001)
19. Lagrange, L.: Recherches d'arithmétique. Nouv. Mém. Acad. (1773)

20. Lenstra, A.K., Lenstra Jr., H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Ann. 261, 513–534 (1982)
21. May, A.: Using LLL-reduction for solving RSA and factorization problems: A survey. In: [30] (2010)
22. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In: Proc. STOC 2010, pp. 351–358. ACM, New York (2010)
23. Micciancio, D., Voulgaris, P.: Faster exponential time algorithms for the shortest vector problem. In: Proc. ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 1468–1480 (2010)
24. Minkowski, H.: Geometrie der Zahlen. Teubner-Verlag, Leipzig (1896)
25. Morel, I., Stehlé, D., Villard, G.: H-LLL: using householder inside LLL. In: Proc. ISSAC 2009, pp. 271–278. ACM, New York (2009)
26. Nguyen, P.Q.: Public-key cryptanalysis. In: Luengo, I. (ed.) Recent Trends in Cryptography. Contemporary Mathematics, vol. 477. AMS–RSME (2009)
27. Nguyen, P.Q.: Hermite's constant and lattice algorithms. In: [30] (2010)
28. Nguyên, P.Q., Stehlé, D.: LLL on the average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS VII 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)
29. Nguyen, P.Q., Stehlé, D.: An LLL algorithm with quadratic complexity. SIAM J. Comput. 39(3), 874–903 (2009)
30. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm: Survey and Applications. Information Security and Cryptography. Springer, Heidelberg (2010)
31. Nguyen, P.Q., Vidick, T.: Sieve algorithms for the shortest vector problem are practical. J. of Mathematical Cryptology 2(2), 181–207 (2008)
32. Odlyzko, A.M.: The rise and fall of knapsack cryptosystems. In: Cryptology and Computational Number Theory. Proc. of Symposia in Applied Mathematics, vol. 42, pp. 75–88. AMS, Providence (1990)
33. Pohst, M.: On the computation of lattice vectors of minimal length, successive minima and reduced bases with applications. SIGSAM Bull. 15(1), 37–44 (1981)
34. Regev, O.: On the Complexity of Lattice Problems with Polynomial Approximation Factors. In: [30] (2010)
35. Schnorr, C.-P.: A hierarchy of polynomial lattice basis reduction algorithms. Theoretical Computer Science 53(2-3), 201–224 (1987)
36. Schnorr, C.-P.: A more efficient algorithm for lattice basis reduction. J. Algorithms 9(1), 47–62 (1988)
37. Schnorr, C.-P., Euchner, M.: Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Programming 66, 181–199 (1994)
38. Schnorr, C.-P., Hörner, H.H.: Attacking the chor-rivest cryptosystem by improved lattice reduction. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 1–12. Springer, Heidelberg (1995)
39. Schönhage, A.: Factorization of univariate integer polynomials by diophantine aproximation and an improved basis reduction algorithm. In: Paredaens, J. (ed.) ICALP 1984. LNCS, vol. 172, pp. 436–447. Springer, Heidelberg (1984)
40. Shoup, V.: Number Theory C++ Library (NTL) version 5.4.1, http://www.shoup.net/ntl/
41. Siegel, C.L.: Lectures on the Geometry of Numbers. Springer, Heidelberg (1989)
42. Stehlé, D.: Floating-point LLL: theoretical and practical aspects. In: [30] (2010)
43. Wang, X., Liu, M., Tian, C., Bi, J.: Improved Nguyen-Vidick heuristic sieve algorithm for shortest vector problem. Cryptology ePrint Archive, Report 2010/647 (2010), http://eprint.iacr.org/

Efficient Authentication from Hard Learning Problems

Eike Kiltz[1,⋆], Krzysztof Pietrzak[2,⋆⋆], David Cash[3,⋆⋆⋆], Abhishek Jain[4,†], and Daniele Venturi[5,†]

[1] RU Bochum
[2] CWI Amsterdam
[3] UC San Diego
[4] UC Los Angeles
[5] Sapienza University of Rome

Abstract. We construct efficient authentication protocols and message-authentication codes (MACs) whose security can be reduced to the learning parity with noise (LPN) problem.

Despite a large body of work – starting with the HB protocol of Hopper and Blum in 2001 – until now it was not even known how to construct an efficient authentication protocol from LPN which is secure against man-in-the-middle (MIM) attacks. A MAC implies such a (two-round) protocol.

1 Introduction

Authentication is among the most basic and important cryptographic tasks. In the present paper we construct efficient (secret-key) authentication schemes from the *learning parity with noise* (LPN) problem. We construct the first efficient message authentication codes (MACs) from LPN, but also simpler and more efficient two-round authentication protocols that achieve a notion called active security. Prior to our work, the only known way to construct an LPN-based MAC was via a relatively inefficient generic transformation [17] (that works with any pseudorandom generator), and all interactive LPN-based protocols with security properties similar to our new protocol required at least three rounds and had a loose security reduction. Our constructions and techniques diverge significantly from prior work in the area and will hopefully be of independent interest.

The pursuit of LPN-based authentication is motivated by two disjoint concerns, one theoretical and one practical. On the theoretical side, the LPN problem provides an attractive basis for provable security [3, 4, 6, 22, 18, 27]. It is

⋆ Funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

⋆⋆ Supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Starting Grant (259668-PSPC).

⋆⋆⋆ Supported by NSF CCF-0915675. Research done while visiting CWI Amsterdam.

† Research done while visiting CWI Amsterdam.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 7–26, 2011.

closely related to the well-studied problem of decoding random linear codes, and unlike most number-theoretic problems used in cryptography, the LPN problem does not succumb to known quantum algorithms. On the practical side, LPN-based authentication schemes are strikingly efficient, requiring relatively few bit-level operations. Indeed, in their original proposal, Hopper and Blum [18] suggested that humans could perform the computation in their provably-secure scheme, even with realistic parameters. The efficiency of LPN-based schemes also makes them suitable for weak devices like RFID tags, where even evaluating a blockcipher may be prohibitive.

Each of our theoretical and practical motivations, on its own, would be sufficiently interesting for investigation, but together the combination is particularly compelling. LPN-based authentication is able to provide a theoretical improvement in terms of provable security *in addition to* providing better efficiency than approaches based on more classical symmetric techniques that are not related to hard problems. Usually we trade one benefit for the other, but here we hope to get the best of both worlds.

Before describing our contributions in more detail, we start by recalling authentication protocols, the LPN problem, and some of the prior work on which we build.

Authentication protocols. An authentication protocol is a (shared-key) protocol where a prover $\mathcal{P}$ authenticates itself to a verifier $\mathcal{V}$ (in the context of RFID implementations, we think of $\mathcal{P}$ as the "tag" and $\mathcal{V}$ as the "reader"). We recall some of the common definitions for security against impersonation attacks. A *passive attack* proceeds in two phases, where in the first phase the adversary eavesdrops on several interactions between $\mathcal{P}$ and $\mathcal{V}$, and then attempts to cause $\mathcal{V}$ to accept in the second phase (where $\mathcal{P}$ is no longer available). In an *active attack*, the adversary is additionally allowed to interact with $\mathcal{P}$ in the first phase. The strongest and most realistic attack model is a *man-in-the-middle attack* (MIM), where the adversary can arbitrarily interact with $\mathcal{P}$ and $\mathcal{V}$ (with polynomially many concurrent executions allowed) in the first phase.

The LPN problem. Briefly stated, the LPN problem is to distinguish from random several "noisy inner products" of random binary vectors with a random secret vector.

More formally, for $\tau < 1/2$ and a vector $\mathbf{x} \in \mathbb{Z}_2^\ell$, define the distribution $\Lambda_{\tau,\ell}(\mathbf{x})$ on $\mathbb{Z}_2^\ell \times \mathbb{Z}_2$ by $(\mathbf{r}, \mathbf{r}^\mathsf{T}\mathbf{x} \oplus e)$, where $\mathbf{r} \in \mathbb{Z}_2^\ell$ is uniformly random and $e \in \mathbb{Z}_2$ is selected according to Ber_τ, the Bernoulli distribution over $\mathbb{Z}_2$ with parameter τ (i.e. $\Pr[e = 1] = \tau$). The $\mathsf{LPN}_{\tau,\ell}$ problem is to distinguish an oracle returning samples from $\Lambda_{\tau,\ell}(\mathbf{x})$, where $\mathbf{x} \in \mathbb{Z}_2^\ell$ is random and fixed, from an oracle returning uniform samples. It was shown by Blum et al. [4] that this is equivalent to the search version of LPN, where one needs to compute $\mathbf{x}$ given oracle access to $\Lambda_{\tau,\ell}(\mathbf{x})$ (cf. [21, Thm.2] for precise bounds). We note that the search and decision variants are solvable with a linear in ℓ number of samples when there is no noise, i.e. when $\tau = 0$, and the best algorithms take time $2^{\ell/\log \ell}$ when $\tau > 0$ is treated as a constant [5, 6, 23].

Authentication protocols from LPN. Starting with the work of Hopper and Blum [18], several authentication protocols based on the LPN problem have been proposed. Their original elegant protocol is simple enough for us to recall right away. The shared secret key is a binary vector $\mathbf{s} \in \mathbb{Z}_2^\ell$. The interaction consists of two messages. First $\mathcal{V}$ sends a random challenge $\mathbf{r} \in \mathbb{Z}_2^\ell$, and then $\mathcal{P}$ answers with the bit $z = \mathbf{r}^\mathsf{T}\mathbf{s} \oplus e$, where $e \in \mathbb{Z}_2$ is sampled according to Ber_τ. Finally, the verifier accepts if $z = \mathbf{r}^\mathsf{T}\mathbf{s}$.

This basic protocol has a large completeness error τ (as $\mathcal{V}$ will reject if $e = 1$) and soundness error $1/2$ (as a random $\mathbf{r}$, z satisfies $\mathbf{r}^\mathsf{T} \cdot \mathbf{s} = z$ with probability $1/2$). This can be reduced via sequential or parallel composition. The parallel variant, denoted HB, is illustrated in Figure 1 (we represent several $\mathbf{r}$ with a matrix $\mathbf{R}$ and the noise bits are now arranged in a vector $\mathbf{e}$). The verifier accepts if at least a τ' fraction (where $\tau < \tau' < 1/2$) of the n basic authentication steps are correct.

The 2-round HB protocol is provably secure against passive attacks, but efficient active attacks are known against it. This is unsatisfying because in several scenarios, and especially in RFID applications, an adversary *will* be able to mount an active attack. Subsequently, Juels and Weis [19] proposed an efficient 3 round variant of HB, called HB^+, and proved it secure against active attacks. Again the error can be reduced by sequential repetition, and as shown by Katz, Shin and Smith via a non-trivial analysis, parallel repetition works as well [20, 21]. The protocol (in its parallel repetition variant) is illustrated in Figure 2.

Despite a large body of subsequent work[1] no improvements in terms of round complexity, security or tightness of the reduction over HB^+ were achieved: 3 round protocols achieving active security $\sqrt{\varepsilon}$ (assuming LPN is ε-hard) are the state of the art. In particular, Gilbert et al. [14] showed that HB^+ can be broken by a MIM attack. Several variants HB^{++} [9], HB^* [11], HB-MP [24] were proposed to prevent the particular attack from [14], but all of them were later shown to be insecure [15]. In [16], a variant $\mathsf{HB}^\#$ was presented which provably resists the particular attack from [14], but was shown susceptible to a more general MIM attack [25].

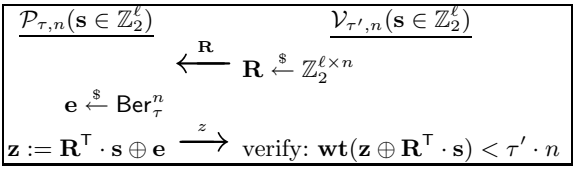


Fig. 1. The HB protocol, secure against passive attacks

[1] cf. http://www.ecrypt.eu.org/lightweight/index.php/HB for an incomplete list of relevant papers.

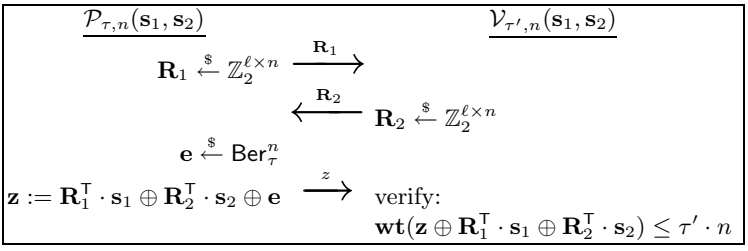


Fig. 2. The HB^+ protocol, secure against active attacks

1.1 Our Contribution

We provide new constructions of authentication protocols and even MACs from LPN. Our first contribution is a *two*-round authentication protocol secure against active adversaries (this is mentioned as an open problem in [19]) which moreover has a tight security reduction (an open problem mentioned in [21]). As a second contribution, we build two efficient MACs, and thus also get two-round authentication protocols secure against MIM attacks, from the LPN assumption. Unlike previous proposals, our constructions are not ad-hoc, and we give a reduction to the LPN problem. Our authentication protocol is roughly as efficient as the HB^+ protocol but has twice the key length. Our MACs perform roughly the same computation as the authentication protocol plus one evaluation of a pairwise independent permutation of an $\approx 2\ell$ bit domain, where ℓ is the length of an LPN secret.

2-Round Authentication with Active Security. Our first contribution is a two-round authentication protocol which we prove secure against *active* attacks assuming the hardness of the LPN problem. Our protocol diverges considerably from all previous HB-type protocols [18, 19, 21, 16], and runs counter to the intuition that the only way to efficiently embed the LPN problem into a two-round protocol is via an HB-type construction.

We now sketch our protocol. In HB and its two-round variants, the prover must compute LPN samples of the form $\mathbf{R}^\mathsf{T}\cdot\mathbf{s}\oplus\mathbf{e}$, where $\mathbf{R}$ is the challenge chosen by the verifier in the first message. We take a different approach. Instead of sending $\mathbf{R}$, we now let the verifier choose a random subset of the bits of $\mathbf{s}$ to act as the "session-key" for this interaction. It represents this subset by sending a binary vector $\mathbf{v}\in\mathbb{Z}_2^\ell$ that acts as a "bit selector" of the secret $\mathbf{s}$, and we write $\mathbf{s}_{\downarrow\mathbf{v}}$ for the sub-vector of $\mathbf{s}$ which is obtained by deleting all bits from $\mathbf{s}$ where $\mathbf{v}$ is 0. (E.g. if $\mathbf{s}=111000, \mathbf{v}=011100$ then $\mathbf{s}_{\downarrow\mathbf{v}}=110$). The prover then picks $\mathbf{R}$ by *itself* and computes noisy inner products of the form $\mathbf{R}^\mathsf{T}\cdot\mathbf{s}_{\downarrow\mathbf{v}}\oplus\mathbf{e}$. Curiously, allowing the verifier to choose which bits of $\mathbf{s}$ to use in each session is sufficient to prevent active attacks. We only need to add a few sanity-checks that no pathological $\mathbf{v}$ or $\mathbf{R}$ were sent by an active adversary.

Our proof relies on the recently introduced *subspace LPN problem* [26]. In contrast to the active-attack security proof of HB^+ [21], our proof does not use

any rewinding techniques. Avoiding rewinding has at least two advantages. First, the security reduction becomes tight. Second, the proofs also works in a quantum setting: our protocol is secure against quantum adversaries assuming LPN is secure against such adversaries. As first observed by van de Graaf [29], classical proofs using rewinding in general do not translate to the quantum setting (cf. [31] for a more recent discussion). Let us emphasise that this only means that there is no security proof for HB^+ in the quantum setting, but we do not know if a quantum attack actually exists.

MAC & Man-In-The-Middle Security. In Section 4, we give two constructions of message authentication codes (MACs) that are secure (formally, unforgeable under chosen message attacks) assuming that the LPN problem is hard. Note that a MAC implies a two-round MIM-secure authentication protocol: the verifier chooses a random message as challenge, and the prover returns the MAC on the message.

As a first attempt, let us try to view our authentication protocol as a MAC. That is, a MAC tag is of the form $\phi = (\mathbf{R}, \mathbf{z} = \mathbf{R}^\mathsf{T} \cdot f_\mathbf{s}(\mathbf{m}) \oplus \mathbf{e})$, where the secret key derivation function $f_\mathbf{s}(\mathbf{m}) \in \mathbb{Z}_2^\ell$ first uniquely encodes the message $\mathbf{m}$ into $\mathbf{v} \in \mathbb{Z}_2^{2\ell}$ of weight ℓ and then returns $\mathbf{s}_{\downarrow\mathbf{v}}$ by selecting ℓ bits from secret $\mathbf{s}$, according to $\mathbf{v}$. However, this MAC is not secure: given a MAC tag $\phi = (\mathbf{R}, \mathbf{z})$ an adversary can ask verification queries where it sets individual rows of $\mathbf{R}$ to zero until verification fails: if the last row set to zero was the ith, then the ith bit of $f_\mathbf{s}(\mathbf{m})$ must be 1. (In fact, the main technical difficulty to build a secure MAC from LPN is to make sure the secret $\mathbf{s}$ does not leak from verification queries). Our solution is to randomize the mapping f, i.e. use $f_\mathbf{s}(\mathbf{m}, \mathbf{b})$ for some randomness $\mathbf{b}$ and compute the tag as $\phi = \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot f_\mathbf{s}(\mathbf{m}, \mathbf{b}) \oplus \mathbf{e}, \mathbf{b})$, where π is a pairwise independent permutation (contained in the secret key). We can prove that if LPN is hard then this construction yields a secure MAC. (The key argument is that, with high probability, all non-trivial verification queries are inconsistent and hence lead to reject). However, the security reduction to the LPN problem is quite loose since it has to guess the value $\mathbf{v}$ from the adversary's forgery. (In the context of identity-based encryption (IBE) a similar idea has been used to go from selective-ID to full security using "complexity leveraging" [7]). In our case, however, this still leads to a polynomial security reduction when one commits to the hardness of the LPN problem at the time of the construction. (See the first paragraph of §4 for a discussion).

To get a strictly polynomial security reduction (without having to commit to the hardness of the LPN problem), in our second construction we adapt a technique originally used by Waters [30] in the context of IBE schemes that has been applied to lattice based signature [8] and encryption schemes [2]. Concretely, we instantiate the above MAC construction with a different secret key derivation function $f_\mathbf{s}(\mathbf{m}, \mathbf{b}) = \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$ (where $\mathbf{v} = h(\mathbf{m}, \mathbf{b})$ and $h(\cdot)$ is a pairwise independent hash). The drawback of our second construction is the larger keysize. Our security reduction uses a technique from [8, 2] based on encodings with full-rank differences (FRD) by Cramer and Damgard [10].

1.2 Efficiency

Figure 3 gives a rough comparison of our new protocol and MACs with the $\mathsf{HB}, \mathsf{HB}^+$ protocols and, as a reference, also the classical tree-based GGM construction [17]. The second row in the table specifies the security notion that is (provably) achieved under the $\mathsf{LPN}_{\tau,\ell}$ assumption. λ is a security parameter and n denotes the number of "repetitions". Typical parameters can be $\ell = 500, \lambda = 80, n = 250$. Computation complexity counts the number of binary operations over $\mathbb{F}_2$. Communication complexity counts the total length of all exchanged messages[2]. The last row in the table states the tightness of the security reduction, i.e. what exact security is achieved (ignoring constants and higher order terms) assuming the $\mathsf{LPN}_{\tau,\ell}$ problem is ε-hard.

The prover and verifier in the $\mathsf{HB}, \mathsf{HB}^+$ and our new protocols have to perform $\Theta(\ell \cdot n)$ basic binary operations, assuming the $\mathsf{LPN}_{\tau,\ell}$ problem (i.e., LPN with secrets of length ℓ) is hard. This seems optimal, as $\Theta(\ell)$ operations are necessary to compute the inner product which generates a single pseudorandom bit. We will thus consider an authentication protocol or MAC *efficient*, if it requires $O(\ell \cdot n)$ binary operations. Let us mention that one gets a length-doubling PRG under the $\mathsf{LPN}_{\tau,\ell}$ assumption with $\Theta(\ell^2)$ binary operations [12]. Via the classical GGM construction [17], we obtain a PRF and hence a MAC. This PRF, however, requires $\Theta(\ell^2 \cdot \lambda)$ operations per invocation (where λ is the size of the domain of the PRF) which is not very practical. (Recall that $\ell \approx 500$).

Communication vs. Key-Size. For all constructions except GGM, there is a natural trade-off between communication and key-size, where for any constant c ($1 \le c \le n$), we can decrease communication by a factor of c and increase key-size by the factor c (cf. the full version [1] for how exactly this can be done). For the first three protocols in the table, the choice of c does not affect the computational efficiency, but it does so for our MACs: to compute or verify a

Construction	Security	Complexity		Key-size	Reduction
		Communication	Computation		
HB [18]	passive (2 rnd)	$\ell \cdot n/c$	$\Theta(\ell \cdot n)$	$\ell \cdot c$	ε (tight)
HB^+ [19]	active (3 rnd)	$\ell \cdot n \cdot 2/\ c$	$\Theta(\ell \cdot n)$	$\ell \cdot 2 \cdot c$	$\sqrt{\varepsilon}$
AUTH § 3	active (2 rnd)	$\ell \cdot n \cdot 2.1/c$	$\Theta(\ell \cdot n)$	$\ell \cdot 4.2 \cdot c$	ε (tight)
MAC_1 § 4.1	MAC → MIM (2 rnd)	$\ell \cdot n \cdot 2.1/c$	$\Theta(\ell \cdot n) + \mathsf{PIP}$	$\ell \cdot 12.6 \cdot c$	$\sqrt{\varepsilon} \cdot Q$ ($\star$)
MAC_2 § 4.2	MAC → MIM (2 rnd)	$\ell \cdot n \cdot 1.1/c$	$\Theta(\ell \cdot n) + \mathsf{PIP}$	$\ell \cdot \lambda \cdot c$	$\varepsilon \cdot Q$
GGM [17]	PRF → MIM (2 rnd)	λ	$\Theta(\ell^2 \cdot \lambda)$	$\Theta(\ell)$	$\varepsilon \cdot \lambda$

Fig. 3. A comparison of our new authentication protocol and MACs with the HB, HB^+ protocols and the classical GGM construction. The trade-off parameter $c, 1 \le c \le n$ and the term PIP will be explained in the "Communication vs. Key-Size" paragraph below. ($\star$) See discussion in §4.

[2] For MACs, we consider the communication one incurs by constructing a MIM secure 2-round protocol from the MAC by having the prover compute the tag on a random challenge message.

tag one has to evaluate a pairwise independent permutation (PIP) on the entire tag of length $m := \Theta(\ell \cdot n/c)$.

The standard way to construct a PIP π over $\mathbb{Z}_{2^m}$ is to define $\pi(x) := a \cdot x + b \in \mathbb{F}_{2^m}$ for random $a, b \in \mathbb{F}_{2^m}$. Thus the computational cost of evaluating the PIP is one multiplication of two m bits values: the PIP term in the table accounts for this complexity. Asymptotically, such a multiplication takes only $O(m \log m \log\log m)$ time [28, 13], but for small m (like in our scheme) this will not be faster than using schoolbook multiplication, which takes $\Theta(m^2)$ time. For parameters $\ell = 500, n = 250$ and trade-off $c = n$ (which minimizes the tag-length m) we get $m \approx 1200$ for MAC_1 (i.e., $1200 = 2\ell$ plus some statistical security parameters) and $m \approx 600$ for MAC_2. Hence, depending on the parameters, the evaluation of the PIP may be the computational bottleneck of our MACs.

2 Definitions

2.1 Notation

We denote the set of integers modulo an integer $q \geq 1$ by $\mathbb{Z}_q$. We will use normal, bold and capital bold letters like x, $\mathbf{x}$, $\mathbf{X}$ to denote single elements, vectors and matrices over $\mathbb{Z}_q$, respectively. For a positive integer k, $[k]$ denotes the set $\{1, \ldots, k\}$; $[0]$ is the empty set. For $a, b \in \mathbb{R}$, $]a, b[= \{x \in \mathbb{R} \ ; \ a < x < b\}$. For a vector $\mathbf{x} \in \mathbb{Z}_q^m$, $|\mathbf{x}| = m$ denotes the length of $\mathbf{x}$; $\mathbf{wt}(\mathbf{x})$ denotes the Hamming weight of the vector $\mathbf{x}$, i.e. the number of indices $i \in \{1, \ldots, |\mathbf{x}|\}$ where $\mathbf{x}[i] \neq 0$. The bit-wise XOR of two binary vectors $\mathbf{x}$ and $\mathbf{y}$ is represented as $\mathbf{z} = \mathbf{x} \oplus \mathbf{y}$, where $\mathbf{z}[i] = \mathbf{x}[i] \oplus \mathbf{y}[i]$. For $\mathbf{v} \in \mathbb{Z}_2^m$ we denote by $\overline{\mathbf{v}}$ its inverse, i.e. $\overline{\mathbf{v}}[i] = 1 - \mathbf{v}[i]$ for all i. For two vectors $\mathbf{v} \in \mathbb{Z}_2^\ell$ and $\mathbf{x} \in \mathbb{Z}_q^\ell$, we denote by $\mathbf{x}_{\downarrow \mathbf{v}}$ the vector (of length $\mathbf{wt}(\mathbf{v})$) which is derived from $\mathbf{x}$ by deleting all the bits $\mathbf{x}[i]$ where $\mathbf{v}[i] = 0$. If $\mathbf{X} \in \mathbb{Z}_2^{\ell \times m}$ is a matrix, then $\mathbf{X}_{\downarrow \mathbf{v}}$ denotes the submatrix we get by deleting the ith row if $\mathbf{v}[i] = 0$. A function in λ is *negligible*, written $\mathrm{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. An algorithm $\mathcal{A}$ is *probabilistic polynomial time* (PPT) if $\mathcal{A}$ uses some randomness as part of its logic (i.e. $\mathcal{A}$ is probabilistic) and for any input $\mathbf{x} \in \{0,1\}^*$ the computation of $\mathcal{A}(\mathbf{x})$ terminates in at most $\mathrm{poly}(|\mathbf{x}|)$ steps.

2.2 Authentication Protocols

An authentication protocol is an interactive protocol executed between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, both PPT algorithms. Both hold a secret $\mathbf{x}$ (generated using a key-generation algorithm KG executed on the security parameter λ in unary) that has been shared in an initial phase. After the execution of the authentication protocol, $\mathcal{V}$ outputs either accept or reject. We say that the protocol has completeness error α if for all secret keys $\mathbf{x}$ generated by $\mathsf{KG}(1^\lambda)$, the honestly executed protocol returns reject with probability at most α.

Passive attacks. An authentication protocol is secure against *passive* attacks, if there exists no PPT adversary $\mathcal{A}$ that can make the verifier return accept with

non-negligible probability after (passively) observing any number of interactions between the verifier and prover.

ACTIVE ATTACKS. A stronger notion for authentication protocols is security against *active* attacks. Here the adversary $\mathcal{A}$ runs in two stages. First, she can interact with the honest prover a polynomial number of times (with concurrent executions allowed). In the second phase $\mathcal{A}$ interacts with the verifier only, and wins if the verifier returns accept. Here we only give the adversary one shot to convince the verifier[3]. An authentication protocol is (t, Q, ε)-*secure against active adversaries* if every PPT $\mathcal{A}$, running in time at most t and making Q queries to the honest prover, has probability at most ε to win the above game.

MAN-IN-THE-MIDDLE ATTACKS. The strongest standard security notion for authentication protocols is security against man-in-the-middle (MIM) attacks. Here the adversary can initially interact (concurrently) with any number of provers and – unlike in an active attacks – also verifiers. The adversary gets to learn the verifiers $\mathsf{accept}/\mathsf{reject}$ decisions. One can construct two-round authentication schemes which are secure against MIM attacks from basic cryptographic primitives like MACs, which we define next.

2.3 Message Authentication Codes

A message authentication code $\mathsf{MAC} = \{\mathsf{KG}, \mathsf{TAG}, \mathsf{VRFY}\}$ is a triple of algorithms with associated key space $\mathcal{K}$, message space $\mathcal{M}$, and tag space $\mathcal{T}$.

- Key Generation. The probabilistic key-generation algorithm KG takes as input a security parameter $\lambda \in \mathbb{N}$ (in unary) and outputs a secret key $K \in \mathcal{K}$.
- Tagging. The probabilistic authentication algorithm TAG takes as input a secret key $K \in \mathcal{K}$ and a message $\mathbf{m} \in \mathcal{M}$ and outputs an authentication tag $\phi \in \mathcal{T}$.
- Verification. The deterministic verification algorithm VRFY takes as input a secret key $K \in \mathcal{K}$, a message $\mathbf{m} \in \mathcal{M}$ and a tag $\phi \in \mathcal{T}$ and outputs $\{\mathsf{accept}, \mathsf{reject}\}$.

If the TAG algorithm is deterministic one does not have to explicitly define VRFY, since it is already defined by the TAG algorithm as $\mathsf{VRFY}(K, \mathbf{m}, \phi) = \mathsf{accept}$ iff $\mathsf{TAG}(K, \mathbf{m}) = \phi$.

COMPLETENESS. We say that MAC has completeness error α if for all $\mathbf{m} \in \mathcal{M}$ and $\lambda \in \mathbb{N}$

$$\Pr[\mathsf{VRFY}(K, \mathbf{m}, \phi) = \mathsf{reject}\ ; K \leftarrow \mathsf{KG}(1^\lambda)\ ,\ \phi \leftarrow \mathsf{TAG}(K, \mathbf{m})] \leq \alpha.$$

SECURITY. The standard security notion for a MAC is unforgeability under a chosen message attack (uf-cma). We denote by $\mathsf{Adv}^{\mathrm{uf-cma}}_{\mathsf{MAC}}(\mathcal{A}, \lambda, Q)$, the advantage of the adversary $\mathcal{A}$ in forging a message under a chosen message attack for

[3] By using a hybrid argument one can show that this implies security even if the adversary can interact in $k \geq 1$ independent instances concurrently (and wins if the verifier accepts in at least one instance). The use of the hybrid argument looses a factor of k in the security reduction.

MAC when used with security parameter λ. Formally this is the probability that the following experiment outputs 1.

Experiment $\mathbf{Exp}^{\mathrm{uf-cma}}_{\mathsf{MAC}}(\mathcal{A}, \lambda, Q)$
$K \leftarrow \mathsf{KG}(1^\lambda)$
Invoke $\mathcal{A}^{\mathsf{TAG}(K,\cdot),\mathsf{VRFY}(K,\cdot,\cdot)}$ who can make up to Q queries to $\mathsf{TAG}(K,\cdot)$ and $\mathsf{VRFY}(K,\cdot,\cdot)$.
Output 1 if $\mathcal{A}$ made a query $(\mathbf{m}, \phi)$ to $\mathsf{VRFY}(K,\cdot,\cdot)$ where
1. $\mathsf{VRFY}(K, \mathbf{m}, \phi) = \mathsf{accept}$
2. $\mathcal{A}$ did not already make the query $\mathbf{m}$ to $\mathsf{TAG}(K,\cdot)$
Output 0 otherwise.

We say that MAC is (t, Q, ε)-secure against uf-cma adversaries if for any $\mathcal{A}$ running in time t in the experiment above, we have $\mathsf{Adv}^{\mathrm{uf-cma}}_{\mathsf{MAC}}(\mathcal{A}, \lambda, Q) \leq \varepsilon$.

2.4 Hard Learning Problems

Let Ber_τ be the Bernoulli distribution over $\mathbb{Z}_2$ with parameter (bias) $\tau \in\ ]0, 1/2[$ (i.e., $\Pr[x = 1] = \tau$ if $x \leftarrow \mathsf{Ber}_\tau$). For $\ell \geq 1$, Ber^ℓ_τ denotes the distribution over $\mathbb{Z}^\ell_2$ where each vector consists of ℓ independent samples drawn from Ber_τ. Given a secret $\mathbf{x} \in \mathbb{Z}^\ell_2$ and $\tau \in]0, \frac{1}{2}[$, we write $\Lambda_{\tau,\ell}(\mathbf{x})$ for the distribution over $\mathbb{Z}^\ell_2 \times \mathbb{Z}_2$ whose samples are obtained by choosing a vector $\mathbf{r} \xleftarrow{\$} \mathbb{Z}^\ell_2$ and outputting $(\mathbf{r}, \mathbf{r}^\mathsf{T} \cdot \mathbf{x} \oplus e)$ with $e \xleftarrow{\$} \mathsf{Ber}_\tau$.

The LPN assumption, formally defined below, states that it is hard to distinguish $\Lambda_{\tau,\ell}(\mathbf{x})$ (with a random secret $\mathbf{x} \in \mathbb{Z}^\ell_2$) from the uniform distribution.

Definition 1 (Learning Parity with Noise). *The (decisional)* $\mathsf{LPN}_{\tau,\ell}$ *problem is* (t, Q, ε)*-hard if for every distinguisher* D *running in time* t *and making* Q *queries,*

$$\left|\Pr\left[\mathbf{x} \xleftarrow{\$} \mathbb{Z}^\ell_2\ :\ D^{\Lambda_{\tau,\ell}(\mathbf{x})} = 1\right] - \Pr\left[\ D^{U_{\ell+1}} = 1\right]\right| \leq \varepsilon.$$

Below we define the (seemingly) stronger *subspace* LPN assumption (SLPN for short) recently introduced in [26]. Here the adversary can ask for inner products not only with the secret $\mathbf{x}$, but even with $\mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}$ where $\mathbf{A}$ and $\mathbf{b}$ can be adaptively chosen, but $\mathbf{A}$ must have sufficiently large rank. For minimal dimension $d \leq \ell$, a secret $\mathbf{x} \in \mathbb{Z}^\ell_2$ and $\mathbf{A} \in \mathbb{Z}^{\ell\times\ell}_2$, $\mathbf{b} \in \mathbb{Z}^\ell_2$, we define the distribution

$$\Gamma_{\tau,\ell,d}(\mathbf{x}, \mathbf{A}, \mathbf{b}) = \begin{cases} \perp & \text{if } \mathsf{rank}(\mathbf{A}) < d \\ \Lambda_{\tau,\ell}(\mathbf{A} \cdot \mathbf{x} \oplus \mathbf{b}) & \text{otherwise} \end{cases}$$

and let $\Gamma_{\tau,\ell,d}(\mathbf{x},\cdot,\cdot)$ denote the oracle which on input $\mathbf{A}, \mathbf{b}$ outputs a sample from $\Gamma_{\tau,\ell,d}(\mathbf{x}, \mathbf{A}, \mathbf{b})$.

Definition 2 (Subspace LPN). *Let* $\ell, d \in \mathbb{Z}$ *where* $d \leq \ell$*. The (decisional)* $\mathsf{SLPN}_{\tau,\ell,d}$ *problem is* (t, Q, ε)*-hard if for every distinguisher* D *running in time* t *and making* Q *queries,*

$$\left|\Pr\left[\mathbf{x} \xleftarrow{\$} \mathbb{Z}^\ell_2\ :\ D^{\Gamma_{\tau,\ell,d}(\mathbf{x},\cdot,\cdot)} = 1\right] - \Pr\left[\ D^{U_{\ell+1}(\cdot,\cdot)} = 1\right]\right| \leq \varepsilon,$$

where $U_{\ell+1}(\cdot,\cdot)$ *on input* $(\mathbf{A},\mathbf{b})$ *outputs a sample of* $U_{\ell+1}$ *if* $\mathsf{rank}(\mathbf{A}) \geq d$ *and* $\perp$ *otherwise.*

The following proposition states that the subspace LPN problem mapping to dimension $d+g$ is almost as hard as the standard LPN problem with secrets of length d. The hardness gap is exponentially small in g.

Proposition 1 (From [26]). *For any* $\ell, d, g \in \mathbb{Z}$ *(where* $\ell \geq d+g$*), if the* $\mathsf{LPN}_{\tau,d}$ *problem is* (t, Q, ε)*-hard then the* $\mathsf{SLPN}_{\tau,\ell,d+g}$ *problem is* (t', Q, ε')*-hard where*

$$t' = t - \mathrm{poly}(\ell, Q) \qquad \varepsilon' = \varepsilon + 2Q/2^{g+1}.$$

For some of our constructions, we will only need a weaker version of the $\mathsf{SLPN}_{\tau,\ell,d}$ problem that we call sub**set** LPN. As the name suggests, here the adversary does not ask for inner products with $\mathbf{A}\cdot\mathbf{x}\oplus\mathbf{b}$ for any $\mathbf{A}$ (of rank $\geq d$), but only with *subsets* of $\mathbf{x}$ (of size $\geq d$). It will be convenient to explicitly define this special case. For $\mathbf{x},\mathbf{v} \in \mathbb{Z}_2^\ell$, let $\mathsf{diag}(\mathbf{v}) \in \mathbb{Z}_2^{\ell\times\ell}$ denote the zero matrix with $\mathbf{v}$ in the diagonal, and let

$$\Gamma^*_{\tau,\ell,d}(\mathbf{x},\mathbf{v}) := \Gamma_{\tau,\ell,d}(\mathbf{x},\mathsf{diag}(\mathbf{v}),0^\ell) = \begin{cases} \perp & \text{if } \mathbf{wt}(\mathbf{v}) < d \\ \Lambda_{\tau,\ell}(\mathbf{x}\wedge\mathbf{v}) & \text{otherwise.} \end{cases}$$

Definition 3 (Subset LPN). *Let* $\ell, d \in \mathbb{Z}$ *where* $d \leq \ell$*. The* $\mathsf{SLPN}^*_{\tau,\ell,d}$ *problem is* (t, Q, ε)*-hard if for every distinguisher* D *running in time* t *and making* Q *queries,*

$$\left|\Pr\left[\mathbf{x} \xleftarrow{\$} \mathbb{Z}_2^\ell \ : \ D^{\Gamma^*_{\tau,\ell,d}(\mathbf{x},\cdot)} = 1\right] - \Pr\left[\ D^{U_{\ell+1}(\cdot)} = 1\right]\right| \leq \varepsilon,$$

where $U_{\ell+1}(\cdot)$ *on input* $\mathbf{v}$ *(where* $\mathbf{wt}(\mathbf{v}) \geq d$*) outputs a sample of* $U_{\ell+1}$ *and* $\perp$ *otherwise.*

Remark 1. $\Gamma^*_{\tau,\ell,d}(\mathbf{x},\mathbf{v})$ samples are of the form $(\mathbf{r}, \mathbf{r}^\mathsf{T}_{\downarrow\mathbf{v}}\cdot\mathbf{x}_{\downarrow\mathbf{v}} \oplus e) \in \mathbb{Z}_2^{\ell+1}$, where $e \xleftarrow{\$} \mathsf{Ber}_\tau$. To compute the inner product only $\mathbf{r}_{\downarrow\mathbf{v}} \in \mathbb{Z}_2^{\mathbf{wt}(\mathbf{v})}$ is needed, the remaining bits $\mathbf{r}_{\downarrow\overline{\mathbf{v}}} \in \mathbb{Z}_2^{\ell-\mathbf{wt}(\mathbf{v})}$ are irrelevant. We use this observation to improve the communication complexity (for protocols) or tag length (for MACs), by using "compressed" samples of the form $(\mathbf{r}_{\downarrow\mathbf{v}}, \mathbf{r}^\mathsf{T}_{\downarrow\mathbf{v}}\cdot\mathbf{x}_{\downarrow\mathbf{v}} \oplus e) \in \mathbb{Z}_2^{\mathbf{wt}(\mathbf{v})+1}$.

3 Two-Round Authentication with Active Security

In this section we describe our new 2-round authentication protocol and prove its active security under the hardness of the $\mathsf{SLPN}^*_{\tau,2\ell,d}$ problem, where $d = \ell/(2+\gamma)$ for some constant $\gamma > 0$. (Concretely, $\gamma = 0.1$ should do for all practical purposes).

– Public parameters. The authentication protocol has the following public parameters, where τ, τ' are constants and ℓ, n depend on the security parameter λ.

$\ell \in \mathbb{N}$	length of the secret key $\mathbf{s} \in \mathbb{Z}_2^{2\ell}$
$\tau \in]0, 1/2[$	parameter of the Bernoulli error distribution Ber_τ
$\tau' = 1/4 + \tau/2$	acceptance threshold
$n \in \mathbb{N}$	number of parallel repetitions (we require $n \leq \ell/2$)

– Key Generation. Algorithm $\mathsf{KG}(1^\lambda)$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{2\ell}$ and returns $\mathbf{s}$ as the secret key.

– Authentication Protocol. The 2-round authentication protocol with prover $\mathcal{P}_{\tau,n}$ and verifier $\mathcal{V}_{\tau',n}$ is given in Figure 4.

Prover $\mathcal{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$		Verifier $\mathcal{V}_{\tau',n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$
	$\xleftarrow{\mathbf{v}}$	$\mathbf{v} \xleftarrow{\$} \{\mathbf{x} \in \mathbb{Z}_2^{2\ell} : \mathbf{wt}(\mathbf{x}) = \ell\}$
if $\mathbf{wt}(\mathbf{v}) \neq \ell$ abort		
$\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$; $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n$		
$\mathbf{z} := \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e} \in \mathbb{Z}_2^n$	$\xrightarrow{(\mathbf{R},\mathbf{z})}$	if $\mathsf{rank}(\mathbf{R}) \neq n$ reject
		if $\mathbf{wt}(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}}) > n \cdot \tau'$ reject, else accept

Fig. 4. Two-round authentication protocol AUTH with active security from the LPN assumption

Theorem 1. *For any constant $\gamma > 0$, let $d = \ell/(2+\gamma)$. If the $\mathsf{SLPN}^*_{\tau,2\ell,d}$ problem is (t, nQ, ε)-hard then the authentication protocol from Figure 4 is (t', Q, ε')-secure against active adversaries, where for constants $c_\gamma, c_\tau > 0$ that depend only on γ and τ respectively,*

$$t' = t - \mathrm{poly}(Q, \ell) \qquad \varepsilon' = \varepsilon + Q \cdot 2^{-c_\gamma \cdot \ell} + 2^{-c_\tau \cdot n} = \varepsilon + 2^{-\Theta(n)} \ .$$

The protocol has completeness error $2^{-c'_\tau \cdot n}$ where $c'_\tau > 0$ depends only on τ.

3.1 Proof of Completeness

For any $n \in \mathbb{N}$, $\tau \in]0, 1/2[$, let

$$\alpha_{\tau,n} := \Pr[\mathbf{wt}(\mathbf{e}) > n \cdot \tau' : \ \mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n] = 2^{-c''_\tau \cdot n} \tag{3.1}$$

denote the probability that n independent Bernoulli samples with bias τ contain more than a $\tau' := 1/4 + \tau/2$ fraction of 1's. The last equality in eq.(3.1) follows from the Hoeffding bound, where the constant $c''_\tau > 0$ depends only on τ.

We now prove that the authentication protocol has completeness error $\alpha \leq 2^{-\ell+n} + \alpha_{\tau,n}$. The verifier performs the following two checks. In the first verification step, the verifier rejects if the random matrix $\mathbf{R}$ does not have full rank. In the full version [1] we prove that the probability of this event is $\leq 2^{-n}$. Now, let $\mathbf{e} := \mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}}$ denote the noise added by $\mathcal{P}_{\tau,n}$. Then, in the second verification step, the verifier rejects if $\mathbf{wt}(\mathbf{e}) > n \cdot \tau'$. From equation 3.1, we have that this happens with probability $\alpha_{\tau,n}$. This completes the proof of completeness.

3.2 Proof of Security

We first define some terms that will be used later in the security proof. For a constant $\gamma > 0$, let $d = \ell/(2+\gamma)$ (as in Theorem 1). Let $\alpha'_{\ell,d}$ denote the probability that a random substring of length ℓ chosen from a string of length 2ℓ with Hamming weight ℓ, has a Hamming weight less than d. Using the fact that the expected Hamming weight is $\ell/2 = d(1+\gamma/2) = d(1+\Theta(1))$, one can show that there exists a constant $c_\gamma > 0$ (only depending on γ), such that

$$\alpha'_{\ell,d} := \frac{\sum_{i=0}^{d-1} \binom{\ell}{i}\binom{\ell}{\ell-i}}{\binom{2\ell}{\ell}} \leq 2^{-c_\gamma \cdot \ell}. \tag{3.2}$$

For $\tau' = 1/4 + \tau/2$, let $\alpha''_{\tau',n}$ denote the probability that a random bitstring $\mathbf{y} \in \mathbb{Z}_2^n$ has Hamming weight $\mathbf{wt}(\mathbf{y}) \leq n \cdot \tau'$. From the Hoeffding bound, it follows that there exists a constant $c_\tau > 0$ (only depending on τ), such that

$$\alpha''_{\tau',n} := 2^{-n} \cdot \sum_{i=0}^{\lfloor n\cdot\tau' \rfloor} \binom{n}{i} \leq 2^{-c_\tau \cdot n}. \tag{3.3}$$

We now prove security of the authentication protocol. Consider an oracle $\mathcal{O}$ which is either the subset LPN oracle $\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)$ or $U_{2\ell+1}(\cdot)$, as defined in Definition 3. We will construct an adversary $\mathcal{B}^{\mathcal{O}}$ that uses $\mathcal{A}$ (who breaks the active security of AUTH with advantage ε') in a black-box way such that:

$$\Pr[\mathcal{B}^{\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)} \to 1] \geq \varepsilon' - Q \cdot \alpha'_{\ell,d} \qquad \text{and} \qquad \Pr[\mathcal{B}^{U_{2\ell+1}(\cdot)} \to 1] \leq \alpha''_{\tau',n}\ .$$

Thus $\mathcal{B}^{\mathcal{O}}$ can distinguish between the two oracles with advantage $\varepsilon := \varepsilon' - Q \cdot \alpha'_{\ell,d} - \alpha''_{\tau',n}$ as claimed in the statement of the Theorem. Below we define $\mathcal{B}^{\mathcal{O}}$.

Setup. Initially, $\mathcal{B}^{\mathcal{O}}$ samples

$$\mathbf{x}^* \xleftarrow{\$} \mathbb{Z}_2^{2\ell}, \qquad \mathbf{v}^* \xleftarrow{\$} \{\mathbf{y} \in \mathbb{Z}_2^{2\ell} : \mathbf{wt}(\mathbf{y}) = \ell\}.$$

The intuition of our simulation below is as follows. Let us first assume $\mathcal{O}$ is a subset LPN oracle $\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)$ with secret $\mathbf{x}$. In the first phase we have to produce answers $(\mathbf{R},\mathbf{z})$ to a query $\mathbf{v} \in \{\mathbf{y} \in \mathbb{Z}_2^{2\ell} : \mathbf{wt}(\mathbf{y}) = \ell\}$ by $\mathcal{A}$. The simulated answers have exactly the same distribution as the answers of an honest prover $\mathcal{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$ where

$$\mathbf{s} = (\mathbf{x}^* \wedge \mathbf{v}^*) \oplus (\mathbf{x} \wedge \overline{\mathbf{v}}^*) \tag{3.4}$$

Thus one part of $\mathbf{s}$'s bits come from $\mathbf{x}^*$, and the other part is from the unknown secret $\mathbf{x}$ (for which we use the oracle $\mathcal{O}$). In the second phase we give $\mathcal{A}$ the challenge $\mathbf{v}^*$. As $\mathbf{s}_{\downarrow\mathbf{v}^*} = (\mathbf{x}^* \wedge \mathbf{v}^*)_{\downarrow\mathbf{v}^*}$ is known, we will be able to verify if $\mathcal{A}$ outputs a valid forgery.

If $\mathcal{O}$ is the random oracle $U_{2\ell+1}(\cdot)$, then after the first phase $\mathbf{s}_{\downarrow\mathbf{v}^*} = (\mathbf{x}^* \wedge \mathbf{v}^*)_{\downarrow\mathbf{v}^*}$ is information theoretically hidden, and thus $\mathcal{A}$ cannot come up with a valid forgery but with exponentially small probability.

First phase. In the first phase $\mathcal{B}^{\mathcal{O}}$ invokes $\mathcal{A}$ who expects access to $\mathcal{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$. We now specify how $\mathcal{B}^{\mathcal{O}}$ samples the answer $(\mathbf{R}, \mathbf{z})$ to a query $\mathbf{v} \in \{\mathbf{y} \in \mathbb{Z}_2^{2\ell} : \mathbf{wt}(\mathbf{y}) = \ell\}$ made by $\mathcal{A}$. Let

$$\mathbf{u}^* := \mathbf{v} \wedge \mathbf{v}^* \qquad \mathbf{u} := \mathbf{v} \wedge \overline{\mathbf{v}}^*$$

1. $\mathcal{B}^{\mathcal{O}}$ queries its oracle n times on the input $\mathbf{u}$. If the oracle's output is $\perp$ (which happens iff $\mathbf{wt}(\mathbf{u}) < d$), $\mathcal{B}^{\mathcal{O}}$ outputs 0 and stops. Otherwise let $\hat{\mathbf{R}}_1 \in \mathbb{Z}_2^{2\ell \times n}, \mathbf{z}_1 \in \mathbb{Z}_2^n$ denote the n outputs of the oracle.
2. Sample $\hat{\mathbf{R}}_0 \xleftarrow{\$} \mathbb{Z}_2^{2\ell \times n}$ and set $\mathbf{z}_0 = \hat{\mathbf{R}}_0^{\mathsf{T}} \cdot (\mathbf{x}^* \wedge \mathbf{u}^*)$.
3. Return $(\mathbf{R} = \hat{\mathbf{R}}_{\downarrow \mathbf{v}} \in \mathbb{Z}_2^{\ell \times n}, \mathbf{z} = \mathbf{z}_0 \oplus \mathbf{z}_1 \in \mathbb{Z}_2^n)$, where $\hat{\mathbf{R}}$ is uniquely determined by requiring $\hat{\mathbf{R}}_{\downarrow \mathbf{v}^*} = \hat{\mathbf{R}}_0$ and $\hat{\mathbf{R}}_{\downarrow \overline{\mathbf{v}}^*} = \hat{\mathbf{R}}_1$.

Second phase. Eventually, $\mathcal{A}$ enters the second phase of the active attack, expecting a challenge from $\mathcal{V}_{\tau',n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$.

1. $\mathcal{B}^{\mathcal{O}}$ forwards $\mathbf{v}^*$ as the challenge to $\mathcal{A}$.
2. $\mathcal{A}$ answers with some $(\mathbf{R}^*, \mathbf{z}^*)$.
3. $\mathcal{B}^{\mathcal{O}}$ checks if

$$\mathsf{rank}(\mathbf{R}^*) = n \quad \text{and} \quad \mathbf{wt}(\mathbf{z}^* \oplus \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{x}^*_{\downarrow \mathbf{v}^*}) \leq n \cdot \tau'. \tag{3.5}$$

The output is 1 if both checks succeed and 0 otherwise.

Claim 2. $\Pr[\mathcal{B}^{U_{2\ell+1}(\cdot)} \to 1] \leq \alpha''_{\tau',n}$.

Proof (of Claim). If $\mathbf{R}^*$ does not have full rank then $\mathcal{B}$ outputs 0 by definition. Therefore, we now consider the case where $\mathsf{rank}(\mathbf{R}^*) = n$.

The answers $(\mathbf{R}, \mathbf{z})$ that the adversary $\mathcal{A}$ obtains from $\mathcal{B}^{U_{2\ell+1}(\cdot)}$ are independent of $\mathbf{x}^*$ (i.e., $\mathbf{z} = \mathbf{z}_0 \oplus \mathbf{z}_1$ is uniform as $\mathbf{z}_1$ is uniform). Since $\mathbf{x}^*_{\downarrow \mathbf{v}^*}$ is uniformly random and $\mathbf{R}^*$ has full rank, the vector

$$\mathbf{y} := \mathbf{R}^{*\mathsf{T}} \cdot \mathbf{x}^*_{\downarrow \mathbf{v}^*} \oplus \mathbf{z}^*$$

is uniformly random over $\mathbb{Z}_2^n$. Thus the probability that the second verification in eq. (3.5) does not fail is $\Pr[\mathbf{wt}(\mathbf{y}) \leq n \cdot \tau'] = \alpha''_{\tau',n}$.

Claim 3. $\Pr[\mathcal{B}^{\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)} \to 1] \geq \varepsilon' - Q \cdot \alpha'_{\ell,d}$.

Proof (of Claim). We split the proof in two parts. First we show that $\mathcal{B}$ outputs 1 with probability $\geq \varepsilon'$ if the subset LPN oracle accepts subsets of arbitrary small size (and does not simply output $\perp$ on inputs $\mathbf{v}$ where $\mathbf{wt}(\mathbf{v}) < d$), i.e.,

$$\Pr[\mathcal{B}^{\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)} \to 1] \geq \varepsilon'. \tag{3.6}$$

Then we'll upper bound the gap between the probability that $\mathcal{B}$ outputs 1 in the above case and the probability that $\mathcal{B}$ outputs 1 when given access to the oracle that we are interested in as:

$$\left|\Pr[\mathcal{B}^{\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)} \to 1] - \Pr[\mathcal{B}^{\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)} \to 1]\right| \leq Q \cdot \alpha'_{\ell,d}. \tag{3.7}$$

The claim then follows by the triangle inequality from the two equations above.

Eq. (3.6) holds as:

– The answers $(\mathbf{R}, \mathbf{z})$ that $\mathcal{B}^{\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)}$ gives to $\mathcal{A}$'s queries in the first phase of the attack have *exactly* the same distribution as what $\mathcal{A}$ would get when interacting with an honest prover $\mathcal{P}_{\tau,n}(\mathbf{s} \in \mathbb{Z}_2^{2\ell})$ where the "simulated" secret $\mathbf{s}$ is defined in eq.(3.4).

To see this, recall that on a query $\mathbf{v}$ from $\mathcal{A}$, $\mathcal{B}^{\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)}$ must compute n SLPN samples $(\hat{\mathbf{R}}, \mathbf{z} = \hat{\mathbf{R}}^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{v}) \oplus \mathbf{e})$ and then forward the compressed version of this samples to $\mathcal{A}$ (that is, $(\mathbf{R}, \mathbf{z} = \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e})$ where $\mathbf{R} = \hat{\mathbf{R}}_{\downarrow \mathbf{v}}$, cf. Remark 1). We next show that the $\mathbf{z}$ computed by $\mathcal{B}$ indeed have exactly this distribution. In the first step, $\mathcal{B}$ queries its oracle with $\mathbf{u} = \mathbf{v} \wedge \overline{\mathbf{v}}^*$ and obtains noisy inner products $(\hat{\mathbf{R}}_1, \mathbf{z}_1)$ with the part of $\mathbf{s}_{\downarrow \mathbf{v}}$ that contains only bits from $\mathbf{x}$, i.e.,

$$\mathbf{z}_1 = \hat{\mathbf{R}}_1^\mathsf{T} \cdot (\mathbf{x} \wedge \mathbf{u}) \oplus \mathbf{e} = \hat{\mathbf{R}}_1^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{u}) \oplus \mathbf{e}.$$

In the second step, $\mathcal{B}$ samples n inner products $(\hat{\mathbf{R}}_0, \mathbf{z}_0)$ (with no noise) with the part of $\mathbf{s}_{\downarrow \mathbf{v}}$ that contains only bits from the known $\mathbf{x}^*$, i.e.,

$$\mathbf{z}_0 = \hat{\mathbf{R}}_0^\mathsf{T} \cdot (\mathbf{x}^* \wedge \mathbf{u}^*) = \hat{\mathbf{R}}_0^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{u}^*).$$

In the third step, $\mathcal{B}$ then generates $(\hat{\mathbf{R}}, \hat{\mathbf{R}}^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{v}) \oplus \mathbf{e})$ from the previous values where $\hat{\mathbf{R}}$ is defined by $\hat{\mathbf{R}}_{\downarrow \mathbf{v}^*} = \hat{\mathbf{R}}_0$ and $\hat{\mathbf{R}}_{\downarrow \overline{\mathbf{v}}^*} = \hat{\mathbf{R}}_1$. Using $\mathbf{v} = \mathbf{u} \oplus \mathbf{u}^*$, we get

$$\begin{aligned} \mathbf{z} &= \mathbf{z}_0 \oplus \mathbf{z}_1 \\ &= \hat{\mathbf{R}}_0^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{u}^*) \oplus \hat{\mathbf{R}}_1^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{u}) \oplus \mathbf{e} \\ &= \hat{\mathbf{R}}^\mathsf{T} \cdot (\mathbf{s} \wedge \mathbf{v}) \oplus \mathbf{e} \end{aligned}$$

– The challenge $\mathbf{v}^*$ sent to $\mathcal{A}$ in the second phase of the active attack is uniformly random (even given the entire view so far), and therefore has the same distribution as a challenge in an active attack.
– $\mathcal{B}^{\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)}$ outputs 1 if eq.(3.5) holds, which is exactly the case when $\mathcal{A}$'s response to the challenge was valid. By assumption this probability is at least ε'.

This concludes the proof of Eq. (3.6). It remains to prove eq.(3.7). Note that $\Gamma^*_{\tau,2\ell,0}(\mathbf{x},\cdot)$ behaves exactly like $\Gamma^*_{\tau,2\ell,d}(\mathbf{x},\cdot)$ as long as one never makes a query $\mathbf{v}$ where $\mathbf{wt}(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$.

Since $\mathbf{v}^* \xleftarrow{\$} \{\mathbf{y} \in \mathbb{Z}_2^{2\ell} : \mathbf{wt}(\mathbf{y}) = \ell\}$, for any $\mathbf{v}$, the probability that $\mathbf{wt}(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$ is (by definition) $\alpha'_{\ell,d}$ as defined in eq.(3.2). Using the union bound, we can upper bound the probability that $\mathbf{wt}(\mathbf{v} \wedge \overline{\mathbf{v}}^*) < d$ for any of the Q different $\mathbf{v}$'s chosen by the adversary as $Q \cdot \alpha'_{\ell,d}$.

3.3 Avoid Checking

One disadvantage of the protocol in Figure 4, compared to HB style protocols, is the necessity to check whether the messages exchanged have the right from:

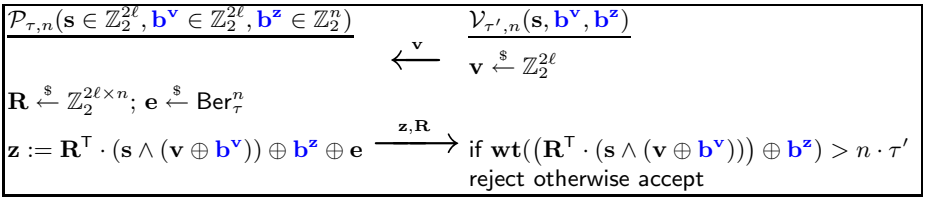


Fig. 5. By blinding the values $\mathbf{v}, \mathbf{z}$ with secret random vectors $\mathbf{b^v}, \mathbf{b^z}$ we can avoid checking whether $\mathbf{wt}(\mathbf{v}) = \ell$ and $\mathsf{rank}(\mathbf{R}) = n$ as in the protocol from Figure 4

the prover checks if $\mathbf{v}$ has weight ℓ, while the verifier must make the even more expensive check whether $\mathbf{R}$ has full rank. Eliminating such verification procedures can be particularly useful if for example the prover is an RFID chip where even the simple verification that a vector has large weight is expensive. We note that it is possible to eliminate these checks by blinding the exchanged messages $\mathbf{v}$ and $\mathbf{z}$ using random vectors $\mathbf{b^v} \in \mathbb{Z}_2^{2\ell}$ and $\mathbf{b^z} \in \mathbb{Z}_2^n$ respectively, as shown in Figure 5. The security and completeness of this protocol is basically the same as for the protocol in Figure 5. The security proof is also very similar and is therefore omitted.

4 Message Authentication Codes

In this section, we construct two message authentication codes whose security can be reduced to the LPN assumption. Our first construction is based on the 2-round authentication protocol from Section 3. We prove that if the LPN problem is ε-hard, then no adversary making Q queries can forge a MAC with probability more than $\Theta(\sqrt{\varepsilon} \cdot Q)$. However, the construction has the disadvantage that one needs to fix the hardness of the LPN problem at the time of the construction, c.f. Remark 2. Our second construction has no such issues and achieves better security $\Theta(\varepsilon \cdot Q)$. The efficiency of this construction is similar to that of the first construction, but a larger key is required.

4.1 First Construction

Recall the 2-round authentication protocol from Section 3. In the protocol the verifier chooses a random challenge subset $\mathbf{v}$. To turn this interactive protocol into a MAC, we will compute this $\mathbf{v}$ from the message $\mathbf{m}$ to be authenticated as $\mathbf{v} = \mathsf{C}(h(\mathbf{m}, \mathbf{b}))$, where h is a pairwise independent hash function, $\mathbf{b} \in \mathbb{Z}_2^\nu$ is some fresh randomness and C is some encoding scheme. The code C is fixed and public, while the function h is part of the secret key. The authentication tag ϕ is computed in the same manner as the prover's answer in the authentication protocol. That is, we sample a random matrix $\mathbf{R} \in \mathbb{Z}_2^{\ell \times n}$ and compute a noisy inner product $\mathbf{z} := \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}$, where $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n$. We note that using $(\mathbf{R}, \mathbf{z})$ as an authentication tag would not be secure, and we need to blind these values.

This is done by applying an (almost) pairwise independent permutation (PIP) π – which is part of the secret key – to $(\mathbf{R}, \mathbf{z}, \mathbf{b}) \in \mathbb{Z}_2^{\ell \times n+n+\nu}$.

CONSTRUCTION. The message authentication code $\mathsf{MAC}_1 = \{\mathsf{KG}, \mathsf{TAG}, \mathsf{VRFY}\}$ with associated message space $\mathcal{M}$ is defined as follows.

- Public parameters. MAC_1 has the following public parameters.

ℓ, τ, τ', n	as in the authentication protocol from Section 3
$\mu \in \mathbb{N}$	output length of the hash function
$\nu \in \mathbb{N}$	length of the randomness
$\mathsf{C} : \mathbb{Z}_2^{\mu} \to \mathbb{Z}_2^{2\ell}$	encoding, where $\forall\ \mathbf{x} \neq \mathbf{x}' \in \mathbb{Z}_2^{\mu}$ we have $\mathbf{wt}(\mathsf{C}(\mathbf{x})) = \ell$ and $\mathbf{wt}(\mathsf{C}(\mathbf{x}) \oplus \mathsf{C}(\mathbf{x}')) \geq 0.9\ell$.

- Key generation. Algorithm $\mathsf{KG}(1^\lambda)$ samples $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_2^{2\ell}$, an (almost) pairwise independent hash function $h : \mathcal{M} \times \mathbb{Z}_2^{\nu} \to \mathbb{Z}_2^{\mu}$ and a pairwise independent permutation π over $\mathbb{Z}_2^{\ell \times n+n+\nu}$. It returns $K = (\mathbf{s}, h, \pi)$ as the secret key.
- Tagging. Given secret key $K = (\mathbf{s}, h, \pi)$ and message $\mathbf{m} \in \mathcal{M}$, algorithm TAG proceeds as follows.
  1. $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_2^{\nu}$, $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n$
  2. $\mathbf{v} := \mathsf{C}(h(\mathbf{m}, \mathbf{b})) \in \mathbb{Z}_2^{2\ell}$
  3. Return $\phi := \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}} \oplus \mathbf{e}, \mathbf{b})$
- Verification. On input a secret-key $K = (\mathbf{s}, h, \pi)$, message $\mathbf{m} \in \mathcal{M}$ and tag ϕ, algorithm VRFY proceeds as follows.
  1. Parse $\pi^{-1}(\phi)$ as $(\mathbf{R} \in \mathbb{Z}_2^{\ell \times n}, \mathbf{z} \in \mathbb{Z}_2^n, \mathbf{b} \in \mathbb{Z}_2^{\nu})$. If $\mathsf{rank}(\mathbf{R}) \neq n$, then return reject
  2. $\mathbf{v} := \mathsf{C}(h(\mathbf{m}, \mathbf{b}))$
  3. If $\mathbf{wt}(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}_{\downarrow \mathbf{v}}) > n \cdot \tau'$ return reject, otherwise return accept

Theorem 4. *For $\mu = \nu \in \mathbb{N}$, a constant $\gamma > 0$ and $d := \ell/(2+\gamma)$, if the $\mathsf{SLPN}^*_{\tau,2\ell,d}$ problem is (t, nQ, ε)-hard then MAC_1 is (t', Q, ε')-secure against uf-cma adversaries, where*

$$t' \approx t, \qquad \varepsilon = \min\left\{ \varepsilon'/2 - \frac{Q^2}{2^{\mu-2}} \ ,\ \frac{\varepsilon'}{2^{\mu+1}} - 2^{-\Theta(n)} \right\} .$$

MAC_1 has completeness error $2^{-c_\tau \cdot n}$ where $c_\tau > 0$ depends only on τ.

Corollary 1. *Choosing μ s.t. $2^\mu = \frac{Q^2 \cdot 2^4}{\varepsilon'}$ in the above theorem, we get $\varepsilon = \min\{\varepsilon'/4, (\varepsilon')^2/(2^5 Q^2) - 2^{-\Theta(n)}\}$. The 2nd term is the minimum here, and solving for ε' gives*

$$\varepsilon' := \sqrt{32} \cdot Q \cdot \sqrt{\varepsilon + 2^{-\Theta(n)}}. \tag{4.1}$$

Remark 2 (about μ). Note that to get security as claimed in the above corollary, we need to choose μ as a function of Q and ε such that $2^\mu \approx Q^2 \cdot 2^4/\varepsilon'$ for ε' as in eq.(4.1). Of course we can just fix Q (as an upper bound to the number of queries made by the adversary) and ε (as our guess on the actual hardness of $\mathsf{SLPN}^*_{\tau,2\ell,d}$). But a too conservative guess on μ (i.e. choosing μ too small) will result in a construction whose security is worse than what is claimed in the above corollary. A too generous guess on the other hand will make the security

reduction meaningless (we don't have any actual attacks on the MAC for large μ though).

We now give an intuition for the proof of Theorem 4. For space reasons, a full proof will only be given in the full version of this paper [1]. Every query $(\mathbf{m}, \phi)$ to VRFY and query $\mathbf{m}$ to TAG defines a subset $\mathbf{v}$ (as computed in the second step in the definitions of both VRFY and TAG). We say that a forgery $(\mathbf{m}, \phi)$ is "fresh" if the $\mathbf{v}$ contained in $(\mathbf{m}, \phi)$ is different from all $\mathbf{v}$'s contained in all the previous VRFY and TAG queries. The proof makes a case distinction and uses a different reduction for the two cases where the forgery found by the adversary is more likely to be fresh, or more likely to be non-fresh. In both cases we consider a reduction $\mathcal{B}^{\mathcal{O}}$ which has access to either a uniform oracle $\mathcal{O} = U$ or a subset LPN oracle $\mathcal{O} = \Gamma^*$. $\mathcal{B}^{\mathcal{O}}$ uses an adversary $\mathcal{A}$ who can find forgeries for the MAC to distinguish those cases and thus break the subset LPN assumption. In the first case, where the first forgery is likely to be **non-fresh**, we can show (using the fact that a pairwise independent permutation is used to blind the tag) that if $\mathcal{B}^{\mathcal{O}}$'s oracle is $\mathcal{O} = U$, even a computationally unbounded $\mathcal{A}$ cannot come up with a message/tag pair $(\mathbf{m}, \phi)$ that contains a non-fresh $\mathbf{v}$. Thus we can distinguish the cases $\mathcal{O} = U$ and $\mathcal{O} = \Gamma^*$ by just observing if $\mathcal{A}$ ever makes a VRFY query $(\mathbf{m}, \phi)$ that contains a non-fresh $\mathbf{v}$ (even without being able to tell if $(\mathbf{m}, \phi)$ is valid or not).

If the forgery found by $\mathcal{A}$ is more likely to be **fresh**, we can use a similar argument as in the proof of our authentication protocol in the last section. An additional difficulty here is that the reduction has to guess the fresh $\mathbf{v} \in \mathbb{Z}_2^\mu$ contained in the first forgery and cannot choose it as in the protocol. This is the reason why the reduction looses a factor 2^μ.

4.2 Second Construction

We now give the construction of another MAC based on the hardness of the LPN problem. The main difference to MAC_1 from the last subsection is the way we generate the values $\mathbf{s}(\mathbf{v})$. In the new construction, we define $\mathbf{s}(\mathbf{v}) := \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$, where each $\mathbf{s}_i$ is a part of the secret key. The construction uses ideas from Waters' IBE scheme [30], and parts of the security reduction use simulation tricks from [8, 2] that we need to adapt to the binary case.

Construction. The message authentication code $\mathsf{MAC}_2 = \{\mathsf{KG}, \mathsf{TAG}, \mathsf{VRFY}\}$ with associated message space $\mathcal{M}$ is defined as follows.

- Public parameters. MAC_2 has the following public parameters.
  ℓ, τ, τ', n as in the authentication protocol from Section 3
  $\mu \in \mathbb{N}$ output length of the hash function
  $\nu \in \mathbb{N}$ length of the randomness
- Key generation. Algorithm $\mathsf{KG}(1^\lambda)$ samples $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_2^\ell$ (for $0 \le i \le \mu$) and chooses a pairwise independent hash function $h : \mathcal{M} \times \mathbb{Z}_2^\nu \to \mathbb{Z}_2^\mu$, as well as a pairwise independent permutation π over $\mathbb{Z}_2^{\ell \times n + n + \nu}$. It returns $K = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$ as the secret key.

- Tagging. Given secret key $K = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$ and message $\mathbf{m} \in \mathcal{M}$, algorithm TAG proceeds as follows.
  1. $\mathbf{R} \xleftarrow{\$} \mathbb{Z}_2^{\ell \times n}$, $\mathbf{b} \xleftarrow{\$} \mathbb{Z}_2^{\nu}$, $\mathbf{e} \xleftarrow{\$} \mathsf{Ber}_\tau^n$
  2. $\mathbf{v} := h(\mathbf{m}, \mathbf{b})$
  3. $\mathbf{s}(\mathbf{v}) := \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$
  4. Return $\phi := \pi(\mathbf{R}, \mathbf{R}^\mathsf{T} \cdot \mathbf{s}(\mathbf{v}) \oplus \mathbf{e}, \mathbf{b})$
- Verification. On input a secret-key $K = (\mathbf{s}_0, \ldots, \mathbf{s}_\mu, h, \pi)$, message $\mathbf{m} \in \mathcal{M}$ and tag ϕ, algorithm VRFY proceeds as follows.
  1. Parse $\pi^{-1}(\phi)$ as $(\mathbf{R} \in \mathbb{Z}_2^{\ell \times n}, \mathbf{z} \in \mathbb{Z}_2^n, \mathbf{b} \in \mathbb{Z}_2^\nu)$. If $\mathsf{rank}(\mathbf{R}) \neq n$, then return reject
  2. $\mathbf{v} := h(m, \mathbf{b})$
  3. $\mathbf{s}(\mathbf{v}) := \mathbf{s}_0 \oplus \bigoplus_{i:\mathbf{v}[i]=1} \mathbf{s}_i$
  4. If $\mathbf{wt}(\mathbf{z} \oplus \mathbf{R}^\mathsf{T} \cdot \mathbf{s}(\mathbf{v})) > n \cdot \tau'$ return reject, otherwise return accept

Theorem 5. *If the* $\mathsf{SLPN}_{\tau,\ell,\ell}$ *problem is* (t, nQ, ε)*-hard, then* MAC_2 *is* (t', Q, ε')*-secure against uf-cma adversaries, where*

$$t' \approx t \qquad \varepsilon \ = \min \left\{ \varepsilon'/2 - \frac{Q^2}{2^{\mu-2}}, \frac{\varepsilon'}{4Q} - 2^{-\Theta(n)} \right\}.$$

MAC_2 *has completeness error* $2^{-c_\tau \cdot n}$ *where* c_τ *only depends on* τ*.*

We now give an intuition for the proof of Theorem 5. For space reasons, a full proof will only be given in the full version of this paper [1]. Similar to the proof of Theorem 4, we distinguish fresh and non-fresh forgeries. Here the new and interesting case is the fresh forgery. The idea is that in the reduction to the SLPN problem we define the function $\mathbf{s}(\mathbf{v}) = \mathbf{A}(\mathbf{v}) \cdot \mathbf{s} \oplus \mathbf{b}(\mathbf{v})$ (where $\mathbf{s}$ is the LPN secret) such that the following holds with non-negligible probability: (i) for each $\mathbf{v}_i$ from the TAG queries, $\mathbf{A}(\mathbf{v}_i)$ has full rank ℓ and hence the tags can be simulated using the provided $\Gamma_{\tau,\ell,\ell}(\mathbf{s}, \cdot, \cdot)$ oracle; (ii) for the first fresh forgery we have $\mathbf{A}(\mathbf{v}) = 0$ such that $\mathbf{s}(\mathbf{v})$ is independent of $\mathbf{s}$ and the reduction can check the forgery's correctness. The above two properties allow to maintain the simulation. The setup of the function $\mathbf{s}(\cdot)$ is the crucial step and here we adapt a technique recently introduced by Boyen [8] based on homomorphic encodings with full-rank differences that allows us to arbitrarily control the probability that the above simulation works.

Acknowledgements

Krzysztof would like to thank Vadim Lyubashevsky for many interesting discussions on LPN while being in Tel Aviv and Eyjafjallajökull for making this stay possible.

References

[1] The full version of this paper will be posted on the Cryptology ePrint Archive, http://eprint.iacr.org/

[2] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[3] Berlekamp, E., McEliece, R., van Tilborg, H.: On the inherent intractability of certain coding problems. IEEE Transactions on Information Theory 24(3), 384–386 (1978)

[4] Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer, Heidelberg (1994)

[5] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. In: 32nd ACM STOC, pp. 435–440. ACM Press, New York (May 2000)

[6] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. J. ACM 50(4), 506–519 (2003)

[7] Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

[8] Boyen, X.: Lattice mixing and vanishing trapdoors: A framework for fully secure short signatures and more. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 499–517. Springer, Heidelberg (2010)

[9] Bringer, J., Chabanne, H., Dottax, E.: HB^{++}: a lightweight authentication protocol secure against some attacks. In: SecPerU, pp. 28–33 (2006)

[10] Cramer, R., Damgard, I.: On the amortized complexity of zero-knowledge protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009)

[11] Duc, D.N., Kim, K.: Securing HB+ against GRS man-in-the-middle attack. In: SCIS (2007)

[12] Fischer, J.-B., Stern, J.: An efficient pseudo-random generator provably as secure as syndrome decoding. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 245–255. Springer, Heidelberg (1996)

[13] Fürer, M.: Faster integer multiplication. SIAM J. Comput. 39(3), 979–1005 (2009)

[14] Gilbert, H., Robshaw, M., Sibert, H.: An active attack against HB+ - a provably secure lightweight authentication protocol. Cryptology ePrint Archive, Report 2005/237 (2005), http://eprint.iacr.org/

[15] Gilbert, H., Robshaw, M.J.B., Seurin, Y.: Good variants of hB^{+} are hard to find. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 156–170. Springer, Heidelberg (2008)

[16] Gilbert, H., Robshaw, M.J.B., Seurin, Y.: $HB^{\#}$: Increasing the security and efficiency of HB^{+}. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 361–378. Springer, Heidelberg (2008)

[17] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM 33, 792–807 (1986)

[18] Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 52–66. Springer, Heidelberg (2001)

[19] Juels, A., Weis, S.A.: Authenticating pervasive devices with human protocols. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 293–308. Springer, Heidelberg (2005)

[20] Katz, J., Shin, J.S.: Parallel and concurrent security of the HB and HB$^+$ protocols. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 73–87. Springer, Heidelberg (2006)
[21] Katz, J., Shin, J.S., Smith, A.: Parallel and concurrent security of the HB and HB+ protocols. Journal of Cryptology 23(3), 402–421 (2010)
[22] Kearns, M.J.: Efficient noise-tolerant learning from statistical queries. J. ACM 45(6), 983–1006 (1998)
[23] Levieil, É., Fouque, P.-A.: An improved LPN algorithm. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 348–359. Springer, Heidelberg (2006)
[24] Munilla, J., Peinado, A.: HB-MP: A further step in the HB-family of lightweight authentication protocols. Computer Networks 51(9), 2262–2267 (2007)
[25] Ouafi, K., Overbeck, R., Vaudenay, S.: On the security of hB$^\#$ against a man-in-the-middle attack. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 108–124. Springer, Heidelberg (2008)
[26] Pietrzak, K.: Subspace LWE (2010) (manuscript) http://homepages.cwi.nl/~pietrzak/publications/SLWE.pdf
[27] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC, pp. 84–93. ACM Press, New York (2005)
[28] Schönhage, A., Strassen, V.: Schnelle Multiplikation grosser Zahlen. Computing 7 (1971)
[29] Van De Graaf, J.: Towards a formal definition of security for quantum protocols. PhD thesis, Monreal, P.Q., Canada, AAINQ35648 (1998)
[30] Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
[31] Watrous, J.: Zero-knowledge against quantum attacks. SIAM J. Comput. 39(1), 25–58 (2009)

Making NTRU as Secure as Worst-Case Problems over Ideal Lattices

Damien Stehlé[1] and Ron Steinfeld[2]

[1] CNRS, Laboratoire LIP (U. Lyon, CNRS, ENS Lyon, INRIA, UCBL),
46 Allée d'Italie, 69364 Lyon Cedex 07, France
damien.stehle@gmail.com
http://perso.ens-lyon.fr/damien.stehle
[2] Centre for Advanced Computing - Algorithms and Cryptography,
Department of Computing, Macquarie University, NSW 2109, Australia
ron.steinfeld@mq.edu.au
http://web.science.mq.edu.au/~rons

Abstract. NTRUEncrypt, proposed in 1996 by Hoffstein, Pipher and Silverman, is the fastest known lattice-based encryption scheme. Its moderate key-sizes, excellent asymptotic performance and conjectured resistance to quantum computers could make it a desirable alternative to factorisation and discrete-log based encryption schemes. However, since its introduction, doubts have regularly arisen on its security. In the present work, we show how to modify NTRUEncrypt to make it provably secure in the standard model, under the assumed quantum hardness of standard worst-case lattice problems, restricted to a family of lattices related to some cyclotomic fields. Our main contribution is to show that if the secret key polynomials are selected by rejection from discrete Gaussians, then the public key, which is their ratio, is statistically indistinguishable from uniform over its domain. The security then follows from the already proven hardness of the R-LWE problem.

Keywords: Lattice-based cryptography, NTRU, provable security.

1 Introduction

NTRUEncrypt, devised by Hoffstein, Pipher and Silverman, was first presented at the Crypto'96 rump session [14]. Although its description relies on arithmetic over the polynomial ring $\mathbb{Z}_q[x]/(x^n-1)$ for n prime and q a small integer, it was quickly observed that breaking it could be expressed as a problem over Euclidean lattices [6]. At the ANTS'98 conference, the NTRU authors gave an improved presentation including a thorough assessment of its practical security against lattice attacks [15]. We refer to [13] for an up-to-date account on the past 15 years of security and performance analyses. Nowadays, NTRUEncrypt is generally considered as a reasonable alternative to the encryption schemes based on integer factorisation and discrete logarithm over finite fields and elliptic curves, as testified by its inclusion in the IEEE P1363 standard [17]. It is also often considered as the most viable post-quantum public-key encryption (see, e.g., [30]).

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 27–47, 2011.

In parallel to a rising number of attacks and practical improvements on NTRUEncrypt the (mainly) theoretical field of provably secure lattice-based cryptography has steadily been developed. It originated in 1996 with Ajtai's acclaimed worst-case to average-case reduction [2], leading to a collision-resistant hash function that is as hard to break as solving several worst-case problems defined over lattices. Ajtai's average-case problem is now referred to as the *Small Integer Solution* problem (SIS). Another major breakthrough in this field was the introduction in 2005 of the *Learning with Errors* problem (LWE) by Regev [31]: LWE is both hard on the average (worst-case lattice problems quantumly reduce to it), and sufficiently flexible to allow for the design of cryptographic functions. In the last few years, many cryptographic schemes have been introduced that are provably as secure as LWE and SIS are hard (and thus provably secure, assuming the worst-case hardness of lattice problems). These include CPA and CCA secure encryption schemes, identity-based encryption schemes, digital signatures, *etc* (see [31,28,11,5,1] among others, and the surveys [24,32]).

The main drawback of cryptography based on LWE and SIS is its limited efficiency. A key typically contains a random matrix defined over $\mathbb{Z}_q$ for a small q, whose dimension is linear in the security parameter; consequently, the space and time requirements seem bound to be at least quadratic with respect to the security parameter. In 2002, Micciancio [22] succeeded in restricting SIS to structured matrices while preserving a worst-case to average-case reduction. The worst-case problem is a restriction of a standard lattice problem to the specific family of cyclic lattices. The structure of Micciancio's matrices allows for an interpretation in terms of arithmetic in the ring $\mathbb{Z}_q[x]/(x^n - 1)$, where n is the dimension of the worst-case lattices and q is a small prime. Micciancio's construction leads to a family of pre-image resistant hash functions, with complexity quasi-linear in n. Peikert, Rosen, Lyubashevsky and Micciancio [**?**,18] later suggested to change the ring to $\mathbb{Z}_q[x]/\Phi$ with a Φ that is irreducible over the rationals, sparse, and with small coefficients (e.g., $\Phi = x^n + 1$ for n a power of 2). The resulting hash function was proven collision-resistant under the assumed hardness of the modified average-case problem, called Ideal-SIS. The latter was itself proven at least as hard as the restrictions of standard worst-case lattice problems to a specific class of lattices (called ideal lattices). In 2009, Stehlé *et al.* [34] introduced a structured variant of LWE, which they proved as hard as Ideal-SIS (under a quantum reduction), and allowed for the design of an asymptotically efficient CPA-secure encryption scheme. In an independent concurrent work, Lyubashevsky *et al.* [20] proposed a ring variant of LWE, called R-LWE, whose great flexibility allows for more natural (and efficient) cryptographic constructions.

Our Results. The high efficiency and industrial standardization of NTRUEncrypt strongly motivate a theoretically founded study of its security. Indeed, in the absence of such a study so far, its security has remained in doubt over the last 15 years since its publication. In this paper, we address this problem. We prove that a mild modification of NTRUEncrypt is CPA-secure, assuming the quantum hardness of standard worst-case problems over ideal lattices (for $\Phi = x^n + 1$ with n a power of 2). The NTRUEncrypt modifications are summarized below. We stress

that our main goal in this paper is to provide a firm theoretical grounding for the security of NTRUEncrypt in the asymptotic sense. We leave to future work the consideration of practical issues, in particular the selection of concrete parameters for given security levels. As for other lattice-based schemes, the latter requires evaluation of security against practical lattice reduction attacks, which is out of the scope of the current work.

Our main contribution is the modification and analysis of the key generation algorithm. The secret key consists of two sparse polynomials of degrees $< n$ and coefficients in $\{-1, 0, 1\}$. The public key is their quotient in $\mathbb{Z}_q[x]/(x^n - 1)$ (the denominator is resampled if it is not invertible). A simple information-theoretic argument shows that the public key cannot be uniformly distributed in the whole ring. It may be possible to extend the results of [4] to show that it is "well-spread" in the ring, but it still would not suffice for showing its cryptographic pseudorandomness, which seems necessary for exploiting the established hardness of R-LWE. To achieve a public key distribution statistically close to uniform, we sample the secret key polynomials according to a discrete Gaussian with standard deviation $\approx q^{1/2}$. An essential ingredient, which could be of independent interest, is a new regularity result for the ring $R_q := \mathbb{Z}_q[x]/(x^n + 1)$ when the polynomial $x^n + 1$ (with n a power of 2) has n factors modulo prime q: given $a_1, \ldots, a_m$ uniform in R_q, we would like $\sum_{i \le m} s_i a_i$ to be within exponentially small statistical distance to uniformity, with small random s_i's and small m. Note that a similar regularity bound can be obtained with an FFT-based technique recently developed by Lyubashevsky, Peikert and Regev [21]. An additional difficulty in the public-key 'uniformity' proof, which we handle via an inclusion-exclusion argument, is that we need the s_i's to be invertible in R_q (the denominator of the public key is one such s_i): we thus sample according to a discrete Gaussian, and reject the sample if it is not invertible.

Brief Comparison of NTRUEncrypt and Its Provably Secure Variant

Let R_{NTRU} be the ring $\mathbb{Z}[x]/(x^n - 1)$ with n prime. Let q be a medium-size integer, typically a power of 2 of the same order of magnitude as n. Finally, let $p \in R_{\text{NTRU}}$ with small coefficients, co-prime with q and such that the plaintext space R_{NTRU}/p is large. Typically, one may take $p = 3$ or $p = x + 2$.

The NTRUEncrypt secret key is a pair of polynomials $(f, g) \in R_{\text{NTRU}}^2$ that are sampled randomly with large prescribed proportions of zeros, and with their other coefficients belonging to $\{-1, 1\}$. For improved decryption efficiency, one may choose f such that $f = 1 \bmod p$. With high probability, the polynomial f is invertible modulo q and modulo p, and if that is the case, the public-key is $h = pg/f \bmod q$ (otherwise, the key generation process is restarted). To encrypt a message $M \in R_{\text{NTRU}}/p$, one samples a random element $s \in R_{\text{NTRU}}$ of small Euclidean norm and computes the ciphertext $C = hs + M \bmod q$. The following procedure allows the owner of the secret key to decrypt:

- Compute $fC \bmod q$. If C was properly generated, this gives $pgs + fM \bmod q$. Since p, g, s, f, M have small coefficients, it can be expected that after reduction modulo q the obtained representative is $pgs + fM$ (in R_{NTRU}).

- Reduce the latter modulo p. This should provide $fM \bmod p$.
- Multiply the result of the previous step by the inverse of f modulo p (this step becomes vacuous if $f = 1 \bmod p$).

Note that the encryption process is probabilistic, and that decryption errors can occur for some sets of parameters. However, it is possible to arbitrarily decrease the decryption error probability, and even to eliminate it completely.

In order to achieve CPA-security we make a few modifications to the original `NTRUEncrypt` (which preserve its quasi-linear time and space complexity):

1. We replace R_{NTRU} by $R = \mathbb{Z}[x]/(x^n + 1)$ with n a power of 2. We will exploit the irreducibility of $x^n + 1$ and the fact that R is the ring of integers of a cyclotomic number field.
2. We choose a prime $q \leq \mathcal{P}oly(n)$ such that $f = x^n + 1 \bmod q$ has n distinct linear factors (i.e., $q = 1 \bmod 2n$). This allows us to use the search to decision reduction for R-LWE with ring $R_q := R/q$ (see [20]), and also to take $p = 2$.
3. We sample f and g from discrete Gaussians over R, rejecting the samples that are not invertible in R_q. We show that $f/g \bmod q$ is essentially uniformly distributed over the set of invertible elements of R_q. We may also choose $f = pf' + 1$ with f' sampled from a discrete Gaussian, to simplify decryption.
4. We add a small error term e in the encryption: $C = hs + pe + M \bmod q$, with s and e sampled from the R-LWE error distribution. This allows us to derive CPA security from the hardness of a variant of R-LWE (which is similar to the variant of LWE from [3, Se. 3.1]).

Work in Progress and Open Problems

Our study is restricted to the sequence of rings $\mathbb{Z}[x]/\Phi_n$ with $\Phi_n = x^n + 1$ with n a power of 2. An obvious drawback is that this does not allow for much flexibility on the choice of n (in the case of NTRU, the degree was assumed prime, which provides more freedom). The R-LWE problem is known to be hard when Φ_n is cyclotomic [20]. We chose to restrict ourselves to cyclotomic polynomials of order a power of 2 because it makes the error generation of R-LWE more efficient, and the description of the schemes simpler to follow. Our results are likely to hold for more general rings than those we considered. An interesting choice could be the cyclotomic rings of prime order (i.e., $\Phi_n = (x^n - 1)/(x - 1)$ with n prime) as these are large subrings of the NTRU rings (and one might then be able to show that the hardness carries over to the NTRU rings).

An interesting open problem is to obtain a CCA secure variant of our scheme in the standard model, while maintaining its efficiency (within constant factors). The selection of concrete parameters based on practical security estimates for the worst-case SVP in ideal lattices or the average-case hardness of R-LWE/Ideal-SIS is also left as a future work.

The authors of `NTRUEncrypt` also proposed a signature scheme based on a similar design. The history of `NTRUSign` started with `NSS` in 2001 [16]. Its development has been significantly more hectic and controversial, with a series

of cryptanalyses and repairs (see the survey [13]). In a work in progress, we construct a variant of NTRUSign with unforgeability related to the worst-case hardness of standard problems over ideal lattices, in the random oracle model. Our construction modifies the NTRUSign key generation and adapts the GPV signature scheme [11] to this setting.

Like NTRUEncrypt, Gentry's somewhat homomorphic scheme [9] also has ciphertexts consisting of a single ring element. It also admits a security proof under the assumed quantum hardness of standard worst-case problems over ideal lattices [10]. Our security analysis for the modified NTRUEncrypt scheme allows encrypting and decrypting $\Omega(n)$ plaintext bits for $\widetilde{O}(n)$ bit operations, while achieving security against $2^{g(n)}$-time attacks, for any $g(n)$ that is $\Omega(\log n)$ and $o(n)$, assuming the worst-case hardness of $\mathcal{P}oly(n)$-Ideal-SVP against $2^{O(g(n))}$-time quantum algorithms. The latter assumption is believed to be valid for any $g(n) = o(n)$. Gentry's analysis from [10,8] can be generalized to handle $2^{g(n)}$-time attacks while encrypting and decrypting $O(g(n))$ plaintext bits for $\widetilde{O}(n)$ bit operations, under the assumed hardness of $2^{\Omega(g(n))}$-Ideal-SVP against $2^{O(g(n))}$-time quantum algorithms. The latter assumption is known to be invalid when $g(n) = \widetilde{\Omega}(\sqrt{n})$ (using [33]), thus limiting the attacker's strength the analysis can handle. On the other hand, Gentry's scheme allows homomorphic additions and multiplications, whereas ours seems restricted to additions. Our scheme and Gentry's seem to be closely related, and we leave to future work the further investigation of this relation.

NOTATION. We denote by $\rho_\sigma(\boldsymbol{x})$ (resp. ν_σ) the standard n-dimensional Gaussian function (resp. distribution) with center $\mathbf{0}$ and variance σ, i.e., $\rho_\sigma(\boldsymbol{x}) = \exp(-\pi\|\boldsymbol{x}\|^2/\sigma^2)$ (resp. $\nu_\sigma(\boldsymbol{x}) = \rho_\sigma(\boldsymbol{x})/\sigma^n$). We denote by $\mathrm{Exp}(\mu)$ the exponential distribution on $\mathbb{R}$ with mean μ and by $U(E)$ the uniform distribution over a finite set E . If D_1 and D_2 are two distributions on discrete domain E, their statistical distance is $\Delta(D_1; D_2) = \frac{1}{2}\sum_{x\in E}|D_1(x) - D_2(x)|$. We write $z \hookleftarrow D$ when the random variable z is sampled from the distribution D.

REMARK. Due to space limitations, some proofs have been omitted; they may be found in the full version of this paper, available on the authors' web pages.

2 A Few Background Results

A (full-rank) *lattice* is a set of the form $L = \sum_{i\le n}\mathbb{Z}\boldsymbol{b}_i$, where the $\boldsymbol{b}_i$'s are linearly independent vectors in $\mathbb{R}^n$. The integer n is called the *lattice dimension*, and the $\boldsymbol{b}_i$'s are called a *basis* of L. The *minimum* $\lambda_1(L)$ (resp. $\lambda_1^\infty(L)$) is the Euclidean (resp. infinity) norm of any shortest vector of $L \setminus \mathbf{0}$. If $B = (\boldsymbol{b}_i)_i$ is a basis matrix of L, the *fundamental parallelepiped* of B is the set $\mathcal{P}(B) = \{\sum_{i\le n} c_i\boldsymbol{b}_i : c_i \in [0,1)\}$. The volume $|\det B|$ of $\mathcal{P}(B)$ is an invariant of the lattice L which we denote by $\det L$. Minkowski's theorem states that $\lambda_1(L) \le \sqrt{n}(\det L)^{1/n}$. More generally, the k-th *minimum* $\lambda_k(L)$ for $k \le n$ is defined as the smallest r such that L contains $\ge k$ linearly independent vectors of norm $\le r$. The *dual* of L is the lattice $\widehat{L} = \{\boldsymbol{c} \in \mathbb{R}^n : \forall i, \langle \boldsymbol{c}, \boldsymbol{b}_i\rangle \in \mathbb{Z}\}$.

For a lattice $L \subseteq \mathbb{R}^n$, $\sigma > 0$ and $\boldsymbol{c} \in \mathbb{R}^n$, we define the *lattice Gaussian distribution* of support L, deviation σ and center $\boldsymbol{c}$ by $D_{L,\sigma,\boldsymbol{c}}(\boldsymbol{b}) = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{b})}{\rho_{\sigma,\boldsymbol{c}}(L)}$, for any $\boldsymbol{b} \in L$. We will omit the subscript $\boldsymbol{c}$ when it is $\boldsymbol{0}$. We extend the definition of $D_{L,\sigma,\boldsymbol{c}}$ to any $M \subseteq L$ (not necessarily a sublattice), by setting $D_{M,\sigma,\boldsymbol{c}}(\boldsymbol{b}) = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{b})}{\rho_{\sigma,\boldsymbol{c}}(M)}$. For $\delta > 0$, we define the *smoothing parameter* $\eta_\delta(L)$ as the smallest $\sigma > 0$ such that $\rho_{1/\sigma}(\widehat{L} \setminus \boldsymbol{0}) \leq \delta$. It quantifies how large σ needs to be for $D_{L,\sigma,\boldsymbol{c}}$ to behave like a continuous Gaussian. We will typically consider $\delta = 2^{-n}$.

Lemma 1 ([23, Le. 3.3]). *For any full-rank lattice $L \subseteq \mathbb{R}^n$ and $\delta \in (0,1)$, we have $\eta_\delta(L) \leq \sqrt{\ln(2n(1+1/\delta))/\pi} \cdot \lambda_n(L)$.*

Lemma 2 ([27, Le. 3.5]). *For any full-rank lattice $L \subseteq \mathbb{R}^n$ and $\delta \in (0,1)$, we have $\eta_\delta(L) \leq \sqrt{\ln(2n(1+1/\delta))/\pi}/\lambda_1^\infty(\widehat{L})$.*

Lemma 3 ([23, Le. 4.4]). *For any full-rank lattice $L \subseteq \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $\delta \in (0,1)$ and $\sigma \geq \eta_\delta(L)$, we have $\Pr_{\boldsymbol{b} \hookleftarrow D_{L,\sigma,\boldsymbol{c}}}[\|\boldsymbol{b}\| \geq \sigma\sqrt{n}] \leq \frac{1+\delta}{1-\delta} 2^{-n}$.*

Lemma 4 ([11, Cor. 2.8]). *Let $L' \subseteq L \subseteq \mathbb{R}^n$ be full-rank lattices. For any $\boldsymbol{c} \in \mathbb{R}^n$, $\delta \in (0, 1/2)$ and $\sigma \geq \eta_\delta(L')$, we have $\Delta(D_{L,\sigma,\boldsymbol{c}} \bmod L'; U(L/L')) \leq 2\delta$.*

Lemma 5 ([11, Th. 4.1]). *There exists a polynomial-time algorithm that takes as input any basis $(\boldsymbol{b}_i)_i$ of any lattice $L \subseteq \mathbb{Z}^n$ and $\sigma = \omega(\sqrt{\log n}) \max \|\boldsymbol{b}_i\|$ (resp. $\sigma = \Omega(\sqrt{n}) \max \|\boldsymbol{b}_i\|$), and returns samples from a distribution whose statistical distance to $D_{L,\sigma}$ is negligible (resp. exponentially small) with respect to n.*

The most famous lattice problem is SVP. Given a basis of a lattice L, it aims at finding a shortest vector in $L \setminus \boldsymbol{0}$. It can be relaxed to γ-SVP by asking for a non-zero vector that is no longer than $\gamma(n)$ times a solution to SVP, for a prescribed function $\gamma(\cdot)$. It is believed that no subexponential quantum algorithm solves the computational variants of γ-SVP in the worst case, for any $\gamma \leq \mathcal{P}oly(n)$. The smallest γ which is known to be achievable in polynomial time is exponential, up to poly-logarithmic factors in the exponent ([33,25]).

Ideal Lattices and Algebraic Number Theory

Ideal lattices. Let n a power of 2 and $\Phi = x^n + 1$ (which is irreducible over $\mathbb{Q}$). Let R be the ring $\mathbb{Z}[x]/\Phi$. An (integral) ideal I of R is a subset of R closed under addition and multiplication by arbitrary elements of R. By mapping polynomials to the vectors of their coefficients, we see that an ideal $I \neq 0$ corresponds to a full-rank sublattice of $\mathbb{Z}^n$: we can thus view I as both a lattice and an ideal. An *ideal lattice* for Φ is a sublattice of $\mathbb{Z}^n$ that corresponds to a non-zero ideal $I \subseteq R$. The *algebraic norm* $\mathcal{N}(I)$ is the cardinality of the additive group R/I. It is equal to $\det I$, where I is regarded as a lattice. Any non-zero ideal I of R satisfies $\lambda_n(I) = \lambda_1(I)$. In the following, an ideal lattice will implicitly refer to a Φ-ideal lattice.

By restricting SVP (resp. γ-SVP) to instances that are ideal lattices, we obtain Ideal-SVP (resp. γ-Ideal-SVP). The latter is implicitly parameterized by the

sequence of polynomials $\Phi_n = x^n + 1$, where n is restricted to powers of 2. No algorithm is known to perform non-negligibly better for (γ-)Ideal-SVP than for (γ-)SVP.

PROPERTIES OF THE RING R. For $v \in R$ we denote by $\|v\|$ its Euclidean norm (as a vector). We define the multiplicative *expansion factor* $\gamma_\times(R)$ by $\gamma_\times(R) = \max_{u,v \in R} \frac{\|u \times v\|}{\|u\| \cdot \|v\|}$. For our choice of Φ, we have $\gamma_\times(R) = \sqrt{n}$ (see [9, p. 174]).

Since Φ is the $2n$-th cyclotomic polynomial, the ring R is exactly the maximal order (i.e., the ring of integers) of the cyclotomic field $\mathbb{Q}[\zeta] \cong \mathbb{Q}[x]/\Phi =: K$, where $\zeta \in \mathbb{C}$ is a primitive $2n$-th root of unity. We denote by $(\sigma_i)_{i \le n}$ the canonical complex embeddings: We can choose $\sigma_i : P \mapsto P(\zeta^{2i+1})$ for $i \le n$. For any α in $\mathbb{Q}[\zeta]$, we define its T_2-norm by $T_2(\alpha)^2 = \sum_{i \le n} |\sigma_i(\alpha)|^2$ and its algebraic norm by $\mathcal{N}(\alpha) = \prod_{i \le n} |\sigma_i(\alpha)|$. The arithmetic-geometric inequality gives $\mathcal{N}(\alpha)^{2/n} \le \frac{1}{n} T_2(\alpha)^2$. Also, for the particular cyclotomic fields we are considering, the polynomial norm (the norm of the coefficient vector of α when expressed as an element of K) satisfies $\|\alpha\| = \frac{1}{\sqrt{n}} T_2(\alpha)$. We also use the fact that for any $\alpha \in R$, we have $|\mathcal{N}(\alpha)| = \det \langle \alpha \rangle$, where $\langle \alpha \rangle$ is the ideal of R generated by α. For simplicity, we will try to use the polynomial terminology wherever possible.

Let q be a prime number such that Φ has n distinct linear factors modulo q (i.e., $q = 1 \bmod 2n$): $\Phi = \prod_{i \le n} \Phi_i = \prod_{i \le n} (x - \phi_i) \bmod q$. Let $R_q = R/qR = \mathbb{Z}_q[x]/\Phi$. Dirichlet's theorem on arithmetic progressions implies that infinitely such primes exist. Furthermore, Linnik's theorem asserts that the smallest such q is $\mathcal{P}oly(n)$, and much effort has been spent to decrease this bound (the current record seems to be $O(n^{5.2})$, see [35]). Furthermore, we can write ϕ_i as r^i, where r is a primitive $(2n)$-th root of unity modulo q. This implies that the Chinese Remainder Theorem in R_q provides a natural fast Discrete Fourier Transform, and thus multiplication of elements of R_q can be performed within $O(n \log n)$ additions and multiplications modulo q (see [7, Ch. 8], [19, Se. 2.1]).

The R-LWE Problem

For $s \in R_q$ and ψ a distribution in R_q, we define $A_{s,\psi}$ as the distribution obtained by sampling the pair $(a, as+e)$ with $(a, e) \hookleftarrow U(R_q) \times \psi$. The Ring Learning With Errors problem (R-LWE) was introduced by Lyubashevsky *et al.* [20] and shown hard for specific error distributions ψ. These are slightly technical to define (see below), but for the present work, the important facts to be remembered are that the samples are small (see Lemma 6), and can be obtained in quasi-linear time.

The error distributions ψ that we use are an adaptation of those introduced in [20]. They are sampled from a family of distributions $\overline{\Upsilon}_\alpha$ that we now define. For $\boldsymbol{\sigma} \in \mathbb{R}^n$ with positive coordinates, we define the ellipsoidal Gaussian $\rho_{\boldsymbol{\sigma}}$ as the row vector of independent Gaussians $(\rho_{\sigma_1}, \ldots, \rho_{\sigma_n})$, where $\sigma_i = \sigma_{i+n/2}$ for $1 \le i \le n/2$. As we want to define R-LWE in the polynomial expression of R rather than with the so-called "space H" of [20], we apply a matrix transformation to the latter Gaussians. We define a sample from $\rho'_{\boldsymbol{\sigma}}$ as a sample from $\rho_{\boldsymbol{\sigma}}$, multiplied first (from the right) by $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ i & -i \end{pmatrix} \otimes \mathrm{Id}_{n/2} \in \mathbb{C}^{n \times n}$, and

second by $V = \frac{1}{n}\left(\zeta^{-(2j+1)k}\right)_{0\leq j,k<n}$. Note that vector multiplication by matrix V corresponds to a complex discrete Fourier transform, and can be performed in $O(n \log n)$ complex-valued arithmetic operations with the Cooley-Tukey FFT. Moreover, it is numerically extremely stable: if all operations are performed with a precision of $p = \Omega(\log n)$ bits, then the computed output vector $fl(\boldsymbol{y})$ satisfies $\|fl(\boldsymbol{y}) - \boldsymbol{y}\| \leq C \cdot (\log n) \cdot 2^{-p} \cdot \|\boldsymbol{y}\|$, where C is some absolute constant and $\boldsymbol{y}$ is the vector that would be obtained with exact computations. We refer to [12, Se. 24.1] for details. We now define a sample from $\overline{\rho'}_{\boldsymbol{\sigma}}$ as follows: compute a sample from $\rho'_{\boldsymbol{\sigma}}$ with absolute error $< 1/n^2$; if it is within distance $1/n^2$ of the middle of two consecutive integers, then restart; otherwise, round it to a closest integer and then reduce it modulo q. Finally, a distribution sampled from $\overline{\Upsilon}_\alpha$ for $\alpha \geq 0$ is defined as $\overline{\rho'}_{\boldsymbol{\sigma}}$, where $\sigma_i = \sqrt{\alpha^2 q^2 + x_i}$ with the x_i's sampled independently from the distribution $\mathrm{Exp}(n\alpha^2 q^2)$.

Sampling from $\rho'_{\boldsymbol{\sigma}}$ can be performed in time $\widetilde{O}(n)$. Sampling from $\overline{\Upsilon}_\alpha$ can also be performed in expected time $\widetilde{O}(n)$, and the running-time is bounded by a quantity that follows a geometric law of parameter < 1. Furthermore, in all our cryptographic applications, one could pre-compute such samples off-line (i.e., before the message M to be processed is known).

Lemma 6. *Assume that $\alpha q \geq \sqrt{n}$. For any $r \in R$, we have $\Pr_{y \hookleftarrow \overline{\Upsilon}_\alpha}[\|yr\|_\infty \geq \alpha q \omega(\log n) \cdot \|r\|] \leq n^{-\omega(1)}$.*

We now define our adaptation of R-LWE.

Definition 1. *The Ring Learning With Errors Problem with parameters q, α and Φ ($\text{R-LWE}^{\Phi}_{q,\alpha}$) is as follows. Let $\psi \hookleftarrow \overline{\Upsilon}_\alpha$ and $s \hookleftarrow U(R_q)$. Given access to an oracle $\mathcal{O}$ that produces samples in $R_q \times R_q$, distinguish whether $\mathcal{O}$ outputs samples from $A_{s,\psi}$ or from $U(R_q \times R_q)$. The distinguishing advantage should be $1/\mathcal{P}oly(n)$ (resp. $2^{-o(n)}$) over the randomness of the input, the randomness of the samples and the internal randomness of the algorithm.*

The following theorem indicates that R-LWE is hard, assuming that the worst-case γ-Ideal-SVP cannot be efficiently solved using quantum computers, for small γ. It was recently improved by Lyubashevsky *et al.* [21]: if the number of samples that can be asked to the oracle $\mathcal{O}$ is bounded by a constant (which is the case in our application), then the result also holds with simpler errors than $e \hookleftarrow \psi \hookleftarrow \overline{\Upsilon}_\alpha$, and with an even smaller Ideal-SVP approximation factor γ. This should allow to both simplify the modified `NTRUEncrypt` and to strengthen its security guarantee.

Theorem 1 (Adapted from [20]). *Assume that $\alpha q = \omega(n\sqrt{\log n})$ (resp. $\Omega(n^{1.5})$) with $\alpha \in (0,1)$ and $q = \mathcal{P}oly(n)$. There exists a randomized polynomial-time (resp. subexponential) quantum reduction from γ-Ideal-SVP to $\text{R-LWE}_{q,\alpha}$, with $\gamma = \omega(n^{1.5}\log n)/\alpha$ (resp. $\Omega(n^{2.5})/\alpha$).*

The differences with [20] in the above formulation are the use of the polynomial representation (handled by applying the complex FFT to the error term), the use

of R_q rather than $R_q^\vee := R^\vee/q$ where $R^\vee$ is the codifferent (here we have $R_q^\vee = \frac{1}{n}R_q$), and the truncation of the error to closest integer if it is far from the middle of two consecutive integers. The new variant remains hard because a sample passes the rejection step with non-negligible probability, and the rounding can be performed on the oracle samples obliviously to the actual error.

Variants of R-LWE. For $s \in R_q$ and ψ a distribution in R_q, we define $A^\times_{s,\psi}$ as the distribution obtained by sampling the pair $(a, as + e)$ with $(a, e) \hookleftarrow U(R_q^\times) \times \psi$, where $R_q^\times$ is the set of invertible elements of R_q. When $q = \Omega(n)$, the probability for a uniform element of R_q of being invertible is non-negligible, and thus R-LWE remains hard even when $A_{s,\psi}$ and $U(R_q \times R_q)$ are respectively replaced by $A^\times_{s,\psi}$ and $U(R_q^\times \times R_q)$. We call R-LWE$^\times$ the latter variant.

Furthermore, similarly to [3, Le. 2] and as explained in [21], the nonce s can also be chosen from the error distribution without incurring any security loss. We call R-LWE$^\times_{\text{HNF}}$ the corresponding modification of R-LWE. We recall the argument, for completeness. Assume an algorithm $\mathcal{A}$ can solve R-LWE$^\times_{\text{HNF}}$. We use $\mathcal{A}$ to solve R-LWE$^\times$. The principle is to transform samples $((a_i, b_i))_i$ into samples $((a_1^{-1}a_i, b_i - a_1^{-1}b_1a_i))_i$, where inversion is performed in $R_q^\times$. This transformation maps $A^\times_{s,\psi}$ to $A^\times_{-e_1,\psi}$, and $U(R_q^\times \times R_q)$ to itself.

3 New Results on Module q-Ary Lattices

In this section, we present strong regularity bounds for the ring R_q. For this purpose, we first study two families of R-modules.

3.1 Duality Results for Some Module Lattices

Let $\boldsymbol{a} \in R_q^m$. We define the following families of R-modules, for I an arbitrary ideal of R_q:

$$\boldsymbol{a}^\perp(I) := \{(t_1, \ldots, t_m) \in R^m : \forall i, (t_i \bmod q) \in I \text{ and } \sum_i t_i a_i = 0 \bmod q\},$$

$$L(\boldsymbol{a}, I) := \{(t_1, \ldots, t_m) \in R^m : \exists s \in R_q, \forall i, (t_i \bmod q) = a_i \cdot s \bmod I\}.$$

We also define $\boldsymbol{a}^\perp$ and $L(\boldsymbol{a})$ as $\boldsymbol{a}^\perp(R_q)$ and $L(\boldsymbol{a}, \langle 0\rangle)$ respectively. The ideals of R_q are of the form $I_S := \prod_{i\in S}(x - \phi_i) \cdot R_q = \{a \in R_q : \forall i \in S, a(\phi_i) = 0\}$, where S is any subset of $\{1, \ldots, n\}$ (the ϕ_i's are the roots of Φ modulo q). We define $I_S^\times = \prod_{i\in S}(x - \phi_i^{-1}) \cdot R_q$.

Lemma 7. *Let $S \subseteq \{1, \ldots, n\}$ and $\boldsymbol{a} \in R_q^m$. Let $\overline{S} = \{1, \ldots, n\} \setminus S$ and $\boldsymbol{a}^\times \in R_q^m$ be defined by $a_i^\times = a_i(x^{-1})$. Then (considering both sets as mn-dimensional lattices by identifying R and $\mathbb{Z}^n$):*

$$\widehat{\boldsymbol{a}^\perp(I_S)} = \frac{1}{q}L(\boldsymbol{a}^\times, I_{\overline{S}}^\times).$$

Proof. We first prove that $\frac{1}{q}L(\boldsymbol{a}^*, I_{\overline{S}}^{\times}) \subseteq \widehat{\boldsymbol{a}^{\perp}(I_S)}$. Let $(t_1, \ldots, t_m) \in \boldsymbol{a}^{\perp}(I_S)$ and $(t'_1, \ldots, t'_m) \in L(\boldsymbol{a}^*, I_{\overline{S}})$. Write $t_i = \sum_{j<n} t_{i,j} x^j$ and $t'_i = \sum_{j<n} t'_{i,j} x^j$ for any $i \leq m$. Our goal is to show that $\sum_{i\leq m, j\leq n} t_{i,j} t'_{i,j} = 0 \bmod q$. This is equivalent to showing that the constant coefficient of the polynomial $\sum_{i\leq m} t_i(x) t'_i(x^{-1})$ is 0 modulo q. It thus suffices to show that $\sum_{i\leq m} t_i(x) t'_i(x^{-1}) \bmod q = 0$ (in R_q). By definition of the t'_i's, there exists $s \in R_q$ such that $(t'_i \bmod q) = a_i^{\times} \cdot s + b'_i$ for some $b'_i \in I_{\overline{S}}^{\times}$. We have the following, modulo q:

$$\sum_{i\leq m} t_i(x) t'_i(x^{-1}) = s(x^{-1}) \cdot \sum_{i\leq m} t_i(x) a_i(x) + \sum_{i\leq m} t_i(x) b'_i(x^{-1}).$$

Both sums in the right hand side evaluate to 0 in R_q, which provides the desired inclusion.

To complete the proof, it suffices to show that $L(\widehat{\boldsymbol{a}^{\times}, I_{\overline{S}}^{\times}}) \subseteq \frac{1}{q}\boldsymbol{a}^{\perp}(I_S)$. It can be seen by considering the elements of $L(\boldsymbol{a}^{\times}, I_{\overline{S}})$ corresponding to $s = 1$. □

3.2 On the Absence of Unusually Short Vectors in $L(\boldsymbol{a}, I_S)$

We show that for $\boldsymbol{a} \hookleftarrow U((R_q^{\times})^m)$, the lattice $L(\boldsymbol{a}, I_S)$ is extremely unlikely to contain unusually short vectors for the infinity norm, i.e., much shorter than guaranteed by the Minkowski upper bound $\det(L(\boldsymbol{a}, I_S))^{\frac{1}{mn}} = q^{(1-\frac{1}{m})\frac{|S|}{n}}$ (we have $\det(L(\boldsymbol{a}, I_S)) = q^{(m-1)|S|}$ because there are $q^{n+(m-1)(n-|S|)}$ points of $L(\boldsymbol{a}, I_S)$ in the cube $[0, q-1]^{mn}$). Note that our lower bound approaches the Minkowski bound as $\frac{|S|}{n}$ approaches 1, but becomes progressively looser as $\frac{|S|}{n}$ drops towards $\approx 1 - \frac{1}{m}$. Fortunately, for our applications, we will be using this bound with $\frac{|S|}{n} = 1 - \varepsilon$ for some small ε, where the bound is close to being tight.

Lemma 8. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \geq 5$. Then, for any $S \subseteq \{1, \ldots, n\}$, $m \geq 2$ and $\varepsilon > 0$, we have $\lambda_1^{\infty}(L(\boldsymbol{a}, I_S)) \geq \frac{1}{\sqrt{n}} q^{\beta}$, with:*

$$\begin{aligned}\beta &:= 1 - \frac{1}{m} + \frac{1 - \sqrt{1 + 4m(m-1)\left(1 - \frac{|S|}{n}\right) + 4m\varepsilon}}{2m} \\ &\geq 1 - \frac{1}{m} - \varepsilon - (m-1)\left(1 - \frac{|S|}{n}\right),\end{aligned}$$

except with probability $\leq 2^n (q-1)^{-\varepsilon n}$ over the uniformly random choice of $\boldsymbol{a}$ in $(R_q^{\times})^m$.

Proof. Recall that $\Phi = \prod_{i\leq n} \Phi_i$ for distinct linear factors Φ_i. By the Chinese Remainder Theorem, we know that R_q (resp. $R_q^{\times}$) is isomorphic to $(\mathbb{Z}_q)^n$ (resp. $(\mathbb{Z}_q^{\times})^n$) via the isomorphism $t \mapsto (t \bmod \Phi_i)_{i\leq m}$. Let $g_{I_S} = \prod_{i\in S} \Phi_i$: it is a degree $|S|$ generator of I_S.

Let p denote the probability (over the randomness of $\boldsymbol{a}$) that $L(\boldsymbol{a}, I_S)$ contains a non-zero vector $\boldsymbol{t}$ of infinity norm $< B$, where $B = \frac{1}{\sqrt{n}} q^{\beta}$. We upper bound p by the union bound, summing the probabilities $p(\boldsymbol{t}, s) = \Pr_{\boldsymbol{a}}[\forall i, t_i = a_i s \bmod I_S]$ over all possible values for $\boldsymbol{t}$ of infinity norm $< B$ and $s \in R_q/I_S$. Since the a_i's are independent, we have $p(\boldsymbol{t}, s) = \prod_{i \le m} p_i(t_i, s)$, where $p_i(t_i, s) = \Pr_{a_i}[t_i = a_i s \bmod I_S]$.

Wlog we can assume that $\gcd(s, g_{I_S}) = \gcd(t_i, g_{I_S})$ (up to multiplication by an element of $\mathbb{Z}_q^{\times}$): If this is not the case, there exists $j \le n$ such that either $t_i \bmod \Phi_j = 0$ and $s \bmod \Phi_j \neq 0$, or $t_i \bmod \Phi_j \neq 0$ and $s \bmod \Phi_j = 0$; In both cases, we have $p_i(t_i, s) = 0$ because $a_i \in R_q^{\times}$. We now assume that $\gcd(s, g_{I_S}) = \gcd(t_i, g_{I_S}) = \prod_{i \in S'} \Phi_i$ for some $S' \subseteq S$ of size $0 \le d \le |S|$. For any $j \in S'$, we have $t_i = a_i s = 0 \bmod \Phi_j$ regardless of the value of $a_i \bmod \Phi_j$, while for $j \in S \setminus S'$, we have $s \neq 0 \bmod \Phi_j$ and there exists a unique value of $a_i \bmod \Phi_j$ such that $t_i = a_i s \bmod \Phi_j$. Moreover for any $j \notin S$, the value of $a_i \bmod \Phi_j$ can be arbitrary in $\mathbb{Z}_q^{\times}$. So, overall, there are $(q-1)^{d+n-|S|}$ differents a_i's in $R_q^{\times}$ such that $t_i = a_i s \bmod I_S$. This leads to $p_i(t_i, s) = (q-1)^{d-|S|}$.

So far, we have showed that the probability p can be upper bounded by:

$$p \;\le\; \sum_{0 \le d \le |S|} \; \sum_{\substack{h = \prod_{i \in S'} \Phi_i \\ S' \subseteq S \\ |S'| = d}} \; \sum_{\substack{s \in R_q/I_S \\ h | s}} \; \sum_{\substack{\boldsymbol{t} \in (R_q)^m \\ \forall i,\, 0 < \|t_i\|_\infty < B \\ \forall i,\, h | t_i}} \; \prod_{i \le m} (q-1)^{d-|S|}.$$

For $h = \prod_{i \in S'} \Phi_i$ of degree d, let $N(B, d)$ denote the number of $t \in R_q$ such that $\|t\|_\infty < B$ and $t = ht'$ for some $t' \in R_q$ of degree $< n - d$. We consider two bounds for $N(B, d)$ depending on d.

Suppose that $d \ge \beta \cdot n$. Then we claim that $N(B, d) = 0$. Indeed, any $t = ht'$ for some $t' \in R_q$ belongs to the ideal $\langle h, q \rangle$ of R generated by h and q. For any non-zero $t \in \langle h, q \rangle$, we have $\mathcal{N}(t) = \mathcal{N}(\langle t \rangle) \ge \mathcal{N}(\langle h, q \rangle) = q^d$, where the inequality is because the ideal $\langle t \rangle$ is a full-rank sub-ideal of $\langle h, q \rangle$, and the last equality is because $\deg h = d$. It follows from the arithmetic-geometric inequality that $\|t\| = \frac{1}{\sqrt{n}} T_2(t) \ge \mathcal{N}(t)^{1/n} \ge q^{d/n}$. By equivalence of norms, we conclude that $\|t\|_\infty \ge \lambda_1^\infty(\langle h, q \rangle) \ge \frac{1}{\sqrt{n}} q^{d/n}$. We see that $d/n \ge \beta$ implies that $\|t\|_\infty \ge B$, so that $N(B, d) = 0$.

Suppose now that $d < \beta \cdot n$. Then we claim that $N(B, d) \le (2B)^{n-d}$. Indeed, since the degree of h is d, the vector $\overline{t}$ formed by the $n - d$ low-order coefficients of t is related to the vector $\overline{t'}$ formed by the $n - d$ low-order coefficients of t' by a lower triangular $(n-d) \times (n-d)$ matrix whose diagonal coefficients are equal to 1. Hence this matrix is non-singular modulo q so the mapping from $\overline{t'}$ to $\overline{t}$ is one-to-one. This provides the claim.

Using the above bounds on $N(B, d)$, the fact that the number of subsets of S of cardinality d is $\le 2^d$, and the fact that the number of $s \in R_q/I_S$ divisible by $h = \prod_{i \in S'} \Phi_i$ is $q^{|S|-d}$, the above bound on p implies

$$p \;\le\; 2^n \max_{d \le \beta \cdot n} \frac{(2B)^{m(n-d)}}{(q-1)^{(m-1)(|S|-d)}}.$$

With our choice of B, we have $2B \leq (q-1)^\beta$ (this is implied by $n \geq 8, q \geq 5$ and $\beta \leq 1$). A straightforward computation then leads to the claimed upper bound on p. □

3.3 Improved Regularity Bounds

We now study the uniformity of distribution of $(m+1)$-tuples from $(R_q^\times)^m \times R_q$ of the form $(a_1, \ldots, a_m, \sum_{i \leq m} t_i a_i)$, where the a_i's are independent and uniformly random in $R_q^\times$, and the t_i's are chosen from some distribution on R_q concentrated on elements with small coefficients. Similarly to [22], we call the distance of the latter distribution to the uniform distribution on $(R_q^\times)^m \times R_q$ the *regularity* of the generalized knapsack function $(t_i)_{i \leq m} \mapsto \sum_{i \leq m} t_i a_i$. For our NTRU application we are particularly interested in the case where $m = 2$.

The regularity result in [22, Se. 4.1] applies when the a_i's are uniformly random in the whole ring R_q, and the t_i's are uniformly random on the subset of elements of R_q with coefficients of magnitude $\leq d$ for some $d < q$. In this case, the regularity bound from [22] is $\Omega(\sqrt{nq/d^m})$. Unfortunately, this bound is non-negligible for small m and q, e.g., for $m = O(1)$ and $q = \mathcal{P}oly(n)$. To make it exponentially small in n, one needs to set $m \log d = \Omega(n)$, which inevitably leads to inefficient cryptographic functions. When the a_i's are chosen uniformly from the whole ring R_q, the actual regularity is not much better than this undesirable regularity bound. This is because R_q contains n proper ideals of size $q^{n-1} = |R_q|/q$, and the probability $\approx n/q^m$ that all of the a_i's fall into one such ideal (which causes $\sum t_i a_i$ to also be trapped in the proper ideal) is non-negligible for small m. To circumvent this problem, we restrict the a_i's to be uniform in $R_q^\times$, and we choose the t_i's from a discrete Gaussian distribution. We show a regularity bound exponentially small in n even for $m = O(1)$, by using an argument similar to that used in [11, Se. 5.1] for unstructured generalized knapsacks, based on the *smoothing parameter* of the underlying lattices. Note that the new regularity result can be used within the Ideal-SIS trapdoor generation of [34, Se. 3], thus extending the latter to a fully splitting q. It also shows that the encryption scheme from [20] can be shown secure against subexponential (quantum) attackers, assuming the subexponential (quantum) hardness of standard worst-case problems over ideal lattices.

Theorem 2. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \geq 5$. Let $m \geq 2$, $\varepsilon > 0$, $\delta \in (0, 1/2)$ and $\boldsymbol{t} \hookleftarrow D_{\mathbb{Z}^{mn}, \sigma}$, with $\sigma \geq \sqrt{n \ln(2mn(1+1/\delta))/\pi} \cdot q^{\frac{1}{m}+\varepsilon}$. Then for all except a fraction $\leq 2^n(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^m$, we have $\eta_\delta(\boldsymbol{a}^\perp) \leq \sqrt{n \ln(2mn(1+1/\delta))/\pi} \cdot q^{\frac{1}{m}+\varepsilon}$, and the distance to uniformity of $\sum_{i \leq m} t_i a_i$ is $\leq 2\delta$. As a consequence:*

$$\Delta\left[\left(a_1, \ldots, a_m, \sum_{i \leq m} t_i a_i\right);\ U\left((R_q^\times)^m \times R_q\right)\right] \leq 2\delta + 2^n(q-1)^{-\varepsilon n}.$$

When using this result, one is typically interested in taking a small constant $\varepsilon > 0$, because it allows to lower the standard deviation σ and thus the required amount of randomness. Then a tiny δ should be chosen (e.g., $\delta \approx 2^n(q-1)^{-\varepsilon n}$), as it drastically lowers the statistical distance upper bound, without strengthening the standard deviation requirement much.

For each $\boldsymbol{a} \in (R_q^\times)^m$, let $D_{\boldsymbol{a}}$ denote the distribution of $\sum_{i \le m} t_i a_i$ where $\boldsymbol{t}$ is sampled from $D_{\mathbb{Z}^{mn},\sigma}$. Note that the above statistical distance is exactly $\frac{1}{|R_q^\times|^m} \sum_{\boldsymbol{a} \in (R_q^\times)^m} \Delta_{\boldsymbol{a}}$, where $\Delta_{\boldsymbol{a}}$ is the distance to uniformity of $D_{\boldsymbol{a}}$. To prove the theorem, it therefore suffices to show a distance bound $\Delta_{\boldsymbol{a}} \le 2\delta$, for all except a fraction $\le 2^n(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^m$.

Now, the mapping $\boldsymbol{t} \mapsto \sum_i t_i a_i$ induces an isomorphism from the quotient group $\mathbb{Z}^{mn}/\boldsymbol{a}^\perp$ to its range (note that $\boldsymbol{a}^\perp$ is the kernel of $\boldsymbol{t} \mapsto \sum_i t_i a_i$). The latter is R_q, thanks to the invertibility of the a_i's. Therefore, the statistical distance $\Delta_{\boldsymbol{a}}$ is equal to the distance to uniformity of $\boldsymbol{t} \bmod \boldsymbol{a}^\perp$. By Lemma 4, we have $\Delta_{\boldsymbol{a}} \le 2\delta$ if σ is greater than the smoothing parameter $\eta_\delta(\boldsymbol{a}^\perp)$ of $\boldsymbol{a}^\perp \subseteq \mathbb{Z}^{mn}$. To upper bound $\eta_\delta(\boldsymbol{a}^\perp)$, we apply Lemma 2, which reduces the task to lower bounding the minimum of the dual lattice $\widehat{\boldsymbol{a}^\perp} = \frac{1}{q} \cdot L(\boldsymbol{a}^\times)$, where $\boldsymbol{a}^\times \in (R_q^\times)^m$ is in one-to-one correspondence with $\boldsymbol{a}$.

The following result is a direct consequence of Lemmata 2, 4, 7 and 8. Theorem 2 follows by taking $S = \emptyset$ and $\boldsymbol{c} = \boldsymbol{0}$.

Lemma 9. *Let $n \ge 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \ge 5$. Let $S \subseteq \{1, \ldots, n\}$, $m \ge 2$, $\varepsilon > 0$, $\delta \in (0, 1/2)$, $\boldsymbol{c} \in \mathbb{R}^{mn}$ and $\boldsymbol{t} \hookleftarrow D_{\mathbb{Z}^{mn},\sigma,\boldsymbol{c}}$, with*

$$\sigma \ge \sqrt{n \ln(2mn(1 + 1/\delta))/\pi} \cdot q^{\frac{1}{m} + (m-1)\frac{|S|}{n} + \varepsilon}.$$

Then for all except a fraction $\le 2^n(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^m$, we have:

$$\Delta\Big[\boldsymbol{t} \bmod \boldsymbol{a}^\perp(I_S);\ U(R/\boldsymbol{a}^\perp(I_S))\Big] \le 2\delta.$$

4 A Revised Key Generation Algorithm

We now use the results of the previous section on modular q-ary lattices to derive a key generation algorithm for NTRUEncrypt, where the generated public key follows a distribution for which Ideal-SVP reduces to R-LWE.

4.1 NTRUEncrypt's Key Generation Algorithm

The new key generation algorithm for NTRUEncrypt is given in Fig. 1. The secret key polynomials f and g are generated by using the Gentry *et al.* sampler of discrete Gaussians (see Lemma 5), and by rejecting so that the output polynomials are invertible modulo q. The Gentry *et al.* sampler may not exactly sample from discrete Gaussians, but since the statistical distance can be made exponentially small, the impact on our results is also exponentially small. Furthermore, it can

be checked that our conditions on standard deviations are much stronger than the one in Lemma 5. From now on, we will assume we have a perfect discrete Gaussian sampler.

By choosing a large enough standard deviation σ, we can apply the results of the previous section and obtain the (quasi-)uniformity of the public key. We sample f of the form $p \cdot f' + 1$ so that it has inverse 1 modulo p, making the decryption process of `NTRUEncrypt` more efficient (as in the original `NTRUEncrypt` scheme). We remark that the rejection condition on f at Step 1 is equivalent to the condition $(f' \bmod q) \notin R_q^\times - p^{-1}$, where p^{-1} is the inverse of p in $R_q^\times$.

Inputs: $n, q \in \mathbb{Z}$, $p \in R_q^\times$, $\sigma \in \mathbb{R}$.
Output: A key pair $(sk, pk) \in R \times R_q^\times$.
1. Sample f' from $D_{\mathbb{Z}^n,\sigma}$; let $f = p \cdot f' + 1$; if $(f \bmod q) \notin R_q^\times$, resample.
2. Sample g from $D_{\mathbb{Z}^n,\sigma}$; if $(g \bmod q) \notin R_q^\times$, resample.
3. Return secret key $sk = f$ and public key $pk = h = pg/f \in R_q^\times$.

Fig. 1. Revised Key Generation Algorithm for `NTRUEncrypt`

The following result ensures that for some appropriate choice of parameters, the key generation algorithm terminates in expected polynomial time.

Lemma 10. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \geq 5$. Let $\sigma \geq \sqrt{n \ln(2n(1+1/\delta))/\pi} \cdot q^{1/n}$, for an arbitrary $\delta \in (0, 1/2)$. Let $a \in R$ and $p \in R_q^\times$. Then $\Pr_{f' \hookleftarrow D_{\mathbb{Z}^n,\sigma}}[(p \cdot f' + a \bmod q) \notin R_q^\times] \leq n(1/q + 2\delta)$.*

Proof. We are to bound the probability that $p \cdot f' + a$ belongs to $I := \langle q, \Phi_k \rangle$ by $1/q + 2\delta$, for any $k \leq n$. The result then follows from the Chinese Remainder Theorem and the union bound. We have $\mathcal{N}(I) = q$, so that $\lambda_1(I) \leq \sqrt{n} q^{1/n}$, by Minkowski's theorem. Since I is an ideal of R, we have $\lambda_n(I) = \lambda_1(I)$, and Lemma 1 gives that $\sigma \geq \eta_\delta(I)$. Lemma 4 then shows that $f \bmod I$ is within distance $\leq 2\delta$ to uniformity on R/I, so we have $p \cdot f' + a = 0 \bmod I$ (or, equivalently, $f' = -a/p \bmod I$) with probability $\leq 1/q + 2\delta$, as required. □

As a consequence of the above bound on the rejection probability, we have the following result, which ensures that the generated secret key is small.

Lemma 11. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \geq 8n$. Let $\sigma \geq \sqrt{2n \ln(6n)/\pi} \cdot q^{1/n}$. The secret key polynomials f, g returned by the algorithm of Fig. 1 satisfy, with probability $\geq 1 - 2^{-n+3}$:*

$$\|f\| \leq 2n\|p\|\sigma \quad \text{and} \quad \|g\| \leq \sqrt{n}\sigma.$$

If $\deg p \leq 1$, then $\|f\| \leq 4\sqrt{n}\|p\|\sigma$ with probability $\geq 1 - 2^{-n+3}$.

Proof. The probability under scope is lower than the probability of the same event without rejection, divided by the rejection probability. The result follows by combining Lemmata 3 and 10. □

4.2 Public Key Uniformity

In the algorithm of Fig. 1, the polynomials f' and g are independently sampled from the discrete Gaussian distribution $D_{\mathbb{Z}^n,\sigma}$ with $\sigma \geq \mathcal{P}oly(n) \cdot q^{1/2+\varepsilon}$ for an arbitrary $\varepsilon > 0$, but restricted (by rejection) to $R_q^\times - p^{-1}$ and $R_q^\times$, respectively. We denote by $D^\times_{\sigma,z}$ the discrete Gaussian $D_{\mathbb{Z}^n,\sigma}$ restricted to $R_q^\times + z$.

Here we apply the result of Section 3 to show that the statistical closeness to uniformity of a quotient of two distributions $(z + p \cdot D^\times_{\sigma,y})$ for $z \in R_q$ and $y = -zp^{-1} \bmod q$. This includes the case of the public key $h = pg/f \bmod q$ computed by the algorithm of Fig. 1.

Theorem 3. *Let $n \geq 8$ be a power of 2 such that $\Phi = x^n + 1$ splits into n linear factors modulo prime $q \geq 5$. Let $\varepsilon > 0$ and $\sigma \geq 2n\sqrt{\ln(8nq)} \cdot q^{\frac{1}{2}+2\varepsilon}$. Let $p \in R_q^\times$, $y_i \in R_q$ and $z_i = -y_i p^{-1} \bmod q$ for $i \in \{1, 2\}$. Then*

$$\Delta\left[\frac{y_1 + p \cdot D^\times_{\sigma,z_1}}{y_2 + p \cdot D^\times_{\sigma,z_2}} \bmod q \ ; \ U\left(R_q^\times\right)\right] \ \leq \ 2^{3n} q^{-\lfloor \varepsilon n \rfloor}.$$

Proof. For $a \in R_q^\times$, we define $Pr_a = \Pr_{f_1,f_2}[(y_1 + pf_1)/(y_2 + pf_2) = a]$, where $f_i \hookleftarrow D^\times_{\sigma,z_i}$ for $i \in \{1, 2\}$. We are to show that $|Pr_a - (q-1)^{-n}| \leq 2^{2n+5} q^{-\lfloor \varepsilon n \rfloor} \cdot (q-1)^{-n} =: \varepsilon'$ for all except a fraction $\leq 2^{2n}(q-1)^{-\varepsilon n}$ of $a \in R_q^\times$. This directly gives the claimed bound. The fraction of $a \in R_q^\times$ such that $|Pr_a - (q-1)^{-n}| \leq \varepsilon'$ is equal to the fraction of $\boldsymbol{a} = (a_1, a_2) \in (R_q^\times)^2$ such that $|Pr_{\boldsymbol{a}} - (q-1)^{-n}| \leq \varepsilon'$, where $Pr_{\boldsymbol{a}} = \Pr_{f_1,f_2}[a_1 f_1 + a_2 f_2 = a_1 z_1 + a_2 z_2]$. This is because $a_1 f_1 + a_2 f_2 = a_1 z_1 + a_2 z_2$ is equivalent to $(y_1 + pf_1)/(y_2 + pf_2) = -a_2/a_1$ (in $R_q^\times$), and $-a_2/a_1$ is uniformly random in $R_q^\times$ when $\boldsymbol{a} \hookleftarrow U((R_q^\times)^2)$.

We observe that $(f_1, f_2) = (z_1, z_2) =: \boldsymbol{z}$ satisfies $a_1 f_1 + a_2 f_2 = a_1 z_1 + a_2 z_2$, and hence the set of solutions $(f_1, f_2) \in R$ to the latter equation is $\boldsymbol{z} + \boldsymbol{a}^{\perp\times}$, where $\boldsymbol{a}^{\perp\times} = \boldsymbol{a}^\perp \cap (R_q^\times + q\mathbb{Z}^n)^2$. Therefore:

$$Pr_{\boldsymbol{a}} = \frac{D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z} + \boldsymbol{a}^{\perp\times})}{D_{\mathbb{Z}^n,\sigma}(z_1 + R_q^\times + q\mathbb{Z}^n) \cdot D_{\mathbb{Z}^n,\sigma}(z_2 + R_q^\times + q\mathbb{Z}^n)}.$$

We now use the fact that for any $\boldsymbol{t} \in \boldsymbol{a}^\perp$ we have $t_2 = -t_1 a_1/a_2$ so, since $-a_1/a_2 \in R_q^\times$, the ring elements t_1 and t_2 must belong to the *same* ideal I_S of R_q for some $S \subseteq \{1, \ldots, n\}$. It follows that $\boldsymbol{a}^{\perp\times} = \boldsymbol{a}^\perp \setminus \bigcup_{S \subseteq \{1,\ldots,n\}, S \neq \emptyset} \boldsymbol{a}^\perp(I_S)$. Similarly, we have $R_q^\times + q\mathbb{Z}^n = \mathbb{Z}^n \setminus \bigcup_{S \subseteq \{1,\ldots,n\}, S \neq \emptyset}(I_S + q\mathbb{Z}^n)$. Using the inclusion-exclusion principle, we obtain:

$$D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z} + \boldsymbol{a}^{\perp\times}) = \sum_{S \subseteq \{1,\ldots,n\}} (-1)^{|S|} \cdot D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z} + \boldsymbol{a}^\perp(I_S)), \quad (1)$$

$$\forall i \in \{1, 2\} : D_{\mathbb{Z}^n,\sigma}(z_i + R_q^\times + q\mathbb{Z}^n) = \sum_{S \subseteq \{1,\ldots,n\}} (-1)^{|S|} \cdot D_{\mathbb{Z}^n,\sigma}(z_i + I_S + q\mathbb{Z}^n). \quad (2)$$

In the rest of the proof, we show that, except for a fraction $\leq 2^{2n}(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^2$:

$$D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z} + \boldsymbol{a}^{\perp\times}) = (1 + \delta_0) \cdot \frac{(q-1)^n}{q^{2n}},$$

$$\forall i \in \{1,2\}: \; D_{\mathbb{Z}^n,\sigma}(z_i + R_q^\times + q\mathbb{Z}^n) = (1+\delta_i)\cdot\frac{(q-1)^n}{q^n}.$$

where $|\delta_i| \leq 2^{2n+2}q^{-\lfloor \varepsilon n\rfloor}$ for $i \in \{0,1,2\}$. The bound on $|Pr_{\boldsymbol{a}} - (q-1)^{-n}|$ follows by a routine computation.

Handling (1). We note that, since $\boldsymbol{z} \in \mathbb{Z}^{2n}$, we have (for any $S \subseteq \{1,\ldots,n\}$):

$$D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z}+\boldsymbol{a}^\perp(I_S)) = \frac{\rho_\sigma(\boldsymbol{z}+\boldsymbol{a}^\perp(I_S))}{\rho_\sigma(\mathbb{Z}^{2n})} = \frac{\rho_\sigma(\boldsymbol{z}+\boldsymbol{a}^\perp(I_S))}{\rho_\sigma(\boldsymbol{z}+\mathbb{Z}^{2n})} = D_{\mathbb{Z}^{2n},\sigma,-\boldsymbol{z}}(\boldsymbol{a}^\perp(I_S)).$$

For the terms of (1) with $|S| \leq \varepsilon n$, we apply Lemma 9 with $m = 2$. Since $|S|/n + \varepsilon \leq 2\varepsilon$, the Lemma 9 assumption on σ holds, with $\delta := q^{-n-\lfloor \varepsilon n\rfloor}$. We have $|R/\boldsymbol{a}^\perp(I_S)| = \det(\boldsymbol{a}^\perp(I_S)) = q^{n+|S|}$: Indeed, since $\boldsymbol{a} \in (R_q^\times)^2$, there are $q^{n-|S|}$ elements of $\boldsymbol{a}^\perp(I_S)$ in $[0, q-1]^{2n}$. We conclude that $|D_{\mathbb{Z}^{2n},\sigma,-\boldsymbol{z}}(\boldsymbol{a}^\perp(I_S)) - q^{-n-|S|}| \leq 2\delta$, for all except a fraction $\leq 2^n(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^2$ (possibly corresponding to a distinct subset of $(R_q^\times)^2$ for each possible S).

For a term of (1) with $|S| > \varepsilon n$, we choose $S' \subseteq S$ with $|S'| = \lfloor \varepsilon n\rfloor$. Then we have $\boldsymbol{a}^\perp(I_S) \subseteq \boldsymbol{a}^\perp(I_{S'})$ and hence $D_{\mathbb{Z}^{2n},\sigma,-\boldsymbol{z}}(\boldsymbol{a}^\perp(I_S)) \leq D_{\mathbb{Z}^{2n},\sigma,-\boldsymbol{z}}(\boldsymbol{a}^\perp(I_{S'}))$. By using with S' the above result for small $|S|$, we obtain $D_{\mathbb{Z}^{2n},\sigma,-\boldsymbol{z}}(\boldsymbol{a}^\perp(I_S)) \leq 2\delta + q^{-n-\lfloor \varepsilon n\rfloor}$.

Overall, we have, except possibly for a fraction $\leq 2^{2n}(q-1)^{-\varepsilon n}$ of $\boldsymbol{a} \in (R_q^\times)^2$:

$$\begin{aligned}\left|D_{\mathbb{Z}^{2n},\sigma}(\boldsymbol{z}+\boldsymbol{a}^{\perp\times}) - \sum_{k=0}^{n}(-1)^k\binom{n}{k}q^{-n-k}\right| &\leq 2^{n+1}\delta + 2\sum_{k=\lceil \varepsilon n\rceil}^{n}\binom{n}{k}q^{-n-\lfloor \varepsilon n\rfloor} \\ &\leq 2^{n+1}(\delta + q^{-n-\lfloor \varepsilon n\rfloor}).\end{aligned}$$

We conclude that $|\delta_0| \leq \frac{q^{2n}}{(q-1)^n}2^{n+1}(\delta + q^{-n-\lfloor \varepsilon n\rfloor}) \leq 2^{2n+1}(\delta q^n + q^{-\lfloor \varepsilon n\rfloor})$, as required.

Handling (2). For the bounds on δ_1 and δ_2, we use a similar argument. Let $i \in \{1,2\}$. The z_i term can be handled like like the $\boldsymbol{z}$ term of (1). We observe that for any $S \subseteq \{1,\ldots,n\}$, we have $\det(I_S + q\mathbb{Z}^n) = q^{|S|}$ and hence, by Minkowski's theorem, $\lambda_1(I_S + q\mathbb{Z}^n) \leq \sqrt{n}\cdot q^{|S|/n}$. Moreover, since $I_S + q\mathbb{Z}^n$ is an ideal lattice, we have $\lambda_n(I_S + q\mathbb{Z}^n) = \lambda_1(I_S + q\mathbb{Z}^n) \leq \sqrt{n}\cdot q^{|S|/n}$. Lemma 1 gives that $\sigma \geq \eta_\delta(I_S + q\mathbb{Z}^n)$ for any S such that $|S| \leq n/2$, with $\delta := q^{-n/2}$. Therefore, by Lemma 4, for such an S, we have $|D_{\mathbb{Z}^n,\sigma,-z_i}(I_S + q\mathbb{Z}^n) - q^{-|S|}| \leq 2\delta$.

For a term of (2) with $|S| > n/2$, we choose $S' \subseteq S$ with $|S'| = n/2$. By using with S' the above result for small $|S|$, we obtain $D_{\mathbb{Z}^n,\sigma,-z_i}(I_S + q\mathbb{Z}^n) \leq D_{\mathbb{Z}^n,\sigma,-z_i}(I_{S'} + q\mathbb{Z}^n) \leq 2\delta + q^{-n/2}$.

Overall, we have:

$$\begin{aligned}\left|D_{\mathbb{Z}^n,\sigma}(z_i + R_q^\times + q\mathbb{Z}^n) - \sum_{k=0}^{n}(-1)^k\binom{n}{k}q^{-k}\right| &\leq 2^{n+1}\delta + 2\sum_{k=n/2}^{n}\binom{n}{k}q^{-n/2} \\ &\leq 2^{n+1}(\delta + q^{-n/2}),\end{aligned}$$

which leads to the desired bound on δ_i (using $\varepsilon < 1/2$). This completes the proof of the theorem. □

5 NTRUEncrypt Revisited

Using our new results above, we describe a modification of NTRUEncrypt for which we can provide a security proof under a worst-case hardness assumption. We use $\Phi = x^n + 1$ with $n \geq 8$ a power of 2, $R = \mathbb{Z}[x]/\Phi$ and $R_q = R/qR$ with $q \geq 5$ prime such that $\Phi = \prod_{k=1}^{n} \Phi_k$ in R_q with distinct Φ_k's.

We define our modified NTRUEncrypt scheme with parameters n, q, p, α, σ as follows. The parameters n and q define the rings R and R_q. The parameter $p \in R_q^\times$ defines the plaintext message space as $\mathcal{P} = R/pR$. It must be a polynomial with 'small' coefficients with respect to q, but at the same time we require $\mathcal{N}(p) = |\mathcal{P}| = 2^{\Omega(n)}$ so that many bits can be encoded at once. Typical choices as used in the original NTRUEncrypt scheme are $p = 3$ and $p = x + 2$, but in our case, since q is prime, we may also choose $p = 2$. By reducing modulo the px^i's, we can write any element of p as $\sum_{0 \leq i < n} \varepsilon_i x^i p$, with $\varepsilon_i \in (-1/2, 1/2]$. Using the fact that $R = \mathbb{Z}[x]/(x^n + 1)$, we can thus assume that any element of $\mathcal{P}$ is an element of R with infinity norm $\leq (\deg(p) + 1) \cdot \|p\|$. The parameter α is the R-LWE noise distribution parameter. Finally, the parameter σ is the standard deviation of the discrete Gaussian distribution used in the key generation process (see Section 4).

- **Key generation.** Use the algorithm of Fig. 1 and return $sk = f \in R_q^\times$ with $f = 1 \bmod p$, and $pk = h = pg/f \in R_q^\times$.
- **Encryption.** Given message $M \in \mathcal{P}$, set $s, e \hookleftarrow \overline{\Upsilon}_\alpha$ and return ciphertext $C = hs + pe + M \in R_q$.
- **Decryption.** Given ciphertext C and secret key f, compute $C' = f \cdot C \in R_q$ and return $C' \bmod p$.

Fig. 2. The encryption scheme NTRUEncrypt$(n, q, p, \sigma, \alpha)$

The correctness conditions for the scheme are summarized below.

Lemma 12. *If* $\omega(n^{1.5} \log n)\alpha \deg(p)\|p\|^2\sigma < 1$ *(resp.* $\omega(n^{0.5} \log n)\alpha\|p\|^2\sigma < 1$ *if* $\deg p \leq 1$*) and* $\alpha q \geq n^{0.5}$*, then the decryption algorithm of* NTRUEncrypt *recovers* M *with probability* $1 - n^{-\omega(1)}$ *over the choice of* s, e, f, g*.*

Proof. In the decryption algorithm, we have $C' = p \cdot (gs + ef) + fM \bmod q$. Let $C'' = p \cdot (gs + ef) + fM$ computed in R (not modulo q). If $\|C''\|_\infty < q/2$ then we have $C' = C''$ in R and hence, since $f = 1 \bmod p$, $C' \bmod p = C'' \bmod p = M \bmod p$, i.e., the decryption algorithm succeeds. It thus suffices to give an upper bound on the probability that $\|C''\|_\infty > q/2$.

From Lemma 11, we know that with probability $\geq 1 - 2^{-n+3}$ both f and g have Euclidean norms $\leq 2n\|p\|\sigma$ (resp. $4\sqrt{n}\|p\|\sigma$ if $\deg p \leq 1$). This implies that $\|pf\|, \|pg\| \leq 2n^{1.5}\|p\|^2\sigma$ (resp. $8\sqrt{n}\|p\|^2\sigma$), with probability $\geq 1 - 2^{-n+3}$.

From Lemma 6, both pfe and pgs have infinity norms $\leq 2\alpha q n^{1.5}\omega(\log n)\cdot\|p\|^2\sigma$ (resp. $8\alpha q\sqrt{n}\omega(\log n)\cdot\|p\|^2\sigma$), with probability $1-n^{-\omega(1)}$. Independently:

$$\|fM\|_\infty \leq \|fM\| \leq \sqrt{n}\|f\|\|M\| \leq 2\cdot(\deg(p)+1)\cdot n^2\|p\|^2\sigma \quad (\text{resp. } 8n\|p\|^2\sigma).$$

Since $\alpha q \geq \sqrt{n}$, we conclude that $\|C''\|_\infty \leq (6+2\deg(p))\cdot\alpha q n^{1.5}\omega(\log n)\cdot\|p\|^2\sigma$ (resp. $24\alpha q n^{0.5}\omega(\log n)\cdot\|p\|^2\sigma$), with probability $1-n^{-\omega(1)}$. □

The security of the scheme follows by an elementary reduction from the decisional R-LWE$^\times_{\mathrm{HNF}}$, exploiting the uniformity of the public key in $R_q^\times$ (Theorem 3), and the invertibility of p in R_q.

Lemma 13. *Suppose n is a power of 2 such that $\Phi = x^n+1$ splits into n linear factors modulo prime $q=\omega(1)$. Let $\varepsilon,\delta>0$, $p\in R_q^\times$ and $\sigma \geq 2n\sqrt{\ln(8nq)}\cdot q^{\frac{1}{2}+\varepsilon}$. If there exists an IND-CPA attack against* `NTRUEncrypt` *that runs in time T and has success probability $1/2+\delta$, then there exists an algorithm solving* R-LWE$^\times_{\mathrm{HNF}}$ *with parameters q and α that runs in time $T'=T+O(n)$ and has success probability $\delta'=\delta-q^{-\Omega(n)}$.*

Proof. Let $\mathcal{A}$ denote the given IND-CPA attack algorithm. We construct an algorithm $\mathcal{B}$ against R-LWE$^\times_{\mathrm{HNF}}$ that runs as follows, given oracle $\mathcal{O}$ that samples from either $U(R_q^\times\times R_q)$ or $A^\times_{s,\psi}$ for some previously chosen $s\hookleftarrow\psi$ and $\psi\hookleftarrow\overline{\Upsilon}_\alpha$. Algorithm $\mathcal{B}$ first calls $\mathcal{O}$ to get a sample (h',C') from $R_q^\times\times R_q$. Then, algorithm $\mathcal{B}$ runs $\mathcal{A}$ with public key $h=p\cdot h'\in R_q$. When $\mathcal{A}$ outputs challenge messages $M_0,M_1\in\mathcal{P}$, algorithm $\mathcal{B}$ picks $b\hookleftarrow U(\{0,1\})$, computes the challenge ciphertext $C=p\cdot C'+M_b\in R_q$, and returns C to $\mathcal{A}$. Eventually, when $\mathcal{A}$ outputs its guess b' for b, algorithm $\mathcal{B}$ outputs 1 if $b'=b$ and 0 otherwise.

The h' used by $\mathcal{B}$ is uniformly random in $R_q^\times$, and therefore so is the public key h given to $\mathcal{A}$, thanks to the invertibility of p modulo q. Thus, by Theorem 3, the public key given to $\mathcal{A}$ is within statistical distance $q^{-\Omega(n)}$ of the public key distribution in the genuine attack. Moreover, since $C'=h\cdot s+e$ with $s,e\hookleftarrow\psi$, the ciphertext C given to $\mathcal{A}$ has the right distribution as in the IND-CPA attack. Overall, if $\mathcal{O}$ outputs samples from $A^\times_{s,\psi}$, then $\mathcal{A}$ succeeds and $\mathcal{B}$ returns 1 with probability $\geq 1/2+\delta-q^{-\Omega(n)}$.

Now, if $\mathcal{O}$ outputs samples from $U(R_q^\times\times R_q)$, then, since $p\in R_q^\times$, the value of $p\cdot C'$ and hence C, is uniformly random in R_q and independent of b. It follows that $\mathcal{B}$ outputs 1 with probability $1/2$. The claimed advantage of $\mathcal{B}$ follows. □

By combining Lemmata 12 and 13 with Theorem 1 we obtain our main result.

Theorem 4. *Suppose n is a power of 2 such that $\Phi=x^n+1$ splits into n linear factors modulo prime $q=\mathcal{P}oly(n)$ such that $q^{\frac{1}{2}-\varepsilon}=\omega(n^{3.5}\log^2 n\deg(p)\|p\|^2)$ (resp. $q^{\frac{1}{2}-\varepsilon}=\omega(n^4\log^{1.5}n\deg(p)\|p\|^2)$), for arbitrary $\varepsilon\in(0,1/2)$ and $p\in R_q^\times$. Let $\sigma=2n\sqrt{\ln(8nq)}\cdot q^{\frac{1}{2}+\varepsilon}$ and $\alpha^{-1}=\omega(n^{1.5}\log n\deg(p)\|p\|^2\sigma)$. If there exists an IND-CPA attack against* `NTRUEncrypt`(n,q,p,σ,α) *which runs in time $T=\mathcal{P}oly(n)$ and has success probability $1/2+1/\mathcal{P}oly(n)$ (resp. time $T=2^{o(n)}$ and*

success probability $1/2 + 2^{-o(n)}$*), then there exists a* $\mathcal{P}oly(n)$*-time (resp.* $2^{o(n)}$*-time) quantum algorithm for* γ-Ideal-SVP *with* $\gamma = O(n^4 \log^{2.5} n \deg(p)\|p\|^2 q^{\frac{1}{2}+\varepsilon})$ *(resp.* $\gamma = O(n^5 \log^{1.5} n \deg(p)\|p\|^2 q^{\frac{1}{2}+\varepsilon})$*). Moreover, the decryption algorithm succeeds with probability* $1-n^{-\omega(1)}$ *over the choice of the encryption randomness.*

In the case where $\deg p \leq 1$, the conditions on q for polynomial-time (resp. subexponential) attacks in Theorem 4 may be relaxed to $q^{\frac{1}{2}-\varepsilon} = \omega(n^{2.5} \log^2 n \cdot \|p\|^2)$ (resp. $q^{\frac{1}{2}-\varepsilon} = \omega(n^3 \log^{1.5} n \cdot \|p\|^2)$) and the resulting Ideal-SVP approximation factor may be improved to $\gamma = O(n^3 \log^{2.5} n \cdot \|p\|^2 q^{\frac{1}{2}+\varepsilon})$ (resp. $\gamma = O(n^4 \log^{1.5} n \cdot \|p\|^2 q^{\frac{1}{2}+\varepsilon})$). Overall, by choosing $\varepsilon = o(1)$, the smallest q for which the analysis holds is $\widetilde{\Omega}(n^5)$ (resp. $\widetilde{\Omega}(n^6)$), and the smallest γ that can be obtained is $\widetilde{O}(n^{5.5})$ (resp. $\widetilde{O}(n^7)$).

Acknowledgements. We thank G. Hanrot, V. Lyubashevsky, Khoa T. T. Nguyen, C. Peikert, O. Regev, I. Shparlinski, J. Silverman and F. Vercauteren for helpful discussions. The authors were partly supported by the LaRedA ANR grant, an Australian Research Fellowship (ARF) from the Australian Research Council under Discovery Grant DP0987734, a Macquarie University Research Fellowship, and ARC Discovery Grant DP110100628.

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
2. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the 28th Symposium on the Theory of Computing (STOC 1996), pp. 99–108. ACM, New York (1996)
3. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
4. Banks, W.D., Shparlinski, I.E.: Distribution of inverses in polynomial rings. Indagationes Mathematicae 12(3), 303–315 (2001)
5. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
6. Coppersmith, D., Shamir, A.: Lattice attacks on NTRU. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 52–61. Springer, Heidelberg (1997)
7. von zur Gathen, J., Gerhardt, J.: Modern Computer Algebra, 2nd edn. Cambridge University Press, Cambridge (2003)
8. Gentry, C.: A fully homomorphic encryption scheme. PhD thesis, Stanford University (2009) (manuscript), http://crypto.stanford.edu/craig
9. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. of STOC, pp. 169–178. ACM, New York (2009)
10. Gentry, C.: Toward basing fully homomorphic encryption on worst-case hardness. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 116–137. Springer, Heidelberg (2010)

11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proc. of STOC, pp. 197–206. ACM, New York (2008)
12. Higham, N.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia (2002)
13. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Whyte, W.: Practical lattice-based cryptography: NTRUEncrypt and NTRUSign. In: Chapter of [26] (2009)
14. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: a new high speed public key cryptosystem. Preprint; presented at the rump session of Crypto 1996 (1996)
15. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998)
16. Hoffstein, J., Pipher, J., Silverman, J.H.: NSS: An NTRU lattice-based signature scheme. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 211. Springer, Heidelberg (2001)
17. IEEE P1363. Standard specifications for public-key cryptography, http://grouper.ieee.org/groups/1363/
18. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006)
19. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A modest proposal for FFT hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
20. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
21. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings, Draft for the extended version of [20], dated 01/02/2011
22. Micciancio, D.: Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. Comput. Complexity 16(4), 365–411 (2007)
23. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM J. Comput. 37(1), 267–302 (2007)
24. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009)
25. Micciancio, D., Voulgaris, P.: A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In: Proc. of STOC, pp. 351–358. ACM, New York (2010)
26. Nguyen, P.Q., Vallée, B. (eds.): The LLL Algorithm: Survey and Applications. Information Security and Cryptography. Springer, Heidelberg (2009)
27. Peikert, C.: Limits on the hardness of lattice problems in ℓ_p norms. Comput. Complexity 2(17), 300–351 (2008)
28. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proc. of STOC, pp. 333–342. ACM, New York (2009)
29. Peikert, C., Rosen, A.: Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 145–166. Springer, Heidelberg (2006)
30. Perlner, R.A., Cooper, D.A.: Quantum resistant public key cryptography: a survey. In: Proc. of IDtrust, pp. 85–93. ACM, New York (2009)
31. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56(6) (2009)

32. Regev, O.: The learning with errors problem. Invited survey in CCC 2010 (2010), http://www.cs.tau.ac.il/~odedr/
33. Schnorr, C.P.: A hierarchy of polynomial lattice basis reduction algorithms. Theor. Comput. Science 53, 201–224 (1987)
34. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009)
35. Xylouris, T.: On Linnik's constant (2009) (in German), http://arxiv.org/abs/0906.2749

Faster Explicit Formulas for Computing Pairings over Ordinary Curves

Diego F. Aranha[1,⋆], Koray Karabina[2,⋆], Patrick Longa[3], Catherine H. Gebotys[3], and Julio López[1]

[1] University of Campinas
{dfaranha,jlopez}@ic.unicamp.br
[2] Certicom Research
kkarabina@rim.com
[3] University of Waterloo
{plonga,cgebotys}@uwaterloo.ca

Abstract. We describe efficient formulas for computing pairings on ordinary elliptic curves over prime fields. First, we generalize lazy reduction techniques, previously considered only for arithmetic in quadratic extensions, to the whole pairing computation, including towering and curve arithmetic. Second, we introduce a new compressed squaring formula for cyclotomic subgroups and a new technique to avoid performing an inversion in the final exponentiation when the curve is parameterized by a negative integer. The techniques are illustrated in the context of pairing computation over Barreto-Naehrig curves, where they have a particularly efficient realization, and are also combined with other important developments in the recent literature. The resulting formulas reduce the number of required operations and, consequently, execution time, improving on the state-of-the-art performance of cryptographic pairings by 28%-34% on several popular 64-bit computing platforms. In particular, our techniques allow to compute a pairing under 2 million cycles for the first time on such architectures.

Keywords: Efficient software implementation, explicit formulas, bilinear pairings.

1 Introduction

The performance of pairing computation has received increasing interest in the research community, mainly because Pairing-Based Cryptography enables efficient and elegant solutions to several longstanding problems in cryptography such as Identity-Based Encryption [1,2], powerful non-interactive zero-knowledge proof systems [3] and communication-efficient multi-party key agreements [4]. Recently, dramatic improvements over the figure of 10 million cycles presented in [5] made possible to compute a pairing at the 128-bit security level in 4.38 million cycles [6] when using high-speed vector floating-point operations, and

⋆ This work was completed while these authors were at the University of Waterloo.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 48–68, 2011.

2.33 million cycles [7] when the fastest integer multiplier available in Intel 64-bit architectures is employed.

This work revisits the problem of efficiently computing pairings over large-characteristic fields and improves the state-of-the-art performance of cryptographic pairings by a significant margin. First of all, it builds on the latest advancements proposed by several authors:

- The Optimal Ate pairing [8] computed entirely on twists [9] with simplified final line evaluations [6] over a recently-introduced subclass [10] of the Barreto-Naehrig (BN) family of pairing-friendly elliptic curves [11].
- The implementation techniques described by [7] for accelerating quadratic extension field arithmetic, showing how to reduce expensive carry handling and function call overheads.

On the other hand, the following new techniques are introduced:

- The notion of lazy reduction, usually applied for arithmetic in quadratic extensions in the context of pairings, as discussed in [12], is generalized to the towering and curve arithmetic performed in the pairing computation. In a sense, this follows a direction opposite to the one taken by other authors. Instead of trying to encode arithmetic so that modular reductions are faster [13,6], we insist on Montgomery reduction and focus our efforts on reducing *the need* of computing reductions. Moreover, for dealing with costly higher-precision additions inserted by lazy reduction, we develop a flexible methodology that keeps intermediate values under Montgomery reduction boundaries and maximizes the use of operations without carry checks. The traditional operation count model is also augmented to take into account modular reductions individually.
- Formulas for point doubling and point addition in Jacobian and homogeneous coordinates are carefully optimized by eliminating several commonly neglected operations that are not inexpensive on modern 64-bit platforms.
- The computation of the final exponentiation is improved with a new set of formulas for compressed squaring and efficient decompression in cyclotomic subgroups, and an arithmetic trick to remove a significant penalty incurred when computing pairings over curves parameterized by negative integers.

The described techniques produce significant savings, allowing our illustrative software implementation to compute a pairing under 2 million cycles and improve the state-of-the-art timings by 28%-34% on several different 64-bit computing platforms. Even though the techniques are applied on pairings over BN curves at the 128-bit security level, they can be easily extended to other settings using different curves and higher security levels [14].

This paper is organized as follows. Section 2 gives an overview of Miller's Algorithm when employed for computing the Optimal Ate pairing over Barreto-Naehrig curves. Section 3 presents the generalized lazy reduction technique and its application to the improvement of towering arithmetic performance. Different optimizations to curve arithmetic, including the application of lazy reduction,

are discussed in Section 4. Section 5 describes our improvements on the final exponentiation. Section 6 summarizes operation counts and Section 7 describes our high-speed software implementation and comparison of results with the previously fastest implementation in the literature. Section 8 concludes the paper.

2 Preliminaries

An *admissible bilinear pairing* is a non-degenerate efficiently-computable map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are additive groups of points in an elliptic curve E and $\mathbb{G}_T$ is a subgroup of the multiplicative group of a finite field. The core property of map e is linearity in both arguments, allowing the construction of novel cryptographic schemes with security relying on the hardness of the Discrete Logarithm Problem in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$.

Barreto and Naehrig [11] described a parameterized family of elliptic curves $E_b : y^2 = x^3 + b, b \neq 0$ over a prime field $\mathbb{F}_p$, $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$, with prime order $n = 36u^4 + 36u^3 + 18u^2 + 6u + 1$, where $u \in \mathbb{Z}$ is an arbitrary integer. This family is rather large and easy to generate [10], providing a multitude of parameter choices; and, having embedding degree $k = 12$, is well-suited for computing asymmetric pairings at the 128-bit security level [12]. It admits several optimal derivations [8] of different variants of the Ate pairing [15] such as R-ate [16], Optimal Ate [8] and χ-ate [17].

Let $E[n]$ be the subgroup of n-torsion points of E and $E' : y^2 = x^3 + b/\xi$ be a sextic twist of E with ξ not a cube nor a square in $\mathbb{F}_{p^2}$. For the clear benefit of direct benchmarking, but also pointing that performance among variants is roughly the same, we restrict the discussion to computing the Optimal Ate pairing defined as in [6]:

$$a_{opt} : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T$$
$$(Q, P) \to (f_{r,Q}(P) \cdot l_{[r]Q,\pi_p(Q)}(P) \cdot l_{[r]Q+\pi_p(Q),-\pi_p^2(Q)}(P))^{\frac{p^{12}-1}{n}},$$

where $r = 6u + 2 \in \mathbb{Z}$; the map $\pi_p : E \to E$ is the Frobenius endomorphism $\pi_p(x, y) = (x^p, y^p)$; groups $\mathbb{G}_1, \mathbb{G}_2$ are determined by the eigenspaces of π_p as $\mathbb{G}_1 = E[n] \cap \mathrm{Ker}(\pi_p - [1]) = E(\mathbb{F}_p)[n]$ and $\mathbb{G}_2$ as the preimage $E'(\mathbb{F}_{p^2})[n]$ of $E[n] \cap \mathrm{Ker}(\pi_p - [p]) \subseteq E(\mathbb{F}_{p^{12}})[n]$ under the twisting isomorphism $\psi : E' \to E$; the group $\mathbb{G}_T$ is the subgroup of n-th roots of unity $\mu_n \subset \mathbb{F}^*_{p^{12}}$; $f_{r,Q}(P)$ is a normalized function with divisor $(f_{r,Q}) = r(Q) - ([r]Q) - (r - 1)(\mathcal{O})$ and $l_{Q_1,Q_2}(P)$ is the line arising in the addition of Q_1 and Q_2 evaluated at point P.

Miller [18,19] proposed an algorithm that constructs $f_{r,P}$ in stages by using a double-and-add method. When generalizing the denominator-free version [20] of Miller's Algorithm for computing the pairing a_{opt} with the set of implementation-friendly parameters suggested by [10] at the 128-bit security level, we obtain Algorithm 1. For the BN curve we have $E : y^2 = x^3 + 2, u = -(2^{62} + 2^{55} + 1) < 0$. In order to accommodate the negative r (line 9 in Algorithm 1), it is required to compute a cheap negation in $\mathbb{G}_2$ to make the final accumulator T the result of $[-|r|]Q$, and an expensive inversion in the big field $\mathbb{G}_T$ to obtain the correct

pairing value $f_{-|r|,Q}(P) = (f_{|r|,Q}(P))^{-1}$, instead of the value $f_{|r|,Q}(P)$ produced at the end of the algorithm. The expensive inversion will be handled later at Section 5 with the help of the final exponentiation.

Algorithm 1. Optimal Ate pairing on BN curves (generalized for $u < 0$)

Input: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, r = |6u+2| = \sum_{i=0}^{\log_2(r)} r_i 2^i$
Output: $a_{opt}(Q,P)$

1. $T \leftarrow Q$, $f \leftarrow 1$
2. **for** $i = \lfloor \log_2(r) \rfloor - 1$ **downto** 0 **do**
3. $\quad f \leftarrow f^2 \cdot l_{T,T}(P), T \leftarrow 2T$
4. $\quad$ **if** $r_i = 1$ **then**
5. $\quad\quad f \leftarrow f \cdot l_{T,Q}(P), T \leftarrow T + Q$
6. **end for**
7. $Q_1 \leftarrow \pi_p(Q), Q_2 \leftarrow \pi_p^2(Q)$
8. **if** $u < 0$ **then**
9. $\quad T \leftarrow -T, f \leftarrow f^{-1}$
10. **end if**
11. $f \leftarrow f \cdot l_{T,Q_1}(P), T \leftarrow T + Q_1$
12. $f \leftarrow f \cdot l_{T,-Q_2}(P), T \leftarrow T - Q_2$
13. $f \leftarrow f^{(p^{12}-1)/n}$
14. **return** f

3 Tower Extension Field Arithmetic

Miller's Algorithm [18,19] employs arithmetic in $\mathbb{F}_{p^{12}}$ during the accumulation steps (lines 3,5,11-12 in Algorithm 1) and at the final exponentiation (line 13 in the same algorithm). Hence, to achieve a high-performance implementation of pairings it is crucial to perform arithmetic over extension fields efficiently. In particular, it has been recommended in [21] to represent $\mathbb{F}_{p^k}$ with a tower of extensions using irreducible binomials. Accordingly, in our targeted setting we represent $\mathbb{F}_{p^{12}}$ using the flexible towering scheme used in [22,5,7,10] combined with the parameters suggested by [10]:

- $\mathbb{F}_{p^2} = \mathbb{F}_p[i]/(i^2 - \beta)$, where $\beta = -1$.
- $\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[s]/(s^2 - \xi)$, where $\xi = 1 + i$.
- $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$, where $\xi = 1 + i$.
- $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[t]/(t^3 - s)$ or $\mathbb{F}_{p^6}[w]/(w^2 - v)$.

It is possible to convert from one towering $\mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^6} \rightarrow \mathbb{F}_{p^{12}}$ to the other $\mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^4} \rightarrow \mathbb{F}_{p^{12}}$ by simply permuting the order of coefficients. The choice $p \equiv 3 \pmod 4$ accelerates arithmetic in $\mathbb{F}_{p^2}$, since multiplications by $\beta = -1$ can be computed as simple subtractions [10].

3.1 Lazy Reduction for Tower Fields

The concept of lazy reduction goes back to at least [23] and has been advantageously exploited by many works in different scenarios [24,25,12]. Lim and Hwang [24] showed that multiplication in $\mathbb{F}_{p^k}$, when $\mathbb{F}_{p^k} = \mathbb{F}_p[x]/(x^k - w)$ is seen as a direct extension over $\mathbb{F}_p$ via the irreducible binomial $(x^k - w)$ with $w \in \mathbb{F}_p$, can be performed with k reductions modulo p. In contrast, it would normally require either k^2 reductions using conventional multiplication, or $k(k+1)/2$ reductions using Karatsuba multiplication. Lazy reduction was first employed in the context of pairing computation by [12] to eliminate reductions in $\mathbb{F}_{p^2}$ multiplication. If one considers the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$, then this approach requires $2 \cdot 6 \cdot 3 = 36$ reductions modulo p, and $3 \cdot 6 \cdot 3 = 54$ integer multiplications for performing one multiplication in $\mathbb{F}_{p^{12}}$; see [12,5,7].

In this section, we generalize the lazy reduction technique to towering-friendly fields $\mathbb{F}_{p^k}$, $k = 2^i 3^j$, $i \geq 1, j \geq 0$, conveniently built with irreducible binomials [26]. We show that multiplication (and squaring) in a tower extension $\mathbb{F}_{p^k}$ only requires k reductions and still benefits from different arithmetic optimizations available in the literature to reduce the number of subfield multiplications or squarings. For instance, with our approach one now requires $2 \cdot 3 \cdot 2 = 12$ reductions modulo p and 54 integer multiplications using the tower $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$ to compute one multiplication in $\mathbb{F}_{p^{12}}$; or 12 reductions modulo p and 36 integer multiplications to compute one squaring in $\mathbb{F}_{p^{12}}$. Although wider in generality, these techniques are analyzed in detail in the context of Montgomery multiplication and Montgomery reduction [27], which are commonly used in the context of pairings over ordinary curves. We explicitly state our formulas for the towering construction $\mathbb{F}_p \to \mathbb{F}_{p^2} \to \mathbb{F}_{p^6} \to \mathbb{F}_{p^{12}}$ in Section 3.3. To remove ambiguity, the term *reduction modulo p* always refers to modular reduction of double-precision integers.

Theorem 1. *Let $k = 2^i 3^j$, $i, j \in \mathbb{Z}$ and $i \geq 1, j \geq 0$. Let*

$$\mathbb{F}_p = \mathbb{F}_{p^{k_0}} \to \mathbb{F}_{p^{k_1}} = \mathbb{F}_{p^2} \to \cdots \to \mathbb{F}_{p^{k_{i+j-2}}} \to \mathbb{F}_{p^{k_{i+j-1}}} \to \mathbb{F}_{p^{k_{i+j}}} = \mathbb{F}_{p^k}$$

be a tower extension, where each extension $\mathbb{F}_{p^{k_{\ell+1}}}/\mathbb{F}_{p^{k_\ell}}$ is of degree either 2 or 3, which can be constructed using a second degree irreducible binomial $x^2 - \beta_\ell$, $\beta_\ell \in \mathbb{F}_{p^{k_\ell}}$, or a third degree irreducible binomial $x^3 - \beta_\ell$, $\beta_\ell \in \mathbb{F}_{p^{k_\ell}}$, respectively. Suppose that β_ℓ can be chosen such that, for all $a \in \mathbb{F}_{p^{k_\ell}}$, $a \cdot \beta_\ell$ can be computed without any reduction modulo p. Then multiplication in $\mathbb{F}_{p^k}$ can be computed with $3^i 6^j$ integer multiplications and $k = 2^i 3^j$ reductions modulo p for any k.

Proof. We prove this by induction on $i + j$. The base case is $i + j = 1$ ($i = 1$ and $j = 0$). That is, $k = 2$, and we have a tower $\mathbb{F}_p \to \mathbb{F}_{p^2}$ with $\mathbb{F}_{p^2} = \mathbb{F}_p[x]/(x^2 - \beta)$. For any $a = a_0 + a_1 x, b = b_0 + b_1 x \in \mathbb{F}_{p^2}$, $a_i, b_i \in \mathbb{F}_p$, we can write

$$a \cdot b = (a_0 b_0 + a_1 b_1 \beta) + ((a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1)x,$$

which can be computed with 3 integer multiplications and 2 reductions modulo p (note that we ignore multiplication by β, by our assumption).

Next, consider

$$\mathbb{F}_p \rightarrow \mathbb{F}_{p^2} \rightarrow \cdots \rightarrow \mathbb{F}_{p^{k_{i+j}}} \rightarrow \mathbb{F}_{p^{k_{i+j+1}}},$$

where $k_{i+j+1} = 2^{i+1}3^j$, or $k_{i+j+1} = 2^i 3^{j+1}$. In the former case, let $\mathbb{F}_{p^{k_{i+j+1}}} = \mathbb{F}_{p^{k_{i+j}}}[x]/(x^2-\beta)$ and $a = a_0+a_1x$, $b = b_0+b_1x \in \mathbb{F}_{p^{k_{i+j+1}}}$, $a_i, b_i \in \mathbb{F}_{p^{k_{i+j}}}$. Then

$$a \cdot b = (a_0b_0 + a_1b_1\beta) + [(a_0 + a_1)(b_0 + b_1) - a_0b_0 - a_1b_1]\, x, \tag{1}$$

which can be computed with 3 multiplications in $\mathbb{F}_{p^{k_{i+j}}}$, namely a_0b_0, $a_1b_1\beta$ and $(a_0 + a_1)(b_0 + b_1)$ (again, we ignore multiplication by β). By the induction hypothesis, each multiplication in $\mathbb{F}_{p^{k_{i+j}}}$ requires 3^i6^j integer multiplications, and 2^i3^j reductions modulo p. Also, three reductions modulo p, when computing a_0b_0, $a_1b_1\beta$ and $(a_0+a_1)(b_0+b_1)$, can be minimized to two reductions modulo p (see (1)). Hence, multiplication in $\mathbb{F}_{p^{k_{i+j+1}}}$ can be computed with $3 \cdot 3^i6^j = 3^{i+1}6^j$ integer multiplications and $2 \cdot 2^i3^j = 2^{i+1}3^j$ reductions modulo p.

The latter case, $k_{i+j+1} = 2^i3^{j+1}$, can be proved similarly, by considering $\mathbb{F}_{p^{k_{i+j+1}}} = \mathbb{F}_{p^{k_{i+j}}}[x]/(x^3 - \beta)$, and the Karatsuba multiplication formula for degree 3 extensions instead of (1). □

It is also straightforward to generalize the procedure above to any formula other than Karatsuba which also involves only sums (or subtractions) of products of the form $\sum \pm a_ib_j$, with $a_i, b_j \in \mathbb{F}_{p^{k_l}}$, such as complex squaring or the Chung-Hasan asymmetric squaring formulas [28].

For efficiency purposes, we suggest a different treatment for the highest layer in the tower arithmetic. Theorem 1 implies that reductions can be completely delayed to the end of the last layer by applying lazy reduction, but in some cases (when the optimal k is already reached and no reductions can be saved) it will be more efficient to perform reductions immediately after multiplications or squarings. This will be illustrated with the computation of squaring in $\mathbb{F}_{p^{12}}$ in Section 3.3.

In the Miller Loop, reductions can also be delayed from the underlying $\mathbb{F}_{p^2}$ field during multiplication and squaring to the arithmetic layer immediately above (i.e., the point arithmetic and line evaluation). Similarly to the tower extension, on this upper layer reductions should only be delayed in the cases where this technique leads to fewer reductions. For details, see Section 4.

There are some penalties when delaying reductions. In particular, *single-precision* operations (with operands occupying $n = \lceil \lceil \log_2 p \rceil / w \rceil$ words, where w is the computer word-size) are replaced by *double-precision* operations (with operands occupying $2n$ words). However, this disadvantage can be minimized in terms of speed by selecting a field size smaller than the word-size boundary because this technique can be exploited more extensively for optimizing double-precision arithmetic.

3.2 Selecting a Field Size Smaller Than the Word-Size Boundary

If the modulus p is selected so that $l = \lceil \log_2 p \rceil < N$, where $N = n \cdot w$, n is the exact number of words required to represent p, i.e., $n = \lceil l/w \rceil$, and w is

the computer word-size, then several consecutive additions without carry-out in the most significant word (MSW) can be performed before a multiplication of the form $c = a \cdot b$, where $a, b \in [0, 2^N - 1]$ such that $c < 2^{2N}$. In the case of Montgomery reduction, the restriction is given by the upper bound $c < 2^N \cdot p$. Similarly, when delaying reductions the result of a multiplication without reduction has maximum value $(p-1)^2 < 2^{2N}$ (assuming that $a, b \in [0, p]$) and several consecutive double-precision additions without carry-outs in the MSW (and, in some cases, subtractions without borrow-outs in the MSW) can be performed before reduction. When using Montgomery reduction up to $\sim \lfloor 2^N/p \rfloor$ additions can be performed without carry checks.

Furthermore, cheaper single- and double-precision operations exploiting this "extra room" can be combined for maximal performance. The challenge is to optimally balance their use in the tower arithmetic since both may interfere with each other. For instance, if intermediate values are allowed to grow up to $2p$ before multiplication (instead of p) then the maximum result would be $4p^2$. This strategy makes use of cheaper single-precision additions without carry checks but limits the number of double-precision additions that can be executed without carry checks after multiplication with delayed reduction. As it will be evident later, to maximize the gain obtained with the proposed methodology one should take into account relative costs of operations and maximum bounds.

In the case of double-precision arithmetic, different optimizing alternatives are available. Let us analyze them in the context of Montgomery arithmetic. First, as pointed out by [7], if $c > 2^N \cdot p$, where c is the result of a double-precision addition, then c can be restored with a cheaper single-precision subtraction by $2^N \cdot p$ (note that the first half of this value consists of zeroes only). Second, different options are available to convert negative numbers to positive after double-precision subtraction. In particular, let us consider the computation $c = a + l \cdot b$, where $a, b \in [0, mp^2]$, $m \in \mathbb{Z}^+$ and $l < 0 \in \mathbb{Z}$ s.t. $|lmp| < 2^N$, which is a recurrent operation (for instance, when $l = \beta$). For this operation, we have explored the following alternatives, which can be integrated in the tower arithmetic with different advantages:

Option 1: $r = c + (2^N \cdot p/2^h),\ r \in [0, mp^2 + 2^N \cdot p/2^h],\ h$ is a small integer s.t. $\left|lmp^2\right| < 2^N \cdot p/2^h < 2^N \cdot p - mp^2$.
Option 2: if $c < 0$ then $r = c + 2^N \cdot p,\ r \in [0, 2^N \cdot p]$.
Option 3: $r = c - lmp^2,\ r \in [0, (|l| + 1)mp^2]$, s.t. $(|l| + 1)mp < 2^N$.
Option 4: if $c < 0$ then $r = c - lmp^2, r \in [0, |lmp^2|]$.

In particular, Options 2 and 4 require conditional checks that make the corresponding operations more expensive. Nevertheless, these options may be valuable when negative values cannot be corrected with other options without violating the upper bound. Also note that Option 2 can make use of a cheaper single-precision subtraction for converting negative results to positive. Options 1 and 3 are particularly efficient because no conditional checks are required. Moreover, if l is small enough (and h maximized for Option 1) several following operations can avoid carry checks. Between both, Option 1 is generally more efficient

because adding $2^N \cdot p/2^h$ requires less than double-precision if $h \leq w$, where w is the computer word-size.

Next, we demonstrate how the different design options discussed in this section can be exploited with a clever selection of parameters and applied to different operations combining single- and double-precision arithmetic to speed up the extension field arithmetic.

3.3 Analysis for Selected Parameters

For our illustrative analysis, we use the tower $\mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^6} \rightarrow \mathbb{F}_{p^{12}}$ constructed with the irreducible binomials described at the beginning of this section. When targeting the 128-bit security level, single- and double-precision operations are defined by operands with sizes $N = 256$ and $2N = 512$, respectively. For our selected prime, $\lceil \log_2 p \rceil = 254$ and $2^N \cdot p \approx 6.8p^2$. Notation is fixed as following: (i) $+, -, \times$ are operators not involving carry handling or modular reduction for boundary keeping; (ii) $\oplus, \ominus, \otimes$ are operators producing reduced results through carry handling or modular reduction; (iii) a superscript in an operator is used to denote the extension degree involved in the operation; (iv) notation $a_{i,j}$ is used to address j-th subfield element in extension field element a_i; (v) lower case t and upper case T variables represent single- and double-precision integers or extension field elements composed of single and double-precision integers, respectively. The precision of the operators is determined by the precision of the operands and result. Note that, as stated before, if $c > 2^N \cdot p$ after adding $c = a + b$ in double-precision, we correct the result by computing $c - 2^N \cdot p$. Similar to subtraction, we refer to the latter as "Option 2".

The following notation is used for the cost of operations: (i) m, s, a denote the cost of multiplication, squaring and addition in $\mathbb{F}_p$, respectively; (ii) $\tilde{m}, \tilde{s}, \tilde{a}, \tilde{i}$ denote the cost of multiplication, squaring, addition and inversion in $\mathbb{F}_{p^2}$, respectively; (iii) m_u, s_u, r denote the cost of unreduced multiplication and squaring producing double-precision results, and modular reduction of double-precision integers, respectively; (iv) $\tilde{m}_u, \tilde{s}_u, \tilde{r}$ denote the cost of unreduced multiplication and squaring, and modular reduction of double-precision elements in $\mathbb{F}_{p^2}$, respectively. For the remainder of the paper, and unless explicitly stated otherwise, we assume that double-precision addition has the cost of $2a$ and $2\tilde{a}$ in $\mathbb{F}_p$ and $\mathbb{F}_{p^2}$, respectively, which approximately follows what we observe in practice.

We will now illustrate a selection of operations for efficient multiplication in $\mathbb{F}_{p^{12}}$, beginning with multiplication in $\mathbb{F}_{p^2}$. Let $a, b, c \in \mathbb{F}_{p^2}$ such that $a = a_0 + a_1 i, b = b_0 + b_1 i, c = a \cdot b = c_0 + c_1 i$. The required operations for computing $\mathbb{F}_{p^2}$ multiplication are detailed in Algorithm 2. As explained in Beuchat *et al.* [7, Section 5.2], when using the Karatsuba method and $a_i, b_i \in \mathbb{F}_p$, $c_1 = (a_0 + a_1)(b_0 + b_1) - a_0 b_0 - a_1 b_1 = a_0 b_1 + a_1 b_0 < 2p^2 < 2^N \cdot p$, additions are single-precision, reduction after multiplication can be delayed and hence subtractions are double-precision (steps 1-3 in Algorithm 2). Obviously, these operations do not require carry checks. For $c_0 = a_0 b_0 - a_1 b_1$, c_0 is in interval $[-p^2, p^2]$ and a negative result can be converted to positive using **Option 1** with $h = 2$ or **Option 2**, for which the final c_0 is in the range $[0, (2^N \cdot p/4) + p^2] \subset [0, 2^N \cdot p]$ or

$[0, 2^N \cdot p]$, respectively (step 4 in Algorithm 2). Following Theorem 1, all reductions can be completely delayed to the next arithmetic layer (higher extension or curve arithmetic).

Algorithm 2. Multiplication in $\mathbb{F}_{p^2}$ without reduction ($\times^2$, cost $\tilde{m}_u = 3m_u + 8a$)

Input: $a = (a_0 + a_1 i)$ and $b = (b_0 + b_1 i) \in \mathbb{F}_{p^2}$
Output: $c = a \cdot b = (c_0 + c_1 i) \in \mathbb{F}_{p^2}$

1. $T_0 \leftarrow a_0 \times b_0$, $T_1 \leftarrow a_1 \times b_1$, $t_0 \leftarrow a_0 + a_1$, $t_1 \leftarrow b_0 + b_1$
2. $T_2 \leftarrow t_0 \times t_1$, $T_3 \leftarrow T_0 + T_1$
3. $T_3 \leftarrow T_2 - T_3$
4. $T_4 \leftarrow T_0 \ominus T_1$ (Option 1 or 2)
5. **return** $c = (T_4 + T_3 i)$

Let us now define multiplication in $\mathbb{F}_{p^6}$. Let $a, b, c \in \mathbb{F}_{p^6}$ such that $a = (a_0 + a_1 v + a_2 v^2), b = (b_0 + b_1 v + b_2 v^2), c = a \cdot b = (c_0 + c_1 v + c_2 v^2)$. The required operations for computing $\mathbb{F}_{p^6}$ multiplication are detailed in Algorithm 3. In this case, $c_0 = v_0 + \xi[(a_1 + a_2)(b_1 + b_2) - v_1 - v_2], c_1 = (a_0 + a_1)(b_0 + b_1) - v_0 - v_1 + \xi v_2$ and $c_2 = (a_0 + a_2)(b_0 + b_2) - v_0 - v_2 + v_1$, where $v_0 = a_0 b_0, v_1 = a_1 b_1$ and $v_2 = a_2 b_2$. First, note that the pattern $s_x = (a_i + a_j)(b_i + b_j) - v_i - v_j$ repeats for each $c_x, 0 \leq x \leq 2$. After multiplications using Alg. 2 with **Option 1** ($h = 2$), we have $v_{i,0}, v_{j,0} \in [0, (2^N \cdot p/4) + p^2]$ and $v_{i,1}, v_{j,1} \in [0, 2p^2]$ (step 1 of Alg. 3). Outputs of single-precision additions of the forms $(a_i + a_j)$ and $(b_i + b_j)$ are in the range $[0, 2p]$ and hence do not produce carries (steps 2, 9 and 17 of Alg. 3). Corresponding $\mathbb{F}_{p^2}$ multiplications $r_x = (a_i + a_j)(b_i + b_j)$ using Alg. 2 with **Option 2** give results in the ranges $r_{x,0} \in [0, 2^N \cdot p]$ and $r_{x,1} \in [0, 8p^2]$ (steps 3, 10 and 18). Although $max(r_{x,1}) = 8p^2 > 2^N \cdot p$, note that $8p^2 < 2^{2N}$ and $s_{x,1} = a_{i,0} b_{j,1} + a_{i,1} b_{j,0} + a_{j,0} b_{i,1} + a_{j,1} b_{i,0} \in [0, 4p^2]$ since $s_x = a_i b_j + a_j b_i$. Hence, for $0 \leq x \leq 2$, double-precision subtractions for computing $s_{x,1}$ using Karatsuba do not require carry checks (steps 4 and 6, 11 and 13, 19 and 21). For computing $s_{x,0} = r_{x,0} - (v_{i,0} + v_{j,0})$, addition does not require carry check (output range $[0, 2(2^N \cdot p/4 + p^2)] \subset [0, 2^N \cdot p]$) and subtraction gives result in the range $[0, 2^N \cdot p]$ when using **Option 2** (steps 5, 12 and 20). For computing c_0, multiplication by ξ, i.e., $S_0 = \xi s_0$ involves the operations $S_{0,0} = s_{0,0} - s_{0,1}$ and $S_{0,1} = s_{0,0} + s_{0,1}$, which are computed in double-precision using **Option 2** to get the output range $[0, 2^N \cdot p]$ (step 7). Similarly, final additions with v_0 require **Option 2** to get again the output range $[0, 2^N \cdot p]$ (step 8). For computing c_1, $S_1 = \xi v_2$ is computed as $S_{1,0} = v_{2,0} - v_{2,1}$ and $S_{1,1} = v_{2,0} + v_{2,1}$, where the former requires a double-precision subtraction using **Option 1** ($h = 1$) to get a result in the range $[0, 2^N \cdot p/2 + 2^N \cdot p/4 + p^2] \subset [0, 2^N \cdot p]$ (step 14) and the latter requires a double-precision addition with no carry check to get a result in the range $[0, (2^N \cdot p/4) + 3p^2] \subset [0, 2^N \cdot p]$ (step 15). Then, $c_{1,0} = s_{1,0} + S_{1,0}$ and $c_{1,1} = s_{1,1} + S_{1,1}$ involve double-precision additions using **Option 2** to obtain results in the range $[0, 2^N \cdot p]$ (step 16). Results $c_{2,0} = s_{2,0} + v_{1,0}$ and $c_{2,1} = s_{2,1} + v_{1,1}$ require a double-precision addition using **Option 2** (final output range $[0, 2^N \cdot p]$, step 22) and a double-precision addition without carry

check (final output range $[0, 6p^2] \subset [0, 2^N \cdot p]$, step 23), respectively. Modular reductions have been delayed again to the last layer $\mathbb{F}_{p^{12}}$.

Algorithm 3. Multiplication in $\mathbb{F}_{p^6}$ without reduction ($\times^6$, cost of $6\tilde{m}_u + 28\tilde{a}$)

Input: $a = (a_0 + a_1 v + a_2 v^2)$ and $b = (b_0 + b_1 v + b_2 v^2) \in \mathbb{F}_{p^6}$
Output: $c = a \cdot b = (c_0 + c_1 v + c_2 v^2) \in \mathbb{F}_{p^6}$

1. $T_0 \leftarrow a_0 \times^2 b_0$, $T_1 \leftarrow a_1 \times^2 b_1$, $T_2 \leftarrow a_2 \times^2 b_2$ (Option 1, $h = 2$)
2. $t_0 \leftarrow a_1 +^2 a_2$, $t_1 \leftarrow b_1 +^2 b_2$
3. $T_3 \leftarrow t_0 \times^2 t_1$ (Option 2)
4. $T_4 \leftarrow T_1 +^2 T_2$
5. $T_{3,0} \leftarrow T_{3,0} \ominus T_{4,0}$ (Option 2)
6. $T_{3,1} \leftarrow T_{3,1} - T_{4,1}$
7. $T_{4,0} \leftarrow T_{3,0} \ominus T_{3,1}$, $T_{4,1} \leftarrow T_{3,0} \oplus T_{3,1}$ ($\equiv T_4 \leftarrow \xi \cdot T_3$) (Option 2)
8. $T_5 \leftarrow T_4 \oplus^2 T_0$ (Option 2)
9. $t_0 \leftarrow a_0 +^2 a_1$, $t_1 \leftarrow b_0 +^2 b_1$
10. $T_3 \leftarrow t_0 \times^2 t_1$ (Option 2)
11. $T_4 \leftarrow T_0 +^2 T_1$
12. $T_{3,0} \leftarrow T_{3,0} \ominus T_{4,0}$ (Option 2)
13. $T_{3,1} \leftarrow T_{3,1} - T_{4,1}$
14. $T_{4,0} \leftarrow T_{2,0} \ominus T_{2,1}$ (Option 1, $h = 1$)
15. $T_{4,1} \leftarrow T_{2,0} + T_{2,1}$ (steps 14-15 $\equiv T_4 \leftarrow \xi \cdot T_2$)
16. $T_6 \leftarrow T_3 \oplus^2 T_4$ (Option 2)
17. $t_0 \leftarrow a_0 +^2 a_2$, $t_1 \leftarrow b_0 +^2 b_2$
18. $T_3 \leftarrow t_0 \times^2 t_1$ (Option 2)
19. $T_4 \leftarrow T_0 +^2 T_2$
20. $T_{3,0} \leftarrow T_{3,0} \ominus T_{4,0}$ (Option 2)
21. $T_{3,1} \leftarrow T_{3,1} - T_{4,1}$
22. $T_{7,0} \leftarrow T_{3,0} \oplus T_{1,0}$ (Option 2)
23. $T_{7,1} \leftarrow T_{3,1} + T_{1,1}$
24. **return** $c = (T_5 + T_6 v + T_7 v^2)$

Finally, let $a, b, c \in \mathbb{F}_{p^{12}}$ such that $a = a_0 + a_1 w, b = b_0 + b_1 w, c = a \cdot b = c_0 + c_1 w$. Algorithm 4 details the required operations for computing multiplication. In this case, $c_1 = (a_0 + a_1)(b_0 + b_1) - a_1 b_1 - a_0 b_0$. At step 1, $\mathbb{F}_{p^6}$ multiplications $a_0 b_0$ and $a_1 b_1$ give outputs in range $\subset [0, 2^N \cdot p]$ using Algorithm 3. Additions $a_0 + a_1$ and $b_0 + b_1$ are single-precision reduced modulo p so that multiplication $(a_0 + a_1)(b_0 + b_1)$ in step 2 gives output in range $\subset [0, 2^N \cdot p]$ using Algorithm 3. Then, subtractions by $a_1 b_1$ and $a_0 b_0$ use double-precision operations with **Option 2** to have an output range $[0, 2^N \cdot p]$ so that we can apply Montgomery reduction at step 5 to obtain the result modulo p. For $c_0 = a_0 b_0 + v a_1 b_1$, multiplication by v, i.e., $T = v \cdot v_1$, where $v_i = a_i b_i$, involves the double-precision operations $T_{0,0} = v_{2,0} - v_{2,1}, T_{0,1} = v_{2,0} + v_{2,1}, T_1 = v_0$ and $T_2 = v_1$, all performed with **Option 2** to obtain the output range $[0, 2^N \cdot p]$ (steps 6-7). Final addition with $a_0 b_0$ uses double-precision with **Option 2** again so that we can apply Montgomery reduction at step 9 to obtain the result modulo p. We remark that, by applying the lazy reduction technique using the operation sequence above, we

have reduced the number of reductions in $\mathbb{F}_{p^6}$ from 3 to only 2, or the number of total modular reductions in $\mathbb{F}_p$ from 54 (or 36 if lazy reduction is employed in $\mathbb{F}_{p^2}$) to only $k = 12$.

Algorithm 4. Multiplication in $\mathbb{F}_{p^{12}}$ ($\times^{12}$, cost of $18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}$)

Input: $a = (a_0 + a_1 w)$ and $b = (b_0 + b_1 w) \in \mathbb{F}_{p^{12}}$
Output: $c = a \cdot b = (c_0 + c_1 w) \in \mathbb{F}_{p^{12}}$

1. $T_0 \leftarrow a_0 \times^6 b_0$, $T_1 \leftarrow a_1 \times^6 b_1$, $t_0 \leftarrow a_0 \oplus^6 a_1$, $t_1 \leftarrow b_0 \oplus^6 b_1$
2. $T_2 \leftarrow t_0 \times^6 t_1$
3. $T_3 \leftarrow T_0 \oplus^6 T_1$ (Option 2)
4. $T_2 \leftarrow T_2 \ominus^6 T_3$ (Option 2)
5. $c_1 \leftarrow T_2 \bmod^6 p$
6. $T_{2,0,0} \leftarrow T_{1,2,0} \ominus T_{1,2,1}$, $T_{2,0,1} \leftarrow T_{1,2,0} \oplus T_{1,2,1}$ (Option 2)
7. $T_{2,1} \leftarrow T_{1,0}$, $T_{2,2} \leftarrow T_{1,1}$ (steps 6-7 $\equiv T_2 \leftarrow v \cdot T_1$)
8. $T_2 \leftarrow T_0 \oplus^6 T_2$ (Option 2)
9. $c_0 \leftarrow T_2 \bmod^6 p$
10. **return** $c = (c_0 + c_1 w)$

As previously stated, there are situations when it is more efficient to perform reductions right after multiplications and squarings in the last arithmetic layer of the tower construction. We illustrate the latter with squaring in $\mathbb{F}_{p^{12}}$. As shown in Algorithm 5, a total of 2 reductions in $\mathbb{F}_{p^6}$ are required when performing $\mathbb{F}_{p^6}$ multiplications in step 4. If lazy reduction was applied, the number of reductions would stay at 2, and worse, the total cost would be increased because some operations would require double-precision. The reader should note that the approach suggested by [10], where the formulas in [28] are employed for computing squarings in internal cubic extensions of $\mathbb{F}_{p^{12}}$, saves $1\tilde{m}$ in comparison with Algorithm 5. However, we experimented such approach with several combinations of formulas and towering, and it remained consistently slower than Algorithm 5 due to an increase in the number of additions.

Algorithm 5. Squaring in $\mathbb{F}_{p^{12}}$ (cost of $12\tilde{m}_u + 6\tilde{r} + 73\tilde{a}$)

Input: $a = (a_0 + a_1 w) \in \mathbb{F}_{p^{12}}$
Output: $c = a^2 = (c_0 + c_1 w) \in \mathbb{F}_{p^{12}}$

1. $t_0 \leftarrow a_0 \oplus^6 a_1, t_{1,0,0} \leftarrow a_{1,2,0} \ominus a_{1,2,1}$, $t_{1,0,1} \leftarrow a_{1,2,0} \oplus a_{1,2,1}$
2. $t_{1,1} \leftarrow a_{1,0}$, $t_{1,2} \leftarrow a_{1,1}$ (steps 2-3 $\equiv t_1 \leftarrow v \cdot a_1$)
3. $t_1 \leftarrow a_0 \oplus^6 t_1$
4. $c_1 \leftarrow (a_0 \times^6 a_1) \bmod^6 p$, $t_0 \leftarrow (t_0 \times^6 t_1) \bmod^6 p$
5. $t_{1,0,0} \leftarrow c_{1,2,0} \ominus c_{1,2,1}$, $t_{1,0,1} \leftarrow c_{1,2,0} \oplus c_{1,2,1}$
6. $t_{1,1} \leftarrow c_{1,0}$, $t_{1,2} \leftarrow c_{1,1}$ (steps 6-7 $\equiv t_1 \leftarrow v \cdot c_1$)
7. $t_1 \leftarrow t_1 \oplus^6 c_1$
8. $c_0 \leftarrow t_0 \ominus^6 t_1$, $c_1 \leftarrow c_1 \oplus^6 c_1$
9. **return** $c = (c_0 + c_1 w)$

4 Miller Loop

In this section, we present our optimizations to the curve arithmetic. To be consistent with other results in the literature, we do not distinguish between simple- and double-precision additions in the formulas below.

Recently, Costello *et al.* [9, Section 5] proposed the use of homogeneous coordinates to perform the curve arithmetic entirely on the twist. Their formula for computing a point doubling and line evaluation costs $2\tilde{m}+7\tilde{s}+23\tilde{a}+4m+1m_{b'}$. The twisting of point P, given in our case by $(x_P/w^2, y_P/w^3) = (\frac{x_P}{\xi}v^2, \frac{y_P}{\xi}vw)$, is eliminated by multiplying the whole line evaluation by ξ and relying on the final exponentiation to eliminate this extra factor [9]. Clearly, the main drawback of this formula is the high number of additions. We present the following revised formula:

$$X_3 = \frac{X_1Y_1}{2}\left(Y_1^2 - 9b'Z_1^2\right), \quad Y_3 = \left[\frac{1}{2}\left(Y_1^2 + 9b'Z_1^2\right)\right] - 27b'^2Z_1^4, \quad Z_3 = 2Y_1^3Z_1,$$
$$l = (-2Y_1Z_1y_P)vw + \left(3X_1^2x_P\right)v^2 + \xi\left(3b'Z_1^2 - Y_1^2\right). \tag{2}$$

This doubling formula gives the cost of $3\tilde{m}+6\tilde{s}+17\tilde{a}+4m+m_{b'}+m_{\xi}$. Moreover, if the parameter b' is cleverly selected as in [10], multiplication by b' can be performed with minimal number of additions and subtractions. For instance, if one fixes $b = 2$ then $b' = 2/(1+i) = 1-i$. Accordingly, the following execution has a cost of $3\tilde{m} + 6\tilde{s} + 19\tilde{a} + 4m$ (note that computations for E and $l_{0,0}$ are over $\mathbb{F}_p$ and $\overline{y_P} = -y_P$ is precomputed):

$$A = X_1 \cdot Y_1/2, \quad B = Y_1^2, \quad C = Z_1^2, \quad D = 3C, \quad E_0 = D_0 + D_1,$$
$$E_1 = D_1 - D_0, \quad F = 3E, \quad X_3 = A \cdot (B - F), \quad G = (B + F)/2,$$
$$Y_3 = G^2 - 3E^2, \quad H = (Y_1 + Z_1)^2 - (B + C), \tag{3}$$
$$Z_3 = B \cdot H, \quad I = E - B, \quad J = X_1^2$$
$$l_{0,0,0} = I_0 - I_1, \quad l_{0,0,1} = I_0 + I_1, \quad l_{1,1} = H \cdot \overline{y_P}, \quad l_{0,2} = 3J \cdot x_P.$$

We point out that in practice we have observed that $\tilde{m} - \tilde{s} \approx 3\tilde{a}$. Hence, it is more efficient to compute X_1Y_1 directly than using $(X_1 + Y_1)^2, B$ and J. If this was not the case, the formula could be computed with cost $2\tilde{m}+7\tilde{s}+23\tilde{a}+4m$.

Remarkably, the technique proposed in Section 3 for delaying reductions can also be applied to the point arithmetic over a quadratic extension field. Reductions can be delayed to the end of each $\mathbb{F}_{p^2}$ multiplication/squaring and then delayed further for those sums of products that allow reducing the number of reductions. Although not plentiful (given the nature of most curve arithmetic formulas which have consecutive and redundant multiplications/squarings), there are a few places where this technique can be applied. For instance, doubling formula (2) requires 25 $\mathbb{F}_p$ reductions (3 per $\mathbb{F}_{p^2}$ multiplication using Karatsuba, 2 per $\mathbb{F}_{p^2}$ squaring and 1 per $\mathbb{F}_p$ multiplication). First, by delaying reductions inside $\mathbb{F}_{p^2}$ arithmetic the number of reductions per multiplication goes down to only 2, with 22 reductions in total. Moreover, reductions corresponding to G^2

and $3E^2$ in Y_3 (see execution (3)) can be further delayed and merged, eliminating the need of two reductions. In total, the number of reductions is now 20. Similar optimizations can be applied to other point/line evaluation formulas (see extended version [29] for optimizations to formulas using Jacobian and homogeneous coordinates).

For accumulating line evaluations into the Miller variable, $\mathbb{F}_{p^{12}}$ is represented using the towering $\mathbb{F}_{p^2} \rightarrow \mathbb{F}_{p^4} \rightarrow \mathbb{F}_{p^{12}}$ and a special (dense×sparse)-multiplication costing $13\tilde{m}_u + 6\tilde{r} + 61\tilde{a}$ is used. During the first iteration of the loop, a squaring in $\mathbb{F}_{p^{12}}$ can be eliminated since the Miller variable is initialized as 1 (line 1 in Algorithm 1) and a special (sparse×sparse) multiplication costing $7\tilde{m}_u + 5\tilde{r} + 30\tilde{a}$ is used to multiply the first two line evaluations, resulting in the revised Algorithm 6. This sparser multiplication is also used for multiplying the two final line evaluations in step 10 of the algorithm.

5 Final Exponentiation

The fastest way known for computing the final exponentiation is described in [30]. The power $\frac{p^{12}-1}{n}$ is factored into an easy exponent (p^6-1) which requires a conjugation and an inversion; another easy exponent (p^2+1) which requires a p^2-power Frobenius and a multiplication; and a hard exponent $(p^4-p^2+1)/n$ which can be performed in the cyclotomic subgroup $\mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})$. For computing this last power, one can write the hard exponent as follows [12]:

$$(p^4 - p^2 + 1)/n = \lambda_3 p^3 + \lambda_2 p^2 + \lambda_1 p + \lambda_0,$$

where

$$\lambda_3(u) = 1\ ,\ \lambda_2(u) = 6u^2 + 1,$$
$$\lambda_1(u) = -36u^3 - 18u^2 - 12u + 1\ ,\ \lambda_0(u) = -36u^3 - 30u^2 - 18u - 2,$$

and compute the individual powers by a multi-addition chain, requiring three consecutive exponentiations by the absolute value of the curve parameter $|u|$, 13 multiplications, 4 squarings, 4 p-power Frobenius, 2 p^2-power Frobenius and a single p^3-power Frobenius in $\mathbb{F}_{p^{12}}$. These powers of Frobenius can be efficiently computed with the formulas in [7]. In the following subsections, we explain how to remove the expensive inversion in $\mathbb{F}_{p^{12}}$ mentioned at the end of Section 2; and how the cyclotomic subgroup structure allows faster compressed squarings and consequently faster exponentiation by $|u|$.

5.1 Removing the Inversion Penalty

From Algorithm 1, the Optimal Ate pairing when $u < 0$ can be computed as

$$a_{opt}(Q,P) = \left[g^{-1} \cdot h\right]^{\frac{p^{12}-1}{n}}, \tag{4}$$

with $r = 6u+2, g = f_{|r|,Q}(P)$ and $h = l_{[-|r|]Q,\pi_p(Q)}(P) \cdot l_{[-|r|Q]+\pi_p(Q),-\pi_p^2(Q)}(P)$. Lemma 1 below allows one to replace the expensive inversion g^{-1} with a simple conjugation with no change in the result. This is depicted in line 9 of Algorithm 6.

Lemma 1. *The pairing $a_{opt}(Q,P)$ can be computed as $\left[g^{p^6}\cdot h\right]^{\frac{p^{12}-1}{n}}$, with g,h defined as above.*

Proof. By distributing the power $(p^{12}-1)/n$ in terms g,h in Equation (4):

$$a_{opt}(Q,P) = g^{\frac{1-p^{12}}{n}}\cdot h^{\frac{p^{12}-1}{n}} = g^{\frac{(1-p^6)(1+p^6)}{n}}\cdot h^{\frac{p^{12}-1}{n}}$$
$$= g^{\frac{(p^{12}-p^6)(1+p^6)}{n}}\cdot h^{\frac{p^{12}-1}{n}} = g^{\frac{p^6(p^6-1)(p^6+1)}{n}}\cdot h^{\frac{p^{12}-1}{n}} = \left[g^{p^6}\cdot h\right]^{\frac{p^{12}-1}{n}} \quad \square$$

5.2 Computing u-th Powers in $\mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})$

Let

$$g = \sum_{i=0}^{2}(g_{2i}+g_{2i+1}s)t^i \in \mathbb{G}_{\phi_6}(\mathbb{F}_{p^2}) \text{ and } g^2 = \sum_{i=0}^{2}(h_{2i}+h_{2i+1}s)t^i$$

with $g_i, h_i \in \mathbb{F}_{p^2}$. In [31], it was shown that one can compress g to $\mathcal{C}(g) = [g_2,g_3,g_4,g_5]$, and the compressed representation of g^2 is computed as $\mathcal{C}(g^2) = [h_2,h_3,h_4,h_5]$, where h_i is computed as follows:

$$\begin{aligned} h_2 &= 2(g_2+3\xi B_{4,5}), & h_3 &= 3(A_{4,5}-(\xi+1)B_{4,5})-2g_3,\\ h_4 &= 3(A_{2,3}-(\xi+1)B_{2,3})-2g_4, & h_5 &= 2(g_5+3B_{2,3}), \end{aligned} \tag{5}$$

where $A_{i,j} = (g_i+g_j)(g_i+\xi g_j)$ and $B_{i,j} = g_ig_j$. The above formula requires 4 multiplications in $\mathbb{F}_{p^2}$. Considering the lazy reduction technique discussed in Section 3.3, we propose another formula that is slightly faster and has a cost of $6\tilde{s}_u + 4\tilde{r} + 31\tilde{a}$. The formula is given as follows:

$$\begin{aligned} h_2 &= 2g_2+3(S_{4,5}-S_4-S_5)\xi, & h_3 &= 3(S_4+S_5\xi)-2g_3,\\ h_4 &= 3(S_2+S_3\xi)-2g_4, & h_5 &= 2g_5+3(S_{2,3}-S_2-S_3), \end{aligned} \tag{6}$$

where $S_{i,j} = (g_i+g_j)^2$ and $S_i = g_i^2$; also see extended version [29] for the correctness of our formula and an explicit implementation.

When g is raised to a power via a square-and-multiply exponentiation algorithm, full representation of elements (decompression) is required because, if $\mathcal{C}$ is used as the compression map, it is not known how to perform multiplication given the compressed representation of elements. Given a compressed representation of $g \in \mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})\setminus\{1\}$, $\mathcal{C}(g) = [g_2,g_3,g_4,g_5]$, the decompression map $\mathcal{D}$ is evaluated as follows (see [31] for more details):

$$\mathcal{D}([g_2,g_3,g_4,g_5]) = (g_0+g_1s)+(g_2+g_3s)t+(g_4+g_5s)t^2,$$

$$\begin{cases} g_1 = \frac{g_5^2\xi+3g_4^2-2g_3}{4g_2}, \quad g_0 = (2g_1^2+g_2g_5-3g_3g_4)\xi+1, & \text{if } g_2\neq 0;\\ g_1 = \frac{2g_4g_5}{g_3}, \quad g_0 = (2g_1^2-3g_3g_4)\xi+1, & \text{if } g_2 = 0. \end{cases}$$

In particular, $g^{|u|}$ can be computed in three steps:

1. Compute $\mathcal{C}(g^{2^i})$ for $1 \leq i \leq 62$ using (6) and store $\mathcal{C}(g^{2^{55}})$ and $\mathcal{C}(g^{2^{62}})$.
2. Compute $\mathcal{D}(\mathcal{C}(g^{2^{55}})) = g^{2^{55}}$ and $\mathcal{D}(\mathcal{C}(g^{2^{62}})) = g^{2^{62}}$.
3. Compute $g^{|u|} = g^{2^{62}} \cdot g^{2^{55}} \cdot g$.

Step 1 requires 62 squarings in $\mathbb{G}_{\phi_6}(\mathbb{F}_{p^2})$. Using Montgomery's simultaneous inversion trick [32], Step 2 requires $9\tilde{m} + 6\tilde{s} + 22\tilde{a} + \tilde{i}$. Step 3 requires 2 multiplications in $\mathbb{F}_{p^{12}}$. The total cost is:

$$\begin{aligned} Exp &= 62 \cdot (6\tilde{s}_u + 4\tilde{r} + 31\tilde{a}) + (9\tilde{m} + 6\tilde{s} + 22\tilde{a} + \tilde{i}) + 2 \cdot (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &= 45\tilde{m}_u + 378\tilde{s}_u + 275\tilde{r} + 2164\tilde{a} + \tilde{i}, \end{aligned}$$

Granger-Scott's [33] formula for squaring can be implemented at a cost of $9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}$ if lazy reduction techniques are employed. With this approach, an exponentiation costs:

$$\begin{aligned} Exp' &= 62 \cdot (9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}) + 2 \cdot (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &= 36\tilde{m}_u + 558\tilde{s}_u + 399\tilde{r} + 3072\tilde{a}. \end{aligned}$$

Hence, the faster compressed squaring formulas reduce by 33% the number of squarings and by 30% the number of additions in $\mathbb{F}_{p^2}$.

Algorithm 6. Revised Optimal Ate pairing on BN curves (generalized for $u < 0$).

Input: $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, r = |6u + 2| = \sum_{i=0}^{\log_2(r)} r_i 2^i$
Output: $a_{opt}(Q, P)$

1. $d \leftarrow l_{Q,Q}(P), T \leftarrow 2Q, e \leftarrow 1$
2. **if** $r_{\lfloor \log_2(r) \rfloor - 1} = 1$ **then** $e \leftarrow l_{T,Q}(P), T \leftarrow T + Q$
3. $f \leftarrow d \cdot e$
4. **for** $i = \lfloor \log_2(r) \rfloor - 2$ **downto** 0 **do**
5. $\quad f \leftarrow f^2 \cdot l_{T,T}(P), T \leftarrow 2T$
6. $\quad$ **if** $r_i = 1$ **then** $f \leftarrow f \cdot l_{T,Q}(P), T \leftarrow T + Q$
7. **end for**
8. $Q_1 \leftarrow \pi_p(Q), Q_2 \leftarrow \pi_p^2(Q)$
9. **if** $u < 0$ **then** $T \leftarrow -T, f \leftarrow f^{p^6}$
10. $d \leftarrow l_{T,Q_1}(P), T \leftarrow T + Q_1, e \leftarrow l_{T,-Q_2}(P), T \leftarrow T - Q_2,\ f \leftarrow f \cdot (d \cdot e)$
11. $f \leftarrow f^{(p^6-1)(p^2+1)(p^4-p^2+1)/n}$
12. **return** f

6 Computational Cost

We now consider all the improvements described in the previous sections and present a detailed operation count. Table 1 shows the exact operation count for each operation executed in Miller's Algorithm.

Table 1. Operation counts for arithmetic required by Miller's Algorithm. (†) Work [7] counts these additions in a different way. Considering their criteria, costs for multiplication and squaring in $\mathbb{F}_{p^2}$ are $3m_u + 2r + 4a$ and $2m_u + 2r + 2a$, respectively.

$E'(\mathbb{F}_{p^2})$-Arithmetic	Operation Count
Doubling/Eval.	$3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 22\tilde{a} + 4m$
Addition/Eval.	$11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 12\tilde{a} + 4m$
p-power Frobenius	$6m_u + 4r + 18a$
p^2-power Frobenius	$2m + 2a$
Negation	$\tilde{a}$
$\mathbb{F}_{p^2}$-Arithmetic	Operation Count
Add./Sub./Neg.	$\tilde{a} = 2a$
Conjugation	a
Multiplication	$\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r + 8a^{\dagger}$
Squaring	$\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r + 3a^{\dagger}$
Multiplication by β	a
Multiplication by ξ	$2a$
Inversion	$\tilde{i}$

$\mathbb{F}_{p^{12}}$-Arithmetic	Operation Count
Add./Sub.	$6\tilde{a}$
Conjugation	$3\tilde{a}$
Multiplication	$18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}$
Sparse Mult.	$13\tilde{m}_u + 6\tilde{r} + 61\tilde{a}$
Sparser Mult.	$7\tilde{m}_u + 5\tilde{r} + 30\tilde{a}$
Squaring	$12\tilde{m}_u + 6\tilde{r} + 73\tilde{a}$
Cyc. Squaring	$9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}$
Comp. Squaring	$6\tilde{s}_u + 4\tilde{r} + 31\tilde{a}$
Simult. Decomp.	$9\tilde{m} + 6\tilde{s} + 22\tilde{a} + \tilde{i}$
p-power Frobenius	$15m_u + 10r + 46a$
p^2-power Frobenius	$10m + 2\tilde{a}$
Inversion	$25\tilde{m}_u + 9\tilde{s}_u + 24\tilde{r} + 112\tilde{a} + \tilde{i}$

For the selected parameters and with the presented improvements, the Miller Loop in Algorithm 6 executes 64 point doublings with line evaluations, 6 point additions with line evaluations (4 inside Miller Loop and 2 more at the final steps), 1 negation in $\mathbb{F}_{p^2}$ to precompute $\overline{y_P}$, 1 p-power Frobenius, 1 p^2-power Frobenius and 2 negations in $E(\mathbb{F}_{p^2})$; and 1 conjugation, 1 multiplication, 66 sparse multiplications, 2 sparser multiplications and 63 squarings in $\mathbb{F}_{p^{12}}$. The cost of the Miller Loop is:

$$\begin{aligned} ML &= 64 \cdot (3\tilde{m}_u + 6\tilde{s}_u + 8\tilde{r} + 22\tilde{a} + 4m) + 6 \cdot (11\tilde{m}_u + 2\tilde{s}_u + 11\tilde{r} + 12\tilde{a} + 4m) \\ &\quad + \tilde{a} + 6m_u + 4r + 18a + 2m + 2a + 2\tilde{a} + 3\tilde{a} + (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &\quad + 66 \cdot (13\tilde{m}_u + 6\tilde{r} + 61\tilde{a}) + 2 \cdot (7\tilde{m}_u + 5\tilde{r} + 30\tilde{a}) + 63 \cdot (12\tilde{m}_u + 6\tilde{r} + 73\tilde{a}) \\ &= 1904\tilde{m}_u + 396\tilde{s}_u + 1368\tilde{r} + 10281\tilde{a} + 282m + 6m_u + 4r + 20a. \end{aligned}$$

The final exponentiation executes in total 1 inversion, 4 conjugations, 15 multiplications, 3 u-th powers, 4 cyclotomic squarings, 5 p-power Frobenius, 3 p^2-power Frobenius:

$$\begin{aligned} FE &= 25\tilde{m}_u + 9\tilde{s}_u + 24\tilde{r} + 112\tilde{a} + \tilde{i} + 4 \cdot 3\tilde{a} + 15 \cdot (18\tilde{m}_u + 6\tilde{r} + 110\tilde{a}) \\ &\quad + 3 \cdot Exp + 4 \cdot (9\tilde{s}_u + 6\tilde{r} + 46\tilde{a}) + 5 \cdot (15m_u + 10r + 46a) + 3 \cdot (10m + 2\tilde{a}) \\ &= 430\tilde{m}_u + 1179\tilde{s}_u + 963\tilde{r} + 8456\tilde{a} + 4\tilde{i} + 30m + 75m_u + 50r + 230a. \end{aligned}$$

Table 2 gives a first-order comparison between our implementation and the best implementation available in the literature of the Optimal Ate pairing at the 128-bit security level in the same platform. For the related work, we suppose that lazy reduction is always used in $\mathbb{F}_{p^2}$ and then each multiplication or squaring essentially computes a modular reduction (that is, $\tilde{m} = \tilde{m}_u + \tilde{r} = 3m_u + 2r$ and $\tilde{s} = \tilde{s}_u + \tilde{r} = 2m_u + 2r$). Note that our generalization of the lazy reduction techniques to the whole pairing computation brings the number of modular reductions from the expected 7818 (if lazy reduction was only used for $\mathbb{F}_{p^2}$ arithmetic) to just 4662, avoiding more than 40% of the total required modular reductions. The number of multiplications is also reduced by 13% and the

number of additions is increased by 26% due to lazy reduction trade-offs. Our operation count for the pairing computation is apparently more expensive than Pereira *et al.* [10]. However, the reader should note that, when we consider the real cost of additions in $\mathbb{F}_p$, we cannot exploit the squaring formula in $\mathbb{F}_{p^{12}}$ by [28] (see Section 3.3) and a point doubling formula with fewer multiplications (see Section 4), given the significant increase in the number of additions.

Table 2. Comparison of operation counts for different implementations of the Optimal Ate pairing at the 128-bit security level

Work	Phase	**Operations in $\mathbb{F}_{p^2}$**	**Operations in $\mathbb{F}_p$**
Beuchat *et al.*[7]	ML	$1952(\tilde{m}_u + \tilde{r}) + 568(\tilde{s}_u + \tilde{r}) + 6912\tilde{a}$	$6992m_u + 5040r$
	FE	$403(\tilde{m}_u + \tilde{r}) + 1719(\tilde{s}_u + \tilde{r}) + 7021\tilde{a}$	$4647m_u + 4244r$
	ML+FE	$2355(\tilde{m}_u + \tilde{r}) + 2287(\tilde{s}_u + \tilde{r}) + 13933\tilde{a}$	$11639m_u + 9284r$
This work	ML	$1904\tilde{m}_u + 396\tilde{s}_u + 1368\tilde{r} + 10281\tilde{a}$	$6504m_u + 2736r$
	FE	$430\tilde{m}_u + 1179\tilde{s}_u + 963\tilde{r} + 8456\tilde{a}$	$3648m_u + 1926r$
	ML+FE	$2334\tilde{m}_u + 1575\tilde{s}_u + 2331\tilde{r} + 18737\tilde{a}$	$10152m_u + 4662r$

7 Implementation Results

A software implementation was realized to confirm the performance benefits resulting from the introduced techniques. We implemented $\mathbb{F}_{p^2}$ arithmetic directly in Assembly, largely following advice from [7] to optimize carry handling and eliminate function call overheads. Higher-level algorithms were implemented using the C programming language compiled with the GCC compiler using `-O3` optimization level. Table 3 presents the relevant timings in millions of cycles. Basic Implementation employs homogeneous projective coordinates and lazy reduction below $\mathbb{F}_{p^2}$. Faster arithmetic in cyclotomic subgroups accelerates the Basic Implementation by 5%-7% and, in conjunction with generalized lazy reduction, it improves the Basic Implementation by 18%-22%.

Table 3. Cumulative performance improvement when using new arithmetic in cyclotomic subgroups (Section 5.2) and generalized lazy reduction (Section 3.1) on several Intel and AMD 64-bit architectures. Improvements are calculated relatively to the Basic Implementation. Timings are presented in millions of clock cycles.

	This work							
Method	Phenom II	Impr.	Core i5	Impr.	Opteron	Impr.	Core 2	Impr.
Basic Implementation	1.907	-	2.162	-	2.127	-	2.829	-
Cyclotomic Formulas	1.777	7%	2.020	7%	2.005	6%	2.677	5%
Lazy Reduction	1.562	18%	1.688	22%	1.710	20%	2.194	22%

Table 4 compares our implementation with related work. To ensure that machines with different configurations but belonging to the same microarchitecture had compatible performance (as is the case with Core i5 and Core i7), software from [7] was benchmarked and the results compared with the ones reported in [7].

Machines considered equivalent by this criteria are presented in the same column. We note that Phenom II was not considered in the original study and that we could not find a Core 2 Duo machine producing the same timings as in [7]. For this reason, timings for these two architectures were taken independently by the authors using the available software. Observe that the Basic Implementation in Table 3 consistently outperforms Beuchat *et al.* due to our careful implementation of an optimal choice of parameters $(E(\mathbb{F}_p) : y^2 = x^3 + 2, p = 3 \mod 4)$ [10] combined with optimized curve arithmetic in homogeneous coordinates [9]. When lazy reduction and faster cyclotomic formulas are enabled, pairing computation becomes faster than the best previous result by 28%-34%. For extended benchmark results and comparisons with previous works on different 64-bit processors, the reader is referred to our online database [34].

Table 4. Comparison between implementations on 64-bit architectures. Timings are presented in clock cycles.

	Work/Platform			
	Beuchat *et al.* [7]			
Operation	Phenom II	Core i7	Opteron	Core 2 Duo
Multiplication in $\mathbb{F}_{p^2}$	440	435	443	590
Squaring in $\mathbb{F}_{p^2}$	353	342	355	479
Miller Loop	1,338,000	1,330,000	1,360,000	1,781,000
Final Exponentiation	1,020,000	1,000,000	1,040,000	1,370,000
Optimal Ate Pairing	2,358,000	2,330,000	2,400,000	3,151,000
	This work			
Operation	Phenom II	Core i5	Opteron	Core 2 Duo
Multiplication in $\mathbb{F}_{p^2}$	368	412	390	560
Squaring in $\mathbb{F}_{p^2}$	288	328	295	451
Miller Loop	898,000	978,000	988,000	1,275,000
Final Exponentiation	664,000	710,000	722,000	919,000
Optimal Ate Pairing	1,562,000	1,688,000	1,710,000	2,194,000
Improvement	34%	28%	29%	30%

8 Conclusion

In this work, we revisited the problem of computing optimal pairings on ordinary pairing-friendly curves over prime fields. Several new techniques were introduced for pairing computation, comprised mainly in the generalization of lazy reduction techniques to arithmetic in extensions above $\mathbb{F}_{p^2}$ and inside curve arithmetic; and improvements to the final exponentiation consisting of a formula for compressed squaring in cyclotomic subgroups and an arithmetic trick to remove penalties from negative curve parameterizations. The faster arithmetic in the cyclotomic subgroup improved pairing performance by 5%-7% and the generalized lazy reduction technique was able to eliminate 40% of the required modular reductions, improving pairing performance by further 11%-17%. The introduced techniques allow for the first time a pairing computation under 2 million cycles on 64-bit

desktop computing platforms, improving the state-of-the-art by 28%-34%. The performance improvements are expected to be even higher on embedded architectures, where the ratio between multiplication and addition is typically higher.

Acknowledgements

We would like to express our gratitude to Alfred Menezes, Craig Costello, Michael Scott, Paulo S. L. M. Barreto, Geovandro C. C. F. Pereira and Conrado P. L. Gouvêa for useful discussions during the preparation of this work. The authors thank the Natural Sciences and Engineering Research Council of Canada (NSERC), the Ontario Centres of Excellence (OCE), CNPq, CAPES and FAPESP for partially supporting this work.

References

1. Boneh, D., Franklin, M.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
2. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems Based on Pairing over Elliptic Curve. In: The 2001 Symposium on Cryptography and Information Security. IEICE, Oiso (2001) (in Japanese)
3. 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)
4. Joux, A.: A One Round Protocol for Tripartite Diffie-Hellman. Journal of Cryptology 17(4), 263–276 (2004)
5. Hankerson, D., Menezes, A., Scott, M.: Identity-Based Cryptography, ch. 12, pp. 188–206. IOS Press, Amsterdam (2008)
6. Naehrig, M., Niederhagen, R., Schwabe, P.: New Software Speed Records for Cryptographic Pairings. In: Abdalla, M., Barreto, P.S.L.M. (eds.) LATINCRYPT 2010. LNCS, vol. 6212, pp. 109–123. Springer, Heidelberg (2010)
7. Beuchat, J.L., Díaz, J.E.G., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
8. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory 56(1), 455–461 (2010)
9. Costello, C., Lange, T., Naehrig, M.: Faster Pairing Computations on Curves with High-Degree Twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 224–242. Springer, Heidelberg (2010)
10. Pereira, G.C.C.F., Simplício Jr, M.A., Naehrig, M., Barreto, P.S.L.M.: A Family of Implementation-Friendly BN Elliptic Curves. To appear in Journal of Systems and Software
11. Barreto, P.S.L.M., Naehrig, M.: Pairing-Friendly Elliptic Curves of Prime Order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)

12. Scott, M.: Implementing Cryptographic Pairings. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 197–207. Springer, Heidelberg (2007)
13. Fan, J., Vercauteren, F., Verbauwhede, I.: Faster F_p-arithmetic for Cryptographic Pairings on Barreto-Naehrig Curves. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 240–253. Springer, Heidelberg (2009)
14. Freeman, D., Scott, M., Teske, E.: A Taxonomy of Pairing-Friendly Elliptic Curves. Journal of Cryptology 23(2), 224–280 (2010)
15. Hess, F., Smart, N.P., Vercauteren, F.: The Eta Pairing Revisited. IEEE Transactions on Information Theory 52, 4595–4602 (2006)
16. Lee, E., Lee, H.-S., Park, C.-M.: Efficient and Generalized Pairing Computation on Abelian Varieties. IEEE Transactions on Information Theory 55(4), 1793–1803 (2009)
17. Nogami, Y., Akane, M., Sakemi, Y., Kato, H., Morikawa, Y.: Integer Variable χ-Based Ate Pairing. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 178–191. Springer, Heidelberg (2008)
18. Miller, V.: Uses of Elliptic Curves in Cryptography. In: Williams, H.C. (ed.) CRYPTO 1985. LNCS, vol. 218, pp. 417–426. Springer, Heidelberg (1986)
19. Miller, V.S.: The Weil Pairing, and its Efficient Calculation. Journal of Cryptology 17(4), 235–261 (2004)
20. Barreto, P.S.L.M., Kim, H.Y., Lynn, B., Scott, M.: Efficient Algorithms for Pairing-Based Cryptosystems. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 354–368. Springer, Heidelberg (2002)
21. IEEE: P1363.3: Standard for Identity-Based Cryptographic Techniques using Pairings (2006), http://grouper.ieee.org/groups/1363/IBC/submissions/
22. Devegili, A.J., Scott, M., Dahab, R.: Implementing Cryptographic Pairings over Barreto-Naehrig Curves. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 197–207. Springer, Heidelberg (2007)
23. Weber, D., Denny, T.F.: The Solution of McCurley's Discrete Log Challenge. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 458–471. Springer, Heidelberg (1998)
24. Lim, C.H., Hwang, H.S.: Fast Implementation of Elliptic Curve Arithmetic in GF(p^n). In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 405–421. Springer, Heidelberg (2000)
25. 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)
26. Benger, N., Scott, M.: Constructing Tower Extensions of Finite Fields for Implementation of Pairing-Based Cryptography. In: Hasan, M.A., Helleseth, T. (eds.) WAIFI 2010. LNCS, vol. 6087, pp. 180–195. Springer, Heidelberg (2010)
27. Montgomery, P.L.: Modular Multiplication Without Trial Division. Mathematics of Computation 44(170), 519–521 (1985)
28. Chung, J., Hasan, M.: Asymmetric Squaring Formulae. In: 18th IEEE Symposium on Computer Arithmetic (ARITH-18 2007), pp. 113–122. IEEE Press, Los Alamitos (2007)
29. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster Explicit Formulas for Computing Pairings over Ordinary Curves. Cryptology ePrint Archive, Report 2010/526 (2010), http://eprint.iacr.org/

30. Scott, M., Benger, N., Charlemagne, M., Perez, L.J.D., Kachisa, E.J.: On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 78–88. Springer, Heidelberg (2009)
31. Karabina, K.: Squaring in Cyclotomic Subgroups. Cryptology ePrint Archive, Report 2010/542 (2010), http://eprint.iacr.org/
32. Montgomery, P.: Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation 48, 243–264 (1987)
33. Granger, R., Scott, M.: Faster Squaring in the Cyclotomic Subgroup of Sixth Degree Extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 209–223. Springer, Heidelberg (2010)
34. Longa, P.: Speed Benchmarks for Pairings over Ordinary Curves, http://www.patricklonga.bravehost.com/speed_pairing.html#speed

Pushing the Limits: A Very Compact and a Threshold Implementation of AES

Amir Moradi[1], Axel Poschmann[2,⋆],
San Ling[2,⋆], Christof Paar[1], and Huaxiong Wang[2,⋆]

[1] Horst Görtz Institute for IT Security, Ruhr University Bochum, Germany
{moradi,cpaar}@crypto.rub.de
[2] Division of Mathematical Sciences, School of Physical and Mathematical Sciences,
Nanyang Technological University, Singapore
{aposchmann,lingsan,hxwang}@ntu.edu.sg

Abstract. Our contribution is twofold: first we describe a very compact hardware implementation of AES-128, which requires only 2400 GE. This is to the best of our knowledge the smallest implementation reported so far. Then we apply the threshold countermeasure by Nikova *et al.* to the AES S-box and yield an implementation of the AES improving the level of resistance against first-order side-channel attacks. Our experimental results on real-world power traces show that although our implementation provides additional security, it is still susceptible to some sophisticated attacks having enough number of measurements.

Keywords: side-channel attacks, countermeasures, secret sharing, lightweight, ASIC.

1 Introduction

The mass deployment of pervasive devices promises many benefits such as lower logistic costs, higher process granularity, optimized supply-chains, or location based services among others. Besides these benefits, there are also many risks inherent in pervasive computing: many foreseen applications are security sensitive, such as wireless sensor networks for military, financial or automotive applications. With the widespread presence of embedded computers in such scenarios, security is a striving issue, because the potential damage of malicious attacks also increases. An aggravating factor is that pervasive devices are usually not deployed in a controlled but rather in a hostile environment, *i.e.*, an adversary has physical access to or control over the devices. This adds the whole field of physical attacks to the potential attack scenarios. Most notably are here so-called *side-channel attacks*, especially *Simple*, *Differential* and *Correlation Power Analysis* [6,18].

⋆ The authors were supported in part by the Singapore National Research Foundation under Research Grant NRF-CRP2-2007-03.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 69–88, 2011.

1.1 Related Work

Low-power low-area implementations of the AES have been reported in [15] requiring 3100 GE and 160 clock cycles and in [13] requiring 3400 GE and 1032 clock cycles. Both implementations use an 8-bit serialized data path and implement only a quarter of the *MixColumns* operations. The first design, [15], implements two S-boxes and performs the datapath and the key schedule operations in parallel, while the latter implementation is fully serial and uses a RAM-like architecture.

Canright has investigated very thoroughly how to implement the AES S-box in hardware with minimal area requirements [8]. On the other hand, several masking schemes have been proposed to create a masked AES S-box using either multiplicative or additive approaches. A common approach is to use the tower-field representation for an additive masking scheme because of the linearity of the inversion in $GF(2^2)$. The examples are [4] and [26] which are provably secure, but in practice obvious first-order leakages have been observed [20]. Later, Canright et al. [9] applied the idea of [26] to his very compact S-box resulting in the most compact masked S-box to date. However, as expected its hardware implementation still has first-order leakage [21].

1.2 Our Work

Our first contribution is a description of the smallest hardware implementation of AES known to date. Our design goal was solely low area, and thus we were able to set the time-area and the power-area tradeoffs differently, and in favour for a more compact hardware realization, compared to [13] and [15]. To pursue our goal, we have taken a holistic approach that optimizes the total design, not every component individually. In total we achieved an implementation that requires only 2400 GE and needs 226 clock cycles, which is to the best of our knowledge 23% smaller than any previously published implementations.

As a second contribution, we investigate side-channel countermeasures for this lightweight AES implementation. It turns out that when using Canright's representation, the only non-linear function is the multiplication in GF(2^2). An example for how to share this function using only three shares has been presented by Nikova *et al.* in [24]. Building on these findings, we applied the countermeasure to our unprotected AES implementation. For this architecture we conducted a complete side-channel evaluation based on real-world power traces that we obtain from SASEBO. We use a variety of different power analysis attacks to investigate the achieved level of resistance of our implementation against first order DPA attacks even if an attacker is capable of measuring 100 million power traces.

1.3 Outline

We first give a brief introduction to *Differential Power Analysis* and countermeasures in the following Section. A general overview follows a more detailed description of the masking scheme presented in [23,24], which we use for our experimental evaluation. Subsequently in Section 3 AES and Canright's optimized

S-box are briefly recalled, before we describe a shared AES S-box. Based on these findings, in Section 4 we propose two hardware architectures – unprotected and protected – of AES-128 and mount DPA attacks on its real-world power traces in Section 5. Finally we conclude this article in Section 6.

2 Introduction to DPA

Smart cards and other types of pervasive devices performing cryptographic operations are seriously challenged by side-channel cryptanalysis. Several publications, e.g., [12] have stressed that such physical attacks are an extremely practical and powerful tool for recovering the secrets of unprotected cryptographic devices. In fact, these attacks exploit the information leaking through physical side channels and involved in sensitive computations to reveal the key materials.

Amongst the known sources of side channels and the corresponding attacks most notable are power analysis attacks [18]. Many different kinds of power analysis attacks, e.g., simple and differential power analysis (SPA and DPA) [18], template-based attacks [2], and mutual information analysis [14], have been introduced while each one has its own advantages and is suitable in its special conditions. However, correlation power analysis (CPA) [6], which is a general form of DPA, got more attention since it is able to efficiently reveal the secrets by comparing the measurements to the estimations obtained by means of a theoretical power model which fits to the characteristics of the target implementation.

2.1 Countermeasures

Generally speaking, the goal of a DPA countermeasure is to prevent a dependency between the power consumption of a cryptographic device and characteristics of the executed algorithm, e.g., intermediate values, executed operations, and taken branches [19]. Amongst the countermeasures proposed at different levels of design and abstraction *Masking* methods, which rely on randomizing key-dependent intermediate values processed during the execution of the cipher, are widely applied on either the algorithmic level [26] or the cell level [27]. An n-order masking technique is in fact an $(n+1, n+1)$ secret sharing scheme [3,31], where all shares of the secret are required to proceed.

When an algorithmic masking scheme is applied on a microprocessor-based platform, it is often combined by *shuffling* [16] which randomizes the order of operations. Applying a masking scheme in a software implementation (microprocessor) can be defeated by higher order attacks [11,34]. However, practical experiences like [20] showed that still there is a first-order leakage when hardware (ASIC or FPGA) is protected by a masking scheme at algorithm level. This leakage can be exploited by sophisticated power models, e.g., toggle-count model, or by a template-based DPA attack.

In short, currently there exists no perfect protection against DPA attacks. However, applying appropriate countermeasures makes the attacker's task more

difficult and expensive. Chari *et al.* have shown in [10] that up to n-th order DPA attacks can be prevented by using n masks. Following this direction, Nikova *et al.* extended the idea of masking with more than two shares in [23] to prevent those attacks which use sophisticated power models, e.g., counting the glitches occurring when the inputs of a complex combinational circuit change. They showed that non-linear functions implemented in such a way, achieve provable security against first-order DPA attacks and also resist higher-order attacks that are based on a comparison of mean power consumption. Estimations of a hardware implementation of these ideas are presented in [24] where an S-box of the Noekeon cipher [17] is considered as a case study without practical evaluation of its resistance to DPA attacks. Afterwards, the same approach is applied on the S-box of the PRESENT cipher [5], and its resistance against first-order attacks is verified in [28]. Since it seems to be a promising candidate for a lightweight and side-channel resistant implementation, we have chosen this scheme to implement the AES S-box and have a comparison (on its first-order leakage) to the masked AES S-boxes proposed so far, e.g., [9] and [26].

3 Shared Computation of the AES S-Box Using Composite Fields

In this section first an algorithmic description of AES is given, before the AES S-box as described by Canright is expressed. Finally, the threshold countermeasure of Nikova *et al.* is applied to the Canright AES S-box that will be used in the next section for a protected implementation of the AES.

3.1 Algorithmic Description of AES

In November 2001 the Rijndael algorithm was chosen as the *Advanced Encryption Standard (AES)* by the National Institute of Standards and Technology (NIST) [22]. AES is a symmetric block cipher, that processes data blocks of 128 bits. Three different key lengths are specified: 128, 192, and 256 bits, resulting in 10, 12 or 14 rounds, respectively. AES is, depending on the key length, also referred to as AES-128, AES-192, and AES-256 and in the remainder of this article we focus on the encryption process of AES-128.

At the beginning of the algorithm, the input is copied into the *State* array, which consists of 16 bytes, arranged in four rows and four columns (4×4 - Matrix). At the end, the *State* array is copied to the output.

The bytes of the *State* are interpreted as coefficients of a polynomial representation of finite field elements in $GF(2^8)$. All byte values in the remainder of this article will be written in hexadecimal notation in the form {ab}. In encryption mode, the initial key is added to the input value at the very beginning, which is called an initial round. This is followed by 9 iterations of a normal round and ends with a slightly modified final round. During one normal round the following operations are performed in the following order: *SubBytes, ShiftRows, MixColumns*, and *AddRoundkey*. The final round is a normal round without the *MixColumns* stage.

SubBytes is a non-linear, invertible byte substitution and consists of two transformations that are performed on each of the bytes independently: First each byte is substituted by its multiplicative inverse in $GF(2^8)$ (if existent), element $\{00\}$ is mapped to itself. Then the following affine transformation over $GF(2)$ is applied: $b_i^{'} = b_i \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$ for $0 \leq i \leq 8$, where $b_i(c_i)$ is the i-th bit of the byte $b(c)$, $c = \{63\} = 01100011_2$.

ShiftRows cyclically shifts each row of the *State* by a certain offset. The first row is not shifted at all, the second row is shifted by one, the third row by two, and the fourth row by three bytes to the left.

MixColumns processes one column of the *State* at a time. The bytes are interpreted as coefficients of a four-term polynomial over $GF(2^4)$. Each column is multiplied modulo $x^4 + 1$ with a fixed polynomial $\mathrm{a}(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. This can be written as the following matrix multiplication, where $s'(x) = a(x) \otimes s(x)$:

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02\ 03\ 01\ 01 \\ 01\ 02\ 03\ 01 \\ 01\ 01\ 02\ 03 \\ 03\ 01\ 01\ 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix} \text{for } 0 \leq c \leq 3.$$

AddRoundKey adds the 128-bit *round key* generated from *KeyExpansion* to the 128-bit *State*. It is a simple XOR-addition of the *round key* and the *State*.

KeyExpansion derives 10 round keys from the initial key iteratively. The key is grouped into four words w_0, w_1, w_2, and w_3, that consist of four bytes each. w_3 is cyclically shifted to the left by one byte. The result is bytewise substituted by the S-box and then a round constant *RCon* is XOR-added. Finally the result is XOR added to w_0 yielding w'_0. w'_1 is obtained by XOR adding w'_0 with w_1, $w'_2 = w'_1 \oplus w_2$ and $w'_3 = w'_2 \oplus w_3$. The new key state or round key RK_i is then formed by $RK_i = w'_0|w'_1|w'_2|w'_3$. The round constants $RCon_i$ are derived by the following equation: $RCon_i = x^i \bmod \mathrm{m}(x)$, where i denotes the round number, $0 \leq i \leq 9$ and the irreducible polynomial $\mathrm{m}(x) = x^8 + x^4 + x^3 + x + 1$. For further details on AES, the interested reader is referred to [29].

3.2 Canright's Representation of the AES S-Box

Canright investigated the hardware requirements of the AES S-box very thoroughly in [8]. He proposed a very compact S-box that is composed of smaller fields. As one can see from Fig. 1 the input to the S-box is transformed by a linear mapping that changes the basis from GF(2^8) to GF(2^8)/GF(2^4)/GF(2^2) (please ignore pipelining and register remarks in this step, these issues are addressed in Section 3.3 and in Section 5). The output is transformed by a linear mapping that combines the basis change back to GF(2^8) and the inverse mapping of the AES S-box. Beside two 4-bit XORs, a GF(2^4) inverter (center module), a GF(2^4) square-scaler (top left module) and three instances of a GF(2^4) multiplier (right and bottom left) are required. The GF(2^4) square-scaler uses a normal basis (Γ^4, Γ) and only consists of wiring and three XOR gates. The GF(2^4) inverter uses a normal basis (Γ^4, Γ) and consists of 5 XOR gates, some wiring and three

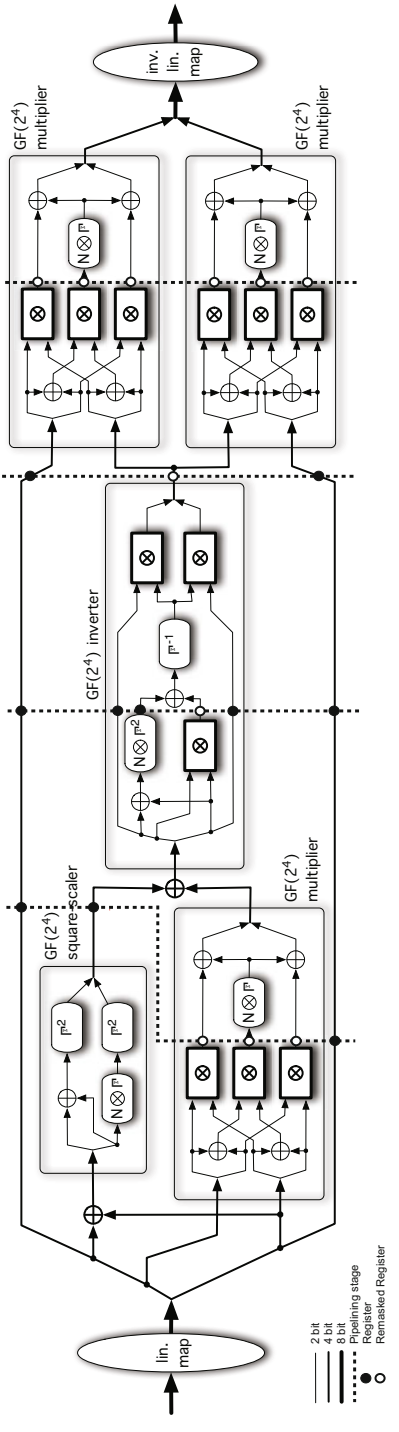


Fig. 1. Composite field representation of the AES S-box, as described in [8]. The thick lined rectangles are multipliers in GF(2^2), the only non-linear parts.

instances of a GF(2^2) multiplier (thick lined rectangles)[1]. The GF(2^4) multiplier consists of nine XOR gates, some wiring and three parallel instances of a GF(2^2) multiplier.

3.3 A Shared AES S-Box

To apply the threshold countermeasure of Nikova *et al.* [24] we need to share the non-linear functions of the algorithms, while the linear functions are simply implemented s times in parallel, where s denotes the amount of shares. Particularly interesting are realizations with minimal amount of shares, *i.e.*, $s = 3$, because they require the fewest hardware resources. Having a closer look on the representation of Canright, it turns out that the only non-linear parts of the AES S-box are the multipliers in GF(2^2). In [24] an exemplary realization of this multiplier using only three shares has been presented. It is noteworthy to point out that the threshold countermeasure requires registers between different stages of shared functions. As can be seen from Fig. 1, Canright's S-box representation requires in total five pipelining stages. Note that not only the output of the shared functions, but all signals have to be pipelined. This implies that in total we need to store 174 bits, which as we will see in Section 4 will increase the area requirements even further (please ignore remasked register remarks in this step, this issue is discussed in Section 5).

4 Hardware Architectures

This section is dedicated to the description of the different hardware profiles that we will attack in the next section. For this purpose we first introduce the design flow used before we detail the hardware architectures, and finally summarize the implementation results.

4.1 Design Flow

We used *Mentor Graphics ModelSimXE 6.4b* and *Synopsys DesignCompiler* version *A-2007.12-SP1* for functional simulation and synthesis of the designs to the *Virtual Silicon* (VST) standard cell library *UMCL18G212T3* [33], which is based on the *UMC L180 0.18µm 1P6M* logic process with a typical voltage of 1.8 V. We used *Synopsys Power Compiler* version *A-2007.12-SP1* to estimate the power consumption of our ASIC implementations. For synthesis and for power estimation we advised the compiler to keep the hierarchy and use a clock frequency of 100 KHz, which is a widely used operating frequency for RFID applications. Note that the wire-load model used, though it is the smallest available for this library, still simulates the typical wire-load of a circuit with a size of around 10 000 GE.

To substantiate our claims on the efficacy of the proposed countermeasures, we implemented the ASIC cores on SASEBO to obtain and evaluate real-world

[1] Note that the inverse in GF(2^2) only consists of some wiring.

power traces. For design synthesis, implementation and configuration of SASEBO we used *Xilinx ISE v10.1.03 WebPACK*. In a typical application scenario the cryptographic core would be part of an integrated ASIC, hence for the power measurements on SASEBO we embedded the cryptographic core in a framework that handles the communication between the two FPGAs.

4.2 A Very Compact Implementation of AES

The most area consumption typically occurs for storing the intermediate state, because typically flip-flops are used, which have high area requirements. In the technology we used, a single-input, positive edge triggered D flip-flop requires 5 GE and can store 1 bit. If you have more than one input, e.g. the output from *SubBytes*, the ouput from *ShiftRows* and the output from *MixColumns*, you need multiplexers. A Multiplexer for a selection from two inputs to one output (2-to-1 MUX) costs 2.33 GE per bit. Scan flip-flops combine a D flip-flop and a 2-to-1 MUX for 6 GE per bit. That is a saving of 1.33 GE per bit of storage. For the AES this sums up to 340 GE. Scan flip flops have been used before, e.g. in implementations of PRESENT [30] and KATAN/KTANTAN [7].

Based on the properties of scan flip-flops (2 inputs "for free"), we designed the architecture for our tinyAES implementation. As can be seen in Fig. 3, both the `State` array and the `Key` array each consist of a 16 stage 8-bit width shift register. Each of the stages comprises 8 scan flip-flops (cells 00 to 33) with two inputs. One input receives the output of the previous stage, while the other one contains the result of *ShiftRows*, which comes for free in our design, since

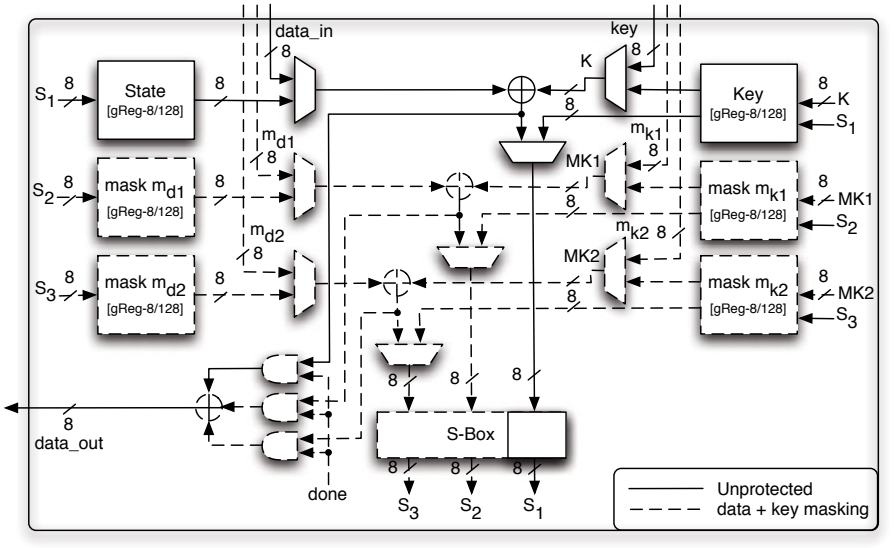
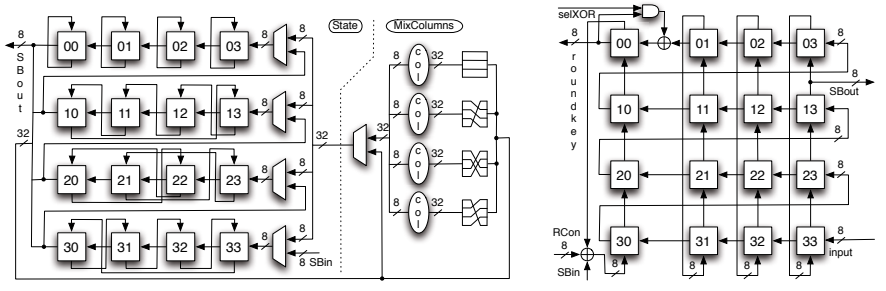


Fig. 2. Hardware architectures of both implementations of a serialized AES-128 encryption-only core

shifting is done by wiring. Instead of adding one 2-to-1 MUX for every cell of the `State` array, we designed our architecture in a way that we only need one additional MUX for every row. These are the 4 2-to-1 MUXes (each 8-bit width) on the right hand side of the cells (03) to (33), accounting for 75 GE instead of 300 GE. This choice is strongly related to the choice of parallelism of the *MixColumns* operation. Both [13] and [15] implemented *MixColumns* in a serialized way, that is, it takes 4 clock cycles to calculate one column. We opted to implement *MixColumns* not in a serialized way, because, as we are going to show below, the hidden overhead is larger than the potential savings.



(a) `State` array with *ShiftRows* and *MixColumns* (b) `Key` array

Fig. 3. Architectures of storage modules for the *State* and the *Key* arrays

The `Key` array consists of a similar 128 flip-flop array as the `State` array, but the wiring between the registers is different. There are two shifting directions: horizontal and vertical. The current 8-bit chunk of the round key is output during the horizontal shifting, while the S-box look-up for the key schedule is performed during vertical shifting. Note that the *RotWord* operation is implemented by taking the output of the (13) cell instead of the (03) cell as the input for the S-box look-up. The S-box output is XORed to the round constant *RCon* and the output of the (00) cell. Once all four S-box look ups have been performed the first column of the key state contains already the new roundkey, but the other three columns do not. The remaining steps of the key update is performed during the output of the round key chunk by XORing the output of cell (00) to the output of cell (01) as the new input of cell (00). Once the whole row is output, *i.e.*, every fourth clock cycle, the feedback XOR is not required, and thus the output of cell (00) is gated with an AND gate. Note that on top of the cost for storage (768 GE) and the calculation and storage of the round constant (89 GE), in our implementation the whole key schedule requires only one 8-bit AND gate (11 GE), an 8-bit XOR gate with two inputs (19 GE) and an 8-bit XOR gate with three inputs (35 GE). We believe that our results are very close to a theoretical optimum. This is reflected in the area savings compared to previous results: 924 GE[2] vs. 1076 in [15]. [13] uses a RAM-like storage, which includes

[2] The key expansion unit of [15] also contains the Round Counter generation, thus in order to have a fair comparison, we have to add the area for both: 835+89=924 GE.

both, the *State* and the *Key* arrays. Thus for a fair comparison we have to add both modules together: 1678 GE vs. 2040 GE in [13].

In our architecture, *MixColumns* is realized by four instances of a module called *col*, which outputs the result of the first row of the *MixColumns* matrix. Since the matrix used is circulant, one can use the same module and just rotate the input accordingly. Note that in hardware rotation can be realized by simple wiring and comes nearly for free. By serializing *MixColumns*, one can save 75% of the area (280 GE). Also, 3 of the 4 MUXes on the right hand side of every row can be discarded, and the 32-bit width 2-to-1 MUX (75 GE) at the right hand side of the dashed line in Fig. 3 could be shrinked to an 8-bit width 2-to-1 MUX (19 GE), leading to savings of 112 GE. So in total, the potential savings for the whole design (not only MixColumns) are 392 GE. However, one needs to temporarily store at least 3 of the output bytes, because we cannot over-write the input bytes, before all four output bytes are calculated. That is a storage overhead of $5 \times 24 = 120$ GE. Since the *MixColumns* matrix is circulant, we need to rotate the input to the *col* module with a different offset for every output byte. This can be implemented by simple wiring (see the right hand side of *col* in Fig. 3), followed by a 32-bit width 4-to-1 MUX (192 GE) to select the correct input. In summary, the potential savings are in this case reduced to 80 GE, while at the same time one needs far more complex control logic to orchestrate the control signals for the MUXes and the additional temporary storage flip-flops (see below).

Instead of using a Finite State Machine (FSM), we rather spent considerable amount of time and effort to decrease the area requirements for the control logic for the unprotected version (*Profile 1*). The control signals are derived from a 5-bit LFSR with taps at bit position 1 and 5 that has a cycle length of 21. This is exactly the amount of cycles required to perform one round of AES and the key schedule: 16 cycles for *AddRoundKey*, 1 for *ShiftRows* (during which the `Key` state is not clocked) and 4 for the parallel execution of *MixColumns* and *SubWord*. Every time a cycle is completed a pulse is generated that is used to control the MUXes and the clock gating logic. Simple Boolean logic is used to derive all control signals from this pulse, such that in total only 73 GE are required for the control logic. In [15] no details about the control logic are given, and 220 GE are required for both control logic and *"others"*. Thus a fairer comparison is 80 GE vs. 220 GE. As a consequence of a very serialized implementation, a RAM-like storage, and usage of an FSM, [13] requires 400 GE for control logic (including the round constant generation) compared to 162 GE for our implementation. Similar to [15], we used Canright's description of the AES S-box [8], which is the smallest known.

Our envisioned target application is a very constrained device, e.g. a low-cost passive RFID-tag or similar. By re-ordering the input and output bytes, it is possible to reduce the area significantly, to be precise by 13.5%. As a consequence, our implementation requires an input and output ordering that is row-wise, *i.e.*, $S_{00}|S_{01}|S_{02}|S_{03}|S_{10} \ldots S_{32}|S_{33}$ and not column-wise ($S_{00}|S_{10}|S_{20}|S_{30}|S_{01} \ldots S_{23}|S_{33}$), where S_{ij} denotes one byte of the input/output with $0 \leq i, j \leq 3$.

If column-wise ordering is needed, 20 additional 8-bit wide 2-to-1 MUXes are required (373 GE). In fact with our approach we forward the effort of re-ordering the bytes to the other communication party. In an RFID scenario this will most likely be a reader or a database server, which is by far not as constrained as a passive RFID tag. Hence, the costs for the byte re-ordering are marginal. Furthermore, when two devices with our AES implementation communicate, no byte re-ordering is needed at all. We believe that this re-ordering does not pose a severe problem in practice, while at the same time results in an attractive area saving.

4.3 A Threshold Implementation of AES

If we share both the data path and the key schedule we obtain the threshold version (*profile 2*). The additional hardware requirements for this profile are depicted in Fig. 2 by the dashed lines. For this profile we need four randomly generated masks (m_{d1}, m_{d2}, m_{k1}, m_{k2}), which are XORed to the data chunk and the key chunk. The unmasking step is performed by simply XORing all three shares yielding the output (`data_out`). The state of the masks also needs to be maintained, which leads to two more instantiations of both the `State` and the `Key` module (`mask` m_{d1}, `mask` m_{d2}, `mask` m_{k1} and `mask` m_{k2}). Furthermore, the S-box is now replaced by a shared S-box module that contains five pipelining stages (see Fig. 1). This delays the computation of the round keys and, as a consequence, the pipeline needs to be emptied in every encryption round. Thus *profile 2* needs 25 clock cycles for one round and uses a small FSM to derive the control signal (77 GE).

4.4 Performance Figures

Table 1 summarizes the implementation figures of both profiles. The upper part gives a detailed breakdown of the area requirements both in absolute and relative values. The lower part lists the smallest achievable area requirements, power estimations, clock cycles, and throughput at 100 KHz.

Profile 1 (unprotected) has an area footprint of 2400 GE of which 70% are required to store the key and the data state. MixColumns and S-box are the other two main contributors to the area requirements. *Profile 2* (threshold version) increases the area demands more than four-fold to 10793 GE. The main reason for this is the S-box, which increases more than 10 fold and now occupies a whopping 35% of the area. This increment mainly comes from the 13-fold increment of the GF(2^2) multiplier (13 GE vs. 173 GE) and the four pipelining stages that need to store an additional 174 bits (870 GE).

Profile 1 requires 21 clock cycles per round and 16 clock cycles to output the result (226 clock cycles in total). *Profile 2* needs 4 additional clock cycles per round, due to the pipelining stages in the S-box, which leads to a total of 266 clock cycles (18% increment). Please note that the time required can be reduced by 16 clock cycles for additional 21 GE for *profile 1* and 64 GE for *profile 2* by adding another XOR gate for the final *KeyAdd* allowing to interleave consecutive

Table 1. Breakdown of the post-synthesis implementation results for both architectures of a serialized AES-128 encryption-only core

Goal	AES-128		*Profile 1* (unprotected)		*Profile 2* (threshold version)	
			%	GE	%	GE
Area	sequential:	Round constant	3	89	0.5	89
		State array	32	843	8	843
		Key array	32	835	8	842
		m_{d1} array			8	843
		m_{d2} array			8	843
		m_{k1} array			8	842
		m_{k2} array			8	842
	combinational:	MUXes	5	128	3	376
		KeyAdd	1	21	0.5	64
		S-box	9	233	35	4071/4244*
		MixCol	14	373	10	1120
		control	3	72	0.5	77
		other	1	7	0.5	89
	compile simple	sum	100	2601	100	10941/11114*
	compile ultra	sum	2400 GE		10793/11031* GE	
	cycles		226 clk		266 clk	
	power @100 KHz		3.7 μA		13.4 μA	
	throughput @100 KHz		57 Kbps.		48 Kbps.	
Area and Speed	*compile ultra*	sum	2421 GE			
	cycles		210 clk			
	power @100 KHz		3.7 μA			
	throughput @100 KHz		61 Kbps.			

* Using remasked registers excluding PRNGs (explained in Section 5)

message blocks. The power consumption was estimated at 100 KHz and a supply voltage of 1.8V. The unprotected implementation (*profile 1*) requires 3.7 μA and thus is suitable for passive RFID-tags. For *profile 2*, however, this figure increases more than threefold to 13.4 μA, which might already decrease the reading range of a passive RFID tag. If required, power saving techniques might be applied to reduce the power consumption at the cost of additional area. Please note that power figures for different standard-cell libraries cannot be compared in a fair manner. Furthermore, power estimates vary greatly depending on the simulation method used and effort spent. Therefore we did compare our power figures with previous works.

5 Experimental Results

In addition to the performance and area consumption features of our threshold implementation, we have implemented the whole AES encryption design on an FPGA-based platform and analyzed the actual power consumption traces to practically investigate its resistance to first-order DPA attacks. Later in this

section the platform used and the measurement setup are introduced, then practical results are shown to validate the desired security levels.

5.1 Measurement Setup

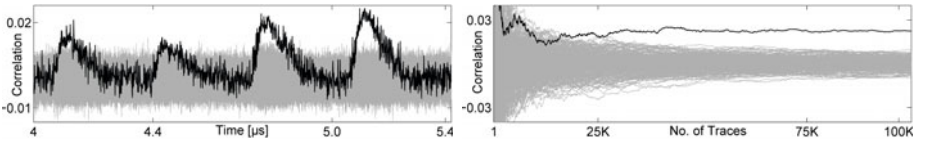
A SASEBO (Side-channel Attack Standard Evaluation Board) which is particularly designed for side-channel attack experiments [1] has been used as the measurement platform. It contains an xc2vp7 Virtex-II Pro FPGA [35] as the crypto FPGA, clocked at a frequency of 3MHz[3], to implement the design. A LeCroy WP715Zi 1.5GHz oscilloscope at a sampling rate of 1GS/s and a differential probe which captures voltage drop of a 1Ω resistor at VDD (1.8V) path are used as the measurement equipments to collect the power traces.

5.2 Side-Channel Resistance

In order to find the leakage points and have a reference to fairly judge about the power analysis resistance of our implementation, we have switched off the mask generators and kept all masks as zero to prevent randomization by masking. 100 000 traces are collected from this implementation while encrypting random plaintexts. As expected and also observed in [20], CPA attacks which use a HW model predicting the S-box input or output are not able to recover the secrets of hardware implementations. What should directly lead to a successful attack is a CPA using HD model which predicts bit flips on a part of the state register when S-box outputs are overwritten to each other. Therefore, two consecutive key bytes, *i.e.*, 2^{16} hypotheses, should be guessed. The results of such an attack, which shows the amount of information leakage related to register updates, is depicted by Fig. 4(a). Note that to reduce the attack complexity we have given a favor to the attacker by knowing a key byte and reducing the key hypotheses to 2^8. As shown in Fig. 4(b), around 30 000 traces are sufficient to perform a successful attack. Because of the pipeline architecture of the S-box the correct key guess appears at more than one clock cycle in the attack results. Also, a mutual information analysis attack using the same distinguisher, *i.e.*, HD of the register updates, is efficiently capable of recovering the secret. The results of this attack are shown in Fig. 5(a) and Fig. 5(b). It is noteworthy to mention that those four clock cycles in which the secret leaks clearly in both Fig. 4 and Fig. 5 are when the intermediate results of the target S-box computation are consecutively stored in the pipeline registers of the shared S-box.

In order to observe the combinational circuit leakage a correlation-enhanced collision attack, presented in [21], is mounted by getting average over the acquired traces based on the plaintext bytes, and correlating the mean traces after alignment based on the clock cycles when the target S-boxes are computed. In fact, this attack is very similar to a template-based DPA attack using only the mean vectors of the templates and avoiding the profiling step. The result of this

[3] This frequency of operation is selected to prevent overlapping power peaks of consecutive clock cycles and hence to simplify the attacks.



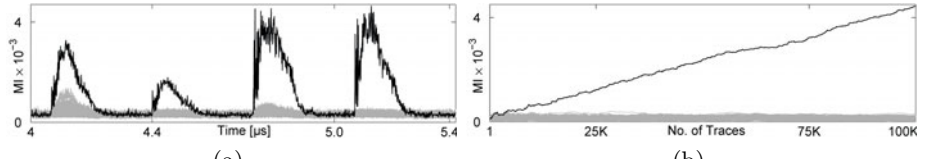
(a) (b)

Fig. 4. CPA attack results when the mask generators are off by means of a HD model (a) using 100K measurements and (b) at point 5.1μs over the number of traces

attack presented in Fig. 6 shows that the leakage of the combinational circuit, *i.e.*, the S-box instance, also leads to successfully revealing the linear difference between two key bytes.

In the second step we have measured 5 million traces while the random number generators are turned on and work normally. The plaintext bytes are randomly selected, and the masks are shared neither between the plaintext and key bytes nor between computation rounds of encryptions. In short, there is no mask reuse in our target design. All attacks, mounted on the first step when the random number generators were off, are repeated on the new measurements. The CPA attack using HD, whose result is shown in Fig. 7(a), is expectedly not successful since registers are masked by means of three shares and the predicted HD does not fit to the register updates. However, the registers which contain the shares are updated at the same time, and their information leakages through power consumption are inherently summed up. As observed in [32] the sum of shared registers leakages is not independent of the actual (unshared) value, and a mutual information analysis is expected to recover the secret. We have repeated the last mutual information analysis attack by means of a HD model as the distinguisher. The corresponding attack result is shown in Fig. 7(b), but it still cannot distinguish the correct hypothesis. This might be related to the number of traces; in other words, 5 million traces seem to be not enough due to the amount of switching and electronic noise in our platform. However, the same issue has been addressed in [25], where it is argued that the combinational functions following the registers change the distribution of shared register leakages leading to failed mutual information analysis attacks.

On the other hand, repeating the last correlation collision attack, whose results are given in Fig. 7(c) and Fig. 7(d), led to revealing the secret using around



(a) (b)

Fig. 5. MIA attack results when the mask generators are off by means of a HD model (a) using 100K measurements and (b) at point 5.1μs over the number of traces

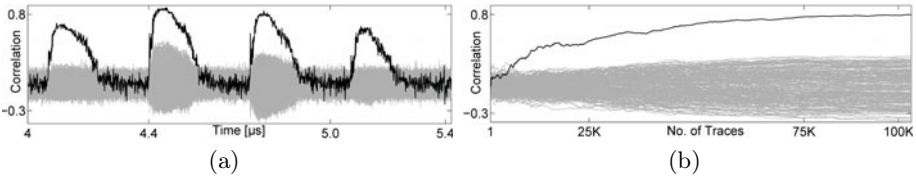


Fig. 6. Correlation collision attack results when the mask generators are off (a) using 100K measurements and (b) at point 4.8μs over the number of traces

3.5 million traces. Since this attack recovers the first-order leakage of combinational circuits, it shows that our shared S-box still has first-order leakage. During the investigation of this issue (as also addressed in [25]) we have realized that the values which are saved in the intermediate registers of our shared S-box are not uniformly distributed. This means, *property 3* illustrated in [23] and [25] does not hold although we have used the shared multiplication in GF(2^2) proposed by the original authors. The problem arises when the output of the shared multiplication modules which have some shared inputs are mixed by means of the linear functions. In fact, the correction terms which have been added to the shared multiplications to provide uniformity are canceled out. It is actually a practical evidence showing that if the uniformity property does not hold, the leakage of the combinational circuit caused by the glitches leads to a recoverable first-order leakage. Since searching through all possible correction terms and their combination to check whether they lead to a uniform distribution in our design was a very time consuming task, we could neither check all possible cases nor could we find a suitable case. Instead, (as also addressed in [25]) we have tried to use random fresh masks inside each pipeline stage when required. The scheme we have used to add fresh masks, so-called *remasking*, is shown by Fig. 8. We have simulated our shared S-box and tried to find the minimum cases

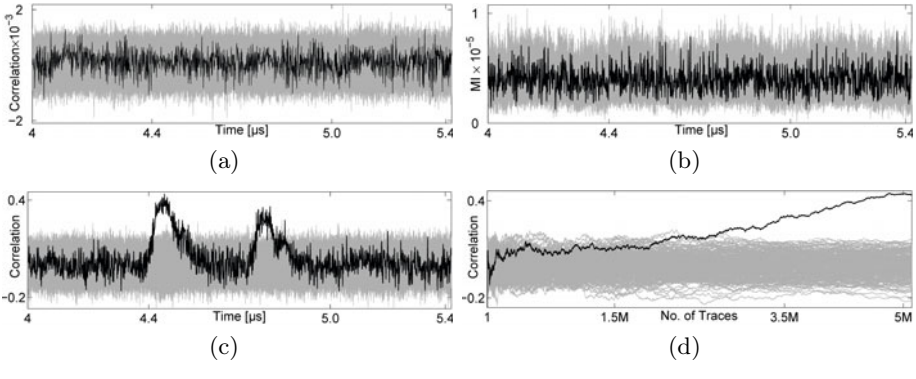


Fig. 7. Attack results when the mask generators are working using 5 million traces (a) CPA using a HD model, (b) MIA using a HD model, (c) correlation collision attack, and (d) correlation collision attack at point 4.45μs over the number of traces

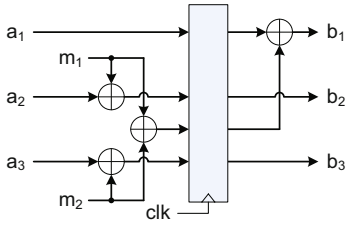


Fig. 8. Remasking scheme for a 3-share case

where remasking is required, and finally yielded the design shown in Fig. 1; the remasked registers are marked by ◯.

Finally 100 million traces have been acquired from the last design when all random number generators worked normally and the plaintext bytes were randomly selected. It should be noted that the fresh masks for the remasked registers are provided by means of LSFRs which have enough period considering 100 million measurements. All the attacks illustrated have been repeated here on all measured traces. A CPA and an MIA using a HD model on S-box outputs are still not applicable; their results are depicted in Fig. 9(a) and Fig. 9(b) respectively. Also, we have performed a third-order CPA attack by cubing the power traces and correlating the results to predictions of a HD model in order to recover the

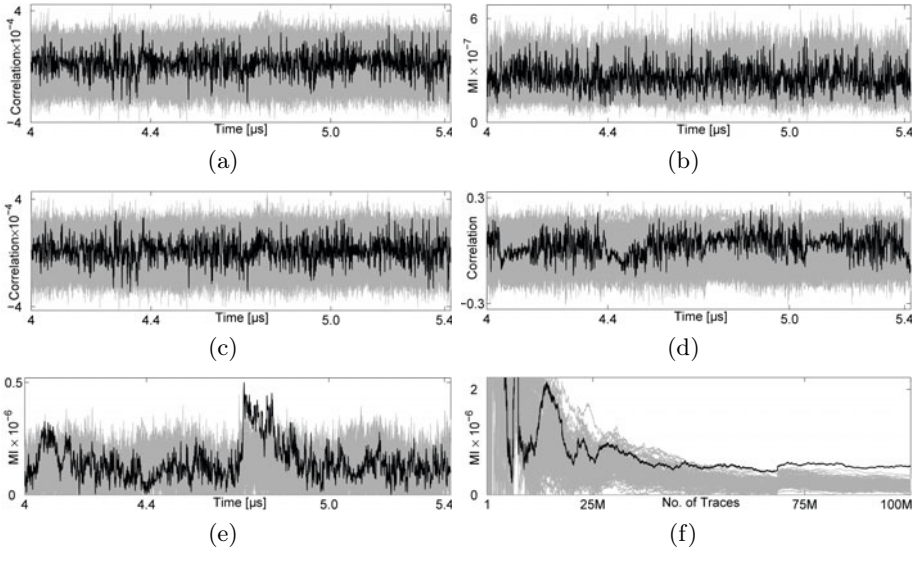


Fig. 9. Attack results when the mask generators are working and the remasked registers are applied using 100 million traces (a) CPA and (b) MIA and (c) third-order CPA using a HD model on S-box outputs, (d) correlation collision attack, (e) MIA using a HD model on S-box input, and (f) MIA at point 4.7μs over the number of traces

leakage of the inherently summed shared register updates. The result of this attack shown in Fig. 9(c) indicates that 100 million traces are still not enough for such a higher-order attack. The correlation collision attack is also not applicable. Its results are shown in Fig. 9(d). This means that our target design could prevent the first-order leakage under Gaussian assumption since correlation collision attack applies only the mean traces[4]. This confirms the statement given in [25] that the average power leakage of a threshold implementation should be independent of the processed values.

We examined several models and performed a couple of mutual information attacks, and finally could make the secret distinguishable using HD of the S-box input. Using this model, similar to correlation collision attacks, the linear difference between two key bytes can be recovered. The result of this attack is shown by Fig. 9(e) and Fig. 9(f), and indicates that the secret gets distinguishable using more than 80 million traces.

6 Conclusions

While implementations of cryptographic algorithms in pervasive devices seriously face area and power constraints, their resistance against physical attacks has to be taken into account. Unfortunately, nearly all side-channel countermeasures introduce power and area overheads which are proportional to the values of the unprotected implementation. Therefore, this fact prohibits the implementation of a wide range of proposed countermeasures and also limits possible cipher candidates for ubiquitous computing applications.

Most of the countermeasures proposed for implementing a side-channel resistant AES in hardware remained unfortunately with a first-order leakage. In this article we have applied a recently proposed secret sharing-based masking scheme to the AES S-box in order to improve the first-order resistance. Decomposition of the AES S-box into a series of S-boxes of algebraic degree two and splitting them into (at least) three shares is a challenging task. However, we have used the architecture of the smallest AES S-box and have shared the non-linear operation which is a $GF(2^2)$ multiplier. To separate the glitches of different parts of the circuit we have designed the S-box in five pipeline stages by adding four sets of intermediate registers and applying a remasking scheme on some selected registers.

Our proposed hardware architecture for the AES reduces the area requirements to only 2400 GE, which is 23% smaller than the smallest previously published. After the secret sharing based countermeasure has been applied, the area requirements are 11031 GE, while the timing overhead compared to our unprotected implementation with a similar architecture is only 18%. According to practical side-channel investigations, masking the state and the key registers by means of two shares each could improve the resistance against the considered (most well-known) first-order DPA attacks. Our protected implementation offers

[4] In fact, we continued the measurements till 400 million, and still this type of attack was not feasible.

128-bit standardized security with improved side-channel resistance for around 11 000 GE.

Acknowledgment

The authors would like to thank Akashi Satoh and Research Center for Information Security (RCIS) of Japan for the prompt and kind help in obtaining SASEBOs, and François-Xavier Standaert for his fruitful and helpful comments and suggestions.

References

1. Side-channel attack standard evaluation board (sasebo), Further information are http://www.rcis.aist.go.jp/special/SASEBO/index-en.html
2. Agrawal, D., Rao, J.R., Rohatgi, P.: Multi-channel Attacks. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 2–16. Springer, Heidelberg (2003)
3. Blakley, G.R.: Safeguarding Cryptographic Keys. In: National Computer Conference, pp. 313–317 (1979)
4. Blömer, J., Guajardo, J., Krummel, V.: Provably Secure Masking of AES. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 69–83. Springer, Heidelberg (2004)
5. Bogdanov, A., Leander, G., Knudsen, L., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Brier, E., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
7. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — A Family of Small and Efficient Hardware-Oriented Block Ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
8. Canright, D.: A Very Compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
9. Canright, D., Batina, L.: A Very Compact "Perfectly Masked" S-Box for AES. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 446–459. Springer, Heidelberg (2008), the corrected version is available at Cryptology ePrint Archive, Report 2009/011 http://eprint.iacr.org/2009/011
10. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards Sound Approaches to Counteract Power-Analysis Attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
11. Coron, J.-S., Prouff, E., Rivain, M.: Side Channel Cryptanalysis of a Higher Order Masking Scheme. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 28–44. Springer, Heidelberg (2007)
12. 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)

13. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES Implementation on a Grain of Sand. IEE Proceedings of Information Security 152(1), 13–20 (2005)
14. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
15. Hämäläinen, P., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In: DSD, pp. 577–583 (2006)
16. Herbst, C., Oswald, E., Mangard, S.: An AES Smart Card Implementation Resistant to Power Analysis Attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 239–252. Springer, Heidelberg (2006)
17. Daemen, G.J., Peeters, M., Rijmen, V.: The Noekeon Block Cipher. In: First Open NESSIE Workshop (2000)
18. 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)
19. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
20. Mangard, S., Pramstaller, N., Oswald, E.: Successfully Attacking Masked AES Hardware Implementations. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 157–171. Springer, Heidelberg (2005)
21. Moradi, A., Mischke, O., Eisenbarth, T.: Correlation-Enhanced Power Analysis Collision Attack. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 125–139. Springer, Heidelberg (2010)
22. National Institute of Standards and Technology (NIST). Announcing the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197 (November 2001)
23. Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 529–545. Springer, Heidelberg (2006)
24. Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 218–234. Springer, Heidelberg (2009)
25. Nikova, S., Rijmen, V., Schläffer, M.: Secure Hardware Implementation of Nonlinear Functions in the Presence of Glitches. Journal of Cryptology (2010) (in press), doi:10.1007/s00145-010-9085-7
26. Oswald, E., Mangard, S., Pramstaller, N., Rijmen, V.: A Side-Channel Analysis Resistant Description of the AES S-Box. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 413–423. Springer, Heidelberg (2005)
27. Popp, T., Mangard, S.: Masked Dual-Rail Pre-charge Logic: DPA-Resistance Without Routing Constraints. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 172–186. Springer, Heidelberg (2005)
28. Poschmann, A., Moradi, A., Khoo, K., Lim, C.-W., Wang, H., Ling, S.: Side-Channel Resistant Crypto for less than 2,300 GE. Journal of Cryptology (2010) (in press), doi: 10.1007/s00145-010-9086-6
29. Rijmen, V., Daemen, J.: The Design of Rijndael: AES. The Advanced Encryption Standard, 1st edn. Springer, Heidelberg (2002)
30. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-Lightweight Implementations for Smart Devices – Security for 1000 Gate Equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)

31. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
32. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010)
33. Virtual Silicon Inc. 0.18 μm VIP Standard Cell Library Tape Out Ready, Part Number: UMCL18G212T3, Process: UMC Logic 0.18 μm Generic II Technology: 0.18μm (July 2004)
34. Waddle, J., Wagner, D.: Towards Efficient Second-Order Power Analysis. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 1–15. Springer, Heidelberg (2004)
35. Xilinx: Virtex-II Pro and Virtex-II ProX Platform FPGAs: Complete Data Sheet (November 2007), http://www.xilinx.com/support/documentation/data_sheets/ds083.pdf

Fully Leakage-Resilient Signatures

Elette Boyle[1,⋆], Gil Segev[2,⋆⋆], and Daniel Wichs[3,⋆⋆⋆]

[1] Massachusetts Institute of Technology, Cambridge, MA 02139, USA
eboyle@mit.edu
[2] Microsoft Research, Mountain View, CA 94043, USA
gil.segev@microsoft.com
[3] New York University, New York, NY 10012, USA
wichs@cs.nyu.edu

Abstract. A signature scheme is *fully leakage resilient* (Katz and Vaikuntanathan, ASIACRYPT '09) if it is existentially unforgeable under an adaptive chosen-message attack even in a setting where an adversary may obtain bounded (yet arbitrary) leakage information on *all intermediate values that are used throughout the lifetime of the system*. This is a strong and meaningful notion of security that captures a wide range of side-channel attacks.

One of the main challenges in constructing fully leakage-resilient signature schemes is dealing with leakage that may depend on the random bits used by the signing algorithm, and constructions of such schemes are known only in the random-oracle model. Moreover, even in the random-oracle model, known schemes are only resilient to leakage of less than half the length of their signing key.

In this paper we construct *fully* leakage-resilient signature schemes without random oracles. We present a scheme that is resilient to any leakage of length $(1 - o(1))L$ bits, where L is the length of the signing key. Our approach relies on generic cryptographic primitives, and at the same time admits rather efficient instantiations based on specific number-theoretic assumptions. In addition, we show that our approach extends to the continual-leakage model, recently introduced by Dodis, Haralambiev, Lopez-Alt and Wichs (FOCS '10), and by Brakerski, Tauman Kalai, Katz and Vaikuntanathan (FOCS '10). In this model the signing key is allowed to be refreshed, while its corresponding verification key remains fixed, and the amount of leakage is assumed to be bounded only in between any two successive key refreshes.

⋆ Research supported by the US National Defense Science and Engineering Graduate Fellowship. This work was partially completed while visiting the Weizmann Institute of Science.

⋆⋆ This work was partially completed while the author was a Ph.D. student at the Weizmann Institute of Science, and supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

⋆⋆⋆ This work was partially completed while visiting the Weizmann Institute of Science.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 89–108, 2011.

1 Introduction

One of the main goals of research in the foundations of cryptography is designing systems that withstand adversarial behavior. Given a cryptographic task, such as public-key encryption, one must formalize an attack model specifying a class of adversaries, and define a notion of security capturing what it means to break the system. Within such a framework, it is then possible to rigorously analyze the security of cryptographic systems.

Starting with the seminal work of Goldwasser and Micali [18], various and increasingly strong attack models and notions of security have been proposed. Over the years, however, theoreticians and practitioners began to notice that a large class of realistic attacks, called *side-channel attacks*, are not captured by the existing models. In such attacks, the adversary may learn some additional information about the internal secret state of a system, by measuring various properties resulting from specific *physical* implementations (e.g., timing information, detection of internal faults, electromagnetic radiation, power consumption etc.). As a result, it has become an important research agenda to extend the standard models to capture such side-channel attacks, and to design cryptographic systems whose security guarantees can be rigorously analyzed and clearly stated in these stronger models. Our work focuses on the model of *memory attacks*, and its *bounded-leakage* and *continual-leakage* variants, which we describe next (several other models are described in the full version).

Memory attacks: bounded-leakage and continual-leakage. The model of *memory attacks* was introduced by Akavia, Goldwasser, and Vaikuntanathan [1]. Its main premise is that the adversary can learn *arbitrary* information about the secret state of a system, subject only to the constraint that the *amount* of information learned is somehow bounded. More precisely, the adversary can adaptively select *arbitrary poly-time computable* functions $f_i : \{0,1\}^* \to \{0,1\}^{\lambda_i}$ and learn the value of f_i applied to the internal state of the system, subject only to some constraint on the output sizes λ_i.

The work of [1] assumes that there is an a priori determined *leakage bound* λ, which bounds the *overall* amount of information learned by the adversary throughout the entire lifetime of the system to be $\sum_i \lambda_i \leq \lambda$. We call this the *bounded leakage model*. Usually the leakage bound λ is also related to the secret-key size, so that a *relatively* large fraction $\lambda/|sk|$ of the secret key can be leaked. A great deal of research has gone into devising various cryptographic primitives in this model, such as public-key and identity-based encryption schemes, signature schemes, and more (see [30,26,3,2,28,8,14]).

A drawback of the bounded-leakage model is that, if a system is being used continually for a sufficiently long time, then the amount of leakage observed by the attacker may exceed any a-priori determined leakage bound. Hence, we would like to bound the *rate* of leakage rather than the *overall amount* of leakage. If we do not bound the overall leakage, then any static piece of information that stays unmodified on the system can eventually be fully recovered by the adversary. Hence the secret keys of such systems must be periodically *refreshed*. Recently, Dodis et al. [13] and Brakerski et al. [10] suggested the *continual-leakage model*,

in which a scheme periodically *self-refreshes* its internal secret key, while the corresponding public key remains fixed. In this model, only the amount of leakage seen by the adversary *in between any two successive refreshes* is assumed to be a priori bounded by some leakage bound λ[1]. However, there is no a-priori bound on the overall amount of information seen by the adversary throughout the lifetime of the system.

We note that in both the bounded-leakage model and the continual-leakage model the adversary may be able to learn partial, but yet *arbitrary*, information on the *entire* secret key. This is in contrast with other models, where either the leakage is assumed to be of "low complexity" (such as AC^0 circuits) [25,16], or certain secret values are assumed to be leak-free.

Leakage-resilient signature schemes. In this paper we study the security of signature schemes in the bounded-leakage and continual-leakage models. Signature schemes in the bounded-leakage model were proposed by Alwen, Dodis, and Wichs [3] and by Katz and Vaikuntanathan [26], who focused mainly on leakage of *(only) the signing key* of the scheme. Specifically, a signature scheme is leakage-resilient in the bounded-leakage model if it is existentially unforgeable against an adaptive chosen-message attack [19] even when adversarially chosen functions of the signing key are leaked in an adaptive fashion. Signature schemes satisfying this notion of security were constructed both based on generic cryptographic primitives in the standard model [26] and based on the Fiat-Shamir transform [17] in the random-oracle model [26,3].

Although this notion of leakage resilience already captures some attacks, it does not fully capture general leakage attacks, which may depend on the *entire internal state* of the system. In particular, the problem is that both of the signature scheme constructions from [26,3] are *randomized* and hence the internal state includes, in addition to the secret-key, all of the random coins used by the signing algorithm[2]. The prior schemes may therefore be vulnerable to leakage-attacks that (also) depend on this randomness.

This was already noted by Katz and Vaikuntanathan [26], who put forward the stricter notion of a *fully leakage-resilient* signature schemes (in the bounded-leakage model). This notion requires a signature scheme to remain existentially unforgeable under an adaptive chosen-message attack even when the adversary obtains bounded leakage information on *all intermediate values* used by the signer throughout the lifetime of the system, including the secret-keys *and* internal random coins (the notion can be naturally extended to the continual-leakage model [13,10]). This stronger notion seems to better capture real attacks, relying

[1] If the time between refreshing is fixed, we can think of this as bounding the *rate* of leakage.

[2] No known deterministic or public-coin constructions of leakage-resilient signatures are known. Without leakage, the signing algorithm of any signature scheme can be made deterministic by using, as its random coins, the output of a pseudorandom function (PRF) applied to the message, where the seed of the PRF is made part of the secret key. However, in the setting of key leakage, this transformation may no longer be secure since the seed to the PRF can also leak.

on e.g. timing or power consumption patterns, since these likely *do* depend on the internal randomness.

Currently, however, the known constructions of fully leakage-resilient signature schemes are proven secure only in the random-oracle model [3,10,13,26]. Moreover, even in the random-oracle model, known schemes are either resilient to leakage of at most half the length of the signing key [3,13,26], or require refreshing of the signing key after every few invocation of the signing algorithm, even when no leakage occurs [10] (this is required even in the bounded-leakage model, where refreshing is not part of the typical functionality). In the standard model, only constructions of "one-time" signatures[3] from [26] are known to be fully leakage resilient.

In a concurrent and independent work, Malkin, Teranishi, Vahlis and Yung [29] propose a alternate signature scheme in the continual-leakage model. Although the two schemes appear very different at first, they can be seen as separate instantiations of a common strategy, which we will explain shortly.

1.1 Our Contributions

We construct the first fully leakage-resilient signature schemes without random oracles. We first present a scheme in the bounded-leakage model that is resilient to any leakage of $(1 - o(1))L$ bits, where L is the bit-length of the signing key. Our scheme is based on generic cryptographic primitives, and is inspired by the approach of Katz and Vaikuntanathan [26] (although their scheme is resilient to leakage from the signing key only). Moreover, we show that our construction can be instantiated based on specific number-theoretic assumptions to yield a rather *efficient* scheme.

We then extend our approach to the continual-leakage model by relying on any *continual leakage-resilient one-way relation*, a primitive recently introduced by Dodis, Haralambiev, Lopez-Alt and Wichs [13]. Our resulting signature scheme construction inherits the leakage resilience properties of the underlying one-way relation with respect to leakage allowed between successive key updates and during the refreshing algorithm. In particular, instantiating our scheme with existing constructions of the one-way relations from [13,10] yields schemes that are resilient to leakage of logarithmic length from the random bits used by the refreshing algorithm, and any leakage of length $(1 - o(1))L$ bits between any two key refreshes based on the Symmetric External Diffie-Hellman (SXDH) assumption, or $(1/2 - o(1))L$ bits between refreshes based on the Decisional-Linear assumption.

Finally, we note that our approach yields the first separation between the bounded-leakage model and the noisy-leakage model, which was formalized by Naor and Segev [30] and later refined by Dodis et al. [13, Definition 7.2]. Noisy leakage is a realistic generalization of bounded leakage, in which the leakage is not necessarily of bounded length, and it is only guaranteed that the secret key

[3] Such schemes can only be used to sign a single message (or, more generally, some a priori bound t on the number of messages). The amount of leakage-resilience is $\Theta(L/t)$ bits, and thus degrades with t.

still has some min-entropy even given the leakage. This settles an open problem posed by Naor and Segev.

1.2 Overview of Our Approach

In this section we present an overview of our approach for constructing fully leakage-resilient signature schemes. We focus here on our construction in the bounded-leakage model, as it already emphasizes the main ideas underlying our approach, and we refer the reader to the full version of the paper for an overview of our construction in the continual-leakage model. We begin by describing more clearly the notion of a fully leakage-resilient signature scheme in the bounded-leakage model. Then, we briefly describe the leakage-resilient signature scheme of Katz and Vaikuntanathan [26], which serves as our starting point, and explain the main challenges in constructing *fully* leakage-resilient signature schemes. The main part of this overview then focuses on our construction.

Modeling fully leakage-resilient signature schemes. A signature scheme is fully leakage-resilient in the bounded-leakage model if it is existentially unforgeable against an adversary that can obtain both signatures on any message of her choice, and bounded leakage information on all intermediate values used by the signer throughout the lifetime of the system.

This is formalized by considering an experiment that involves a signer and an adversary. First, the signer invokes the key-generation algorithm and obtains a verification key vk and a signing key sk. At this point, a value state is initialized to contain the random coins that were used by the key-generation algorithm. The adversary is given the verification key vk and can adaptively submit two types of queries: *signing queries*, and *leakage queries*. A signing query consists of a message m, and is answered by invoking the signing algorithm with the signing key and the message. Following each such query, the random coins that were used by the signing algorithm are added to the state. A leakage query consists of a leakage function f, and is answered by applying f to the value state. The leakage functions have to be efficiently computable, and the sum of their output lengths has to be upper bounded by a predetermined parameter λ. The adversary is successful if she outputs a pair (m^*, σ^*), where m^* is a message with which she did not issue a signing query, and σ^* is a valid signature on m^* with respect to vk. We refer the reader to Section 3 for a formal definition.

The Katz-Vaikuntanathan scheme. The Katz-Vaikuntanathan signature scheme [26] relies on a second-preimage resistant (SPR) function $\mathsf{F} : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$ (for some $\kappa(n) < \mu(n)$), a CPA-secure public-key encryption scheme, and a (unbounded simulation-sound) NIZK proof system[4]. The signing key is a

[4] A function F is second-preimage resistant if, given a random input x it is hard to find $x' \neq x$ such that $\mathsf{F}(x') = \mathsf{F}(x)$. See Definition 2.1 in Section 2. We note that when F is only assumed to be a one-way function, the scheme may not always be resilient to leakage, but it is nevertheless existentially unforgeable under an adaptive chosen-message attack. In this case the scheme can be viewed as a variant of the Bellare-Goldwasser signature scheme [4].

random $x \in \{0,1\}^{\mu(n)}$, and the verification key is a triplet $(y = \mathsf{F}(x), pk, \mathsf{crs})$, where pk is a public key for the encryption scheme, and crs is a common-reference string for the proof system. A signature on a message m consists of a ciphertext c which is an encryption of $m||x$ using pk, and a proof that the ciphertext c is indeed an encryption of $m||x'$, for some $x' \in \mathsf{F}^{-1}(y)$[5].

This scheme is leakage resilient in the bounded-leakage model. That is, it satisfies the weaker variant of the above notion of security, where the leakage is allowed to depend on the signing key only. The security of the scheme is based on three main properties:

1. A typical verification key has many possible secret keys. Specifically, the set $\mathsf{F}^{-1}(y)$ is of size roughly $2^{\mu(n)-\kappa(n)}$.
2. The "real" signatures of the scheme are *computationally indistinguishable* from "fake" signatures, which are *statistically independent* of the signing key. This follows from the semantic security of the encryption scheme and from the zero knowledge of the proof system. Specifically, a "fake" signature on a message m can be produced by encrypting $m||0^n$, and then using the NIZK simulator to generate the proof.
3. Given the decryption key corresponding to pk, any valid forgery produced by the adversary can be used to extract a preimage x' of y. This follows from the soundness of the proof system, which guarantees that the adversary's forgery is a "real" signature[6] and therefore the corresponding ciphertext can be decrypted to a valid preimage x'.

These three properties are used to prove the security of the scheme as follows. Assume there is an adversary that breaks the scheme. Then, given a random pre-image x of y, we can run this adversary and (by the third property) extract some valid preimage x' from the adversary's signing forgery with a reasonable probability. This would break second-preimage resistance of F as long as we can argue that $x' \neq x$. To do so, we use the second property to replace "real signatures" with "fake signatures" without affecting the probability of recovering some valid preimage x'. But now, the signing queries do not reveal any additional information about x, given y. So the only correlated information on x that the adversary sees is the value $y = F(x)$ of size $\kappa(n)$ and the leakage of size λ. Therefore, if $\lambda \leq \mu(n) - \kappa(n) - \omega(\log(n))$, then the adversary has (information theoretically) super-logarithmic uncertainty about the value of x and hence the probability of extracting $x' = x$ from her forgery is negligible.

The main challenges. The security proof of the Katz-Vaikuntanathan scheme relies on the argument that, given many signatures of chosen messages and λ bits of leakage from the signing key x, the value x is still hard to guess by

[5] Katz and Vaikuntanathan show that it is actually possible to encrypt only x (instead of $m||x$), and include m as a label in the statement that is proved using the NIZK proof system. However, for making this informal description more intuitive, we consider here an encryption of both m and x.

[6] In fact, a stronger notion called simulation-soundness is required, because the adversary gets to see several fake proofs before generating her signature.

the adversary. However, when the leakage may depend also on the randomness used by the signing algorithm, this is no longer true, and in fact the scheme is insecure in general. The main problem is that, in the above argument, we crucially used the ability to switch "real" signatures for "fake" signatures. This step, in turn, relied on the security of the encryption scheme and the zero-knowledge property of the proofs. However, we cannot rely on these properties if the adversary can also leak on the random coins of the encryption scheme and the proof system! Consider, for example, an instantiation of the scheme with a CPA-secure encryption scheme defined as $\mathsf{Enc}_{pk}(m||x) = (\mathsf{Enc}'_{pk}(s), \mathsf{PRG}(s) \oplus (m||x))$, where Enc' is secure encryption scheme, and PRG is a pseudorandom generator that is applied on a random seed s. Leaking the seed s, whose length may be arbitrarily shorter then λ, completely reveals the signing key x. A similar instantiation for the proof system can be shown to have a similar effect when the leakage may depend on the randomness used by the prover[7].

Our approach. A natural observation is that the above problems can be avoided if the "real" and "fake" signatures cannot be distinguished *even* given the random coins used to generate them. Remember that fake signatures are statistically independent of the secret key x, while real signatures allow us to extract some preimage using an appropriate trapdoor (decryption key).

The first idea toward achieving the above is to replace the (unbounded simulation-sound) NIZK proof system with a *statistical non-interactive witness-indistinguishable (SNIWI) argument system*. On one hand we relax the (unbounded simulation-sound) zero knowledge property to *witness indistinguishability*, and on the other hand we require that proofs generated using different witnesses are *statistically* indistinguishable from each other. In particular, this guarantees that *even* a correctly generated proof is statistically independent of the witness (in our case the signing key x) used to generate it.

The harder part lies in getting an encryption scheme where the ciphertexts are independent of the message (in our case, the signing key x) that they encrypt. In particular, this clearly contradicts the decryptability of a ciphertext. We could imagine using known lossy encryption schemes, where the encryption key pk can be generated in one of two indistinguishable modes: "*injective*" mode which allows for decryptability, and "*lossy*" mode where ciphertexts statistically hide the message. But remember that we need to satisfy the following two properties simultaneously: (1) the ability to answer the adversary's signing queries with fake signatures that reveal no information about x, (2) the ability to extract a witness x' from the adversary's forgery. By setting the pk to be in either injective or lossy mode, we can achieve either property, but not at the same time! The main tool used in resolving this conflict is to design a *partitioned-lossy encryption scheme*, where the encryption of some messages is lossy while that of others is injective.

[7] Note that even a leakage function with only one output bit can be easily used to distinguish an encryption of $m||x$ from an encryption of $m||0^n$, or to distinguish the prover of the proof system from the simulator of the proof system. Thus, technically speaking, it seems that at no point in time during the various experiments of the security proof it is possible to change the way signing queries are answered.

A selectively-unforgeable signature scheme. For the reader's intuition, we first show how to achieve a weaker notion of signature security that we refer to as *selective unforgeability under a chosen-message attack*. For this notion, we assume the adversary specifies the message m^* on which she plans to forge a signature in advance, before receiving the verification key. The signing queries and leakage are still adaptive.

To achieve this notion of security, we introduce the concept of an *all-lossy-but-one (ALBO) public-key encryption scheme*. This is a tag-based public-key encryption scheme, where the encryption procedure takes as input a tag t in addition to the message. The key-generation procedure takes as input a special tag t^* and produces a key pair (pk, sk) such that encrypting under the tag t^* allows for efficient decryption with sk, but encryption under any other tag $t \neq t^*$ statistically hides the encrypted message. We call t^* the *injective* tag, and any other tag a *lossy* tag[8]. The only computational requirement is that the public key hides the injective tag t^* that was used for its generation.

We now modify the Katz-Vaikuntanathan signature scheme by using an ALBO encryption scheme instead of a standard CPA-secure scheme. To sign m, we encrypt (only) the signing key x under the tag $t = m$. We use a SNIWI argument system instead of a simulation-sound NIZK to generate the proof. To argue security, we note that since the adversary's forgery message m^* is chosen ahead of time, we can generate the encryption key pk such that $t^* = m^*$ is the only injective tag, without affecting the adversary's ability to forge – this change is indistinguishable even given full view of the signing key x and randomness of signing. Now we are in a situation where all the signing queries for $m \neq m^*$ yield signatures which are statistically independent of the signing key x, while the forgery can be used to extract some preimage x'. Therefore, we can argue as before: the bounded leakage on the secret key x and randomness of signing is short enough that x must have entropy left given this leakage, and therefore the outcome $x' = x$ is unlikely.

The full scheme. So far we described our approach as leading to the rather weak notion of selective unforgeability under a chosen-message attack. Our actual scheme is fully leakage-resilient according to the stronger notion that was discussed in the beginning of this section (i.e., where the adversary is allowed to adaptively choose m^* after seing vk and responses to all signing and leakage queries).

We note that, in the random-oracle model, there is a simple generic transformation from selective security to full security by signing the output of the random oracle applied to the message. Alternatively, in the standard model, there is a simple transformation with exponential security loss by simply "guessing" the forgery: this can yield fully secure schemes under some exponential hardness assumptions by using complexity-leveraging. Lastly, there is a completely generic transformation due to [9] (abstracting a non-generic approach of [23]) by hashing the message with a chameleon hash function [27] and signing each prefix of

[8] We note that our notion is the opposite of the notion of an all-but-one lossy trapdoor function, where there is one lossy tag and all the other tags are injective.

the hash separately. Unfortunately, this results in long signatures. All of these generic techniques also work in the setting of full-leakage resilience. We present an alternative that does not suffer from the above disadvantages.

For our actual scheme, we follow the approach of Boneh and Boyen [5] for transforming selectively-secure identity-based encryption schemes into fully secure ones using an admissible hash function (see Section 2.3). This relies on a slightly more refined "partitioning strategy" than the "all-but-one" strategy used for the selectively-secure scheme. In particular, we introduce the notion of a *$\mathcal{R}$-lossy public-key encryption scheme*. This is a generalization of an ALBO encryption scheme where the set of possible tags is partitioned into injective tags and lossy tags according to a relation $\mathcal{R}$ (in particular, there may be more than one injective tag). The main idea of this approach is to ensure that, with polynomial probability, all of the adversary's signing queries will fall into the "lossy" partition, while the forgery falls into the "injective" partition.

Comparison to [29]. An alternate way to view our combination of a SNIWI paired with a partitioned lossy encryption is as a tag-based proof system that is partitioned to be extractable for some tags and statistically witness indistinguishable for others. Our main result shows how to build fully leakage-resilient signatures from such a proof system. The work of [29] can be seen as an alternate instantiation of this strategy which relies on Groth-Sahai NIZKs [22]. These NIZKs are either statistically witness indistinguishable or extractable depending on the choice of the CRS. In the reduction in [29], the CRS of the Groth-Sahai NIZK is derived from the tag in a clever way (using the Waters Hash [33]) so as to give an alternate useful partitioning of lossy/extractable tags.

1.3 Paper Organization

In Section 2 we introduce some preliminaries and notation. Section 3 contains a definition of security in the bounded-leakage model. In Section 4 we introduce $\mathcal{R}$-lossy public-key encryption schemes, a tool used in our constructions. Section 5 contains the construction and intuition for the security proof of our signature scheme in the bounded-leakage model. Finally, in Section 6 we discuss several concluding remarks and open problems. We refer the reader to the full version of the paper for a specific instantiation of our scheme based on the Linear assumption and the extension of our scheme to the continual-leakage model.

2 Preliminaries

In this section we present some basic tools that are used in our constructions.

2.1 Second-Preimage Resistance

A family of efficiently computable functions is a pair of polynomial-time algorithms $(\mathsf{KeyGen}, \mathsf{F})$, where KeyGen is a probabilistic algorithm that on input 1^n outputs a description $s \in \{0,1\}^*$ of a function $\mathsf{F}(s,\cdot) : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$.

Such a family is *second-preimage resistant* (SPR) if given a randomly chosen input $x \in \{0,1\}^{\mu(n)}$ and a description of a randomly chosen function $s \leftarrow \mathsf{KeyGen}(1^n)$, it is computationally infeasible to find an input $x' \in \{0,1\}^{\mu(n)}$ such that $x' \neq x$ and $\mathsf{F}(s,x) = \mathsf{F}(s,x')$. This is a weakening of the notion of a family of universal one-way hash functions introduced by Naor and Yung [31], in which the input x is allowed to be chosen in an adversarial manner (but still independently of the function description s).

Definition 2.1 (Second-preimage resistance). *A family $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ of efficiently computable functions is* second-preimage resistant *if for any probabilistic polynomial-time algorithm $\mathcal{A}$ is holds that*

$$\Pr\left[\mathsf{F}_s(x') = \mathsf{F}_s(x) \wedge x' \neq x \;\middle|\; \begin{array}{c} s \leftarrow \mathsf{KeyGen}(1^n), x \leftarrow \{0,1\}^{\mu(n)} \\ x' \leftarrow \mathcal{A}(s,x) \end{array}\right] < \nu(n) \ ,$$

for some negligible function $\nu(n)$, where the probability is taken over the choice of $x \leftarrow \{0,1\}^{\mu(n)}$ and over the internal randomness of KeyGen *and $\mathcal{A}$.*

In addition, we say that $\mathcal{F} = (\mathsf{KeyGen}, \mathsf{F})$ is a family of *public-coin* second-preimage resistant functions, if it satisfies Definition 2.1 even when the algorithm $\mathcal{A}$ takes as input also the internal randomness that was used by $\mathsf{KeyGen}(1^n)$ for sampling the function. We refer the reader to [24] for more details on public-coin hash functions.

For any integer functions $\mu(n)$ and $\kappa(n)$ that are polynomially related, the existence of universal one-way hash functions (and therefore also of second-preimage resistant functions) with domain $\{0,1\}^{\mu(n)}$ and range $\{0,1\}^{\kappa(n)}$ is known to be equivalent to that of one-way functions [32]. As noted by Katz and Vaikuntanathan [26], standard constructions of universal one-way hash functions are public coin. In practice, such public-coin functions can be constructed rather easily from various number-theoretic assumptions. For example, if the discrete log problem is hard in some group $\mathbb{G}$ of prime order p, the family of functions $f_{g_1,\ldots,g_k} : \mathbb{Z}_p^k \to \mathbb{G}$ defined as $f_{g_1,\ldots,g_k}(x_1,\ldots,x_k) = \prod_{i=1}^{k} g_i^{x_i}$ is second-preimage resistant (and even collision resistant), where $g_1,\ldots,g_k \in \mathbb{G}$ are chosen uniformly and independently at random by the key-generation algorithm.

We note that for public-coin SPR functions, there is actually no need for an explicit key-generation algorithm. Without loss of generality one can define a single function $\mathsf{F}'_r(x) = (r, \mathsf{F}_s(x))$, where $s = \mathsf{KeyGen}(1^n; r)$, and this is also SPR with the same amount of "lossiness" as the family $\mathcal{F}$.

2.2 Statistical Non-interactive Witness-Indistinguishable Argument Systems

A non-interactive argument system for a language L with witness relation R_L is a triplet of algorithms $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$, where CRSGen is an algorithm generating a common reference string crs, and P and V are the prover and verifier algorithms, respectively. The prover takes as input a triplet (crs, x, w), where $(x,w) \in R_L$, and outputs a proof π. The verifier takes as input a triplet (crs, x, π) and either

accepts or rejects. In this paper we consider a setting where all three algorithms run in probabilistic polynomial time. The two requirements of an argument system are completeness and soundness with respect to efficient cheating provers. Informally, for every $(x, w) \in R_L$ the prover generates proofs that are always accepted by the verifier, and for every $x \notin L$ any efficient cheating prover has only a negligible probability of convincing the verifier to accept. An argument system is called statistical witness indistinguishable if for any $x \in L$ and any two witnesses $w_0 \neq w_1$ such that $(x, w_0), (x, w_1) \in R_L$, the proofs generated by $\mathsf{P}(\mathsf{crs}, x, w_0)$ and $\mathsf{P}(\mathsf{crs}, x, w_1)$ are statistically indistinguishable given the common reference string.

Definition 2.2 (SNIWI argument system). *A* statistical non-interactive witness-indistinguishable argument system *for a language L with witness relation R_L is a triplet of probabilistic polynomial-time algorithms* $(\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ *such that the following properties hold:*

1. **Perfect completeness:** *For every $(x, w) \in R_L$ it holds that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \mathsf{P}(\mathsf{crs}, x, w)) = 1\right] = 1 \ ,$$

   where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$*, and the probability is taken over the internal randomness of* CRSGen*,* P*, and* V*.*
2. **Adaptive soundness:** *For every probabilistic polynomial-time prover* P^* *it holds that*

$$\Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \wedge x \notin L \;\middle|\; \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n) \\ (x, \pi) \leftarrow \mathsf{P}^*(1^n, \mathsf{crs}) \end{array}\right] < \nu(n) \ ,$$

   for some negligible function $\nu(n)$
3. **Statistical witness indistinguishability:** *There exists a probabilistic polynomial-time algorithm* CRSGen_{WI} *such that:*
   - *The distributions* $\{\mathsf{CRSGen}(1^n)\}$ *and* $\{\mathsf{CRSGen}_{WI}(1^n)\}$ *are* computationally indistinguishable*.*
   - *For any triplet (x, w_0, w_1) such that $(x, w_0) \in R_L$ and $(x, w_1) \in R_L$, the distributions* $\{\mathsf{crs}, \mathsf{P}(\mathsf{crs}, x, w_0)\}$ *and* $\{\mathsf{crs}, \mathsf{P}(\mathsf{crs}, x, w_1)\}$ *are* statistically indistinguishable*, when* $\mathsf{crs} \leftarrow \mathsf{CRSGen}_{WI}(1^n)$*.*

For our construction we are interested in SNIWI argument systems for NP. Such an argument system is implied by the construction of Groth, Ostrovsky and Sahai [21] that satisfies the stronger notion of a perfect non-interactive zero-knowledge argument system. Their construction can be based on the hardness of either the Decisional Subgroup problem [7] or the Decisional Linear problem [6]. As pointed out by Groth et al. we note that in their Linear-based construction the algorithm CRSGen admits oblivious sampling (specifically, the distribution of the common reference string is statistically-close to the uniform distribution), which is a technical property that is required for our construction in the bounded leakage model.

2.3 Admissible Hash Functions

The concept of an *admissible hash function* was first defined by Boneh and Boyen [5] to convert a natural selectively-secure identity-based encryption scheme into a fully-secure one. In this paper we use such hash functions in a similar manner to convert a selectively-secure signature scheme (where the adversary declares the message to be forged ahead of time, before receiving the verification key) into a fully secure one. The main idea of an admissible hash function is that it allows the reduction in the proof of security to secretly partition the message space into two subsets, which we will label as red ($\mathtt{R}$) and blue ($\mathtt{B}$), such that there is a noticeable probability that all of the messages in the adversary's signing queries will be in the blue set, but the forgery will be on a message in the red set. This is useful if the simulator can efficiently answer signing queries in the blue set, yet break some hard problem given a valid forgery on a message from the red set. Our exposition and definition of admissible hash function follow that of Cash, Hofheinz, Kiltz, and Peikert [11].

For $K \in \{0,1,\perp\}^{\tau(n)}$, we define the function $F_K : \{0,1\}^{\tau(n)} \to \{\mathtt{R},\mathtt{B}\}$ which "colors" the space $\{0,1\}^{\tau(n)}$ of tags in the following way:

$$F_K(y) := \begin{cases} \mathtt{R} & \text{if } \forall\, i \in \{1,\ldots,\tau(n)\} \;:\; K_i = y_i \text{ or } K_i = \perp \\ \mathtt{B} & \text{otherwise} \end{cases}$$

For any $u = u(n) < \tau(n)$, we let $\mathcal{K}_{u,n}$ denote the uniform distribution over $\{0,1,\perp\}^{\tau(n)}$ *conditioned on* exactly u positions having $\perp$ values. (Note, if K is chosen from $\mathcal{K}_{u,n}$, then the map $F_K(\cdot)$ colors exactly 2^u values red.) We would like to pick a distribution $\mathcal{K}_{u,n}$ for choosing K so that, there is a polynomial probability for any set of tags $y_0,\ldots,y_q$ of y_0 being colored "red" and all other tags being colored "blue". Unfortunately, this cannot happen if we allow all tags. Instead, we will need to rely on a special hash function the maps messages x to tags y.

Let $\mathcal{H} = \{\mathcal{H}_n\}_{n\in\mathbb{N}}$ be a hash-function ensemble, where each $H \in \mathcal{H}_n$ is a polynomial-time computable function $H : \{0,1\}^* \to \{0,1\}^{\tau(n)}$. For each $H \in \mathcal{H}_n$, we define the function $F_{K,H} : \{0,1\}^* \to \{\mathtt{R},\mathtt{B}\}$, which "colors" the space $\{0,1\}^*$ according to $F_{K,H}(x) = F_K(H(x))$.

Definition 2.3 (Admissible hash function [5,11]). *We say that $\mathcal{H}$ is an* admissible hash-function ensemble *if for every $H \in \mathcal{H}$ there exists a set $\mathbf{bad}_H$ of string-tuples such that the following two properties hold:*

- *For every probabilistic polynomial-time algorithm $\mathcal{A}$ there exists a negligible function $\nu(n)$ satisfying*

$$\Pr[(x_0,\ldots,x_q) \in \mathbf{bad}_H \mid H \leftarrow \mathcal{H}_n, (x_0,\ldots,x_q) \leftarrow \mathcal{A}(1^n,H)] \le \nu(n) \;.$$

- *For every polynomial $q = q(n)$ there is a polynomial $p = p(n)$ and an efficiently computable $u = u(n)$ such that, for every $H \in \mathcal{H}_n$ and $(x_0,\ldots,x_q) \notin \mathbf{bad}_H$ with $x_0 \notin \{x_1,\ldots,x_q\}$, we have:*

$$\Pr_{K\leftarrow\mathcal{K}_{u,n}}[F_{K,H}(x_0) = \mathtt{R} \wedge F_{K,H}(x_1) = \cdots = F_{K,H}(x_q) = \mathtt{B}\,] \ge \frac{1}{p(n)} \;.$$

We note that for the application to identity-based encryption [5,11] the bad sets $\mathbf{bad}_H$ are required to be efficiently recognizable, but this is not required for our application. In addition, we say that $\mathcal{H}$ is a *public-coin* admissible hash-function ensemble, if it satisfies Definition 2.3 even when the algorithm $\mathcal{A}$ takes as input also the internal randomness that was used by $\mathsf{KeyGen}(1^n)$ for sampling the function.

The work of Boneh and Boyen [5] shows how to construct admissible hash functions from collision-resistant hash functions. Moreover, if the underlying collision-resistant hash functions are public coin, then so are the resulting admissible hash functions. As already mentioned in Section 2.1, public-coin collision-resistant hash functions can be constructed rather easily from various number-theoretic assumptions.

3 Modeling Leakage-Resilient Signature Schemes

A signature scheme is a triplet $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ of probabilistic polynomial-time algorithms with syntax:

- $(vk, sk) \leftarrow \mathsf{KeyGen}(1^n)$ outputs a verification key and signing key.
- $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ signs a message m using the singing key sk.
- $\mathsf{Verify}_{vk}(m, \sigma) \in \{0,1\}$ outputs a bit deciding wether σ is a valid signature for m.

We require perfect correctness, which states that for any valid key pair (vk, sk) output by KeyGen and any message $m \in \{0,1\}^*$ we have $\mathsf{Verify}_{vk}(m, \mathsf{Sign}_{sk}(m)) = 1$.

A signature scheme is fully leakage-resilient (FLR) in the bounded-leakage model if it is existentially unforgeable against an adversary that can obtain both signatures on any message of her choice, and bounded leakage information on all intermediate values used by the key-generation algorithm and the signer throughout the lifetime of the system. To model this, we define a variable state which includes all secret-state used by the system so far. Initially, we set state to be the random-coins of the KeyGen algorithm (note that we do not need to explicitly add sk to the state, since it can be easily computed from it by any leakage function). On each signing query made by the adversary, we append the random-coins of the signing algorithm to the state. The adversary can leak arbitrary information about state as long as the amount is overall-bounded.

Definition 3.1 (FLR security — bounded leakage). *A signature scheme* $\Pi = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *is* λ-fully-leakage-resilient in the bounded-leakage model *if for any probabilistic polynomial-time adversary* $\mathcal{A}$ *it holds that the probability of the event* $\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)$ *is negligible in* n*, where this event is defined via the following experiment:*

1. *Sample* $r \leftarrow \{0,1\}^*$*, compute* $(vk, sk) = \mathsf{KeyGen}(1^n; r)$*, and set* $\mathsf{state} = \{r\}$.
2. *The adversary* $\mathcal{A}$ *receives as input the pair* $(1^n, vk)$*, and can adaptively query a signing oracle and a leakage oracle that are defined as follows:*

- **Signing queries.** *The signing oracle receives as input a message m_i, samples $r_i \leftarrow \{0,1\}^*$, and then computes $\sigma_i \leftarrow \mathsf{Sign}_{sk}(m_i; r_i)$. It updates* $\mathsf{state} := \mathsf{state} \cup \{r_i\}$ *and outputs σ_i.*
- **Leakage queries.** *The leakage oracle receives as input a description of an efficiently computable function $f_j : \{0,1\}^* \to \{0,1\}^{\lambda_j}$, and outputs $f_j(\mathsf{state})$. We call λ_j the* output length *of the j-th leakage function.*

3. *The adversary $\mathcal{A}$ outputs a pair (m^*, σ^*).*
4. $\mathsf{Success}^{\lambda\text{-}\mathsf{FLR}}_{\Pi,\mathcal{A}}(n)$ *denotes the event in which:*
   - $\mathsf{Verify}_{vk}(m^*, \sigma^*) = 1$.
   - *m^* was not queried to the signing oracle.*
   - *The sum of output lengths of all leakage functions is at most $\lambda(n)$.*

For the definition of security within the continual-leakage model, we refer the reader to the full version of the paper.

4 $\mathcal{R}$-Lossy Public-Key Encryption

In this section we introduce the notion of an $\mathcal{R}$-lossy public-key encryption scheme. Informally, such a scheme is a tag-based public-key encryption scheme where the set of possible tags is partitioned into two subsets: *injective* tags, and *lossy* tags. When a message is encrypted under an injective tag, the resulting ciphertext can be correctly decrypted using the secret key. On the other hand, when encrypted under a lossy tag, the ciphertext statistically hides the message. The partitioning of the tags in defined by a binary relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$: the key-generation algorithm receives as input an *initialization value* $K \in \mathcal{K}$ and this partitions the set tags $\mathcal{T}$ so that $t \in \mathcal{T}$ is injective if and only if $(K, t) \in \mathcal{R}$. More, formally, we require that the relation $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ consists of a sequence of efficiently (in n) recognizable sub-relations $\mathcal{R}_n \subseteq \mathcal{K}_n \times \mathcal{T}_n$.

The only computational requirement of an $\mathcal{R}$-lossy public-key encryption scheme is that the public key of the encryption scheme hides the initialization value K. That is, public keys produced by different initialization values are computationally indistinguishable.

Definition 4.1 ($\mathcal{R}$-lossy PKE). *Let $\mathcal{R} \subseteq \mathcal{K} \times \mathcal{T}$ be an efficiently computable binary relation. An* $\mathcal{R}$-lossy public-key encryption scheme *is a triplet of probabilistic polynomial-time algorithms* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *such that:*

1. **Key generation:** *For any initialization value $K \in \mathcal{K}_n$, the key-generation algorithm* KeyGen *on input $(1^n, K)$ outputs a secret key sk and a public key pk.*
2. **Decryption under injective tags:** *For any initialization value $K \in \mathcal{K}_n$ and tag $t \in \mathcal{T}_n$ such that $(K, t) \in \mathcal{R}_n$, and for any message $m \in \{0,1\}^{\ell(n)}$, it holds that*
$$\Pr\left[\mathsf{Dec}^t_{sk}(\mathsf{Enc}^t_{pk}(m)) = m\right] > 1 - \nu(n) \ ,$$
for some negligible function $\nu(n)$, where $(sk, pk) \leftarrow \mathsf{KeyGen}(1^n, K)$, and the probability is taken over the internal randomness of KeyGen, Enc *and* Dec.

3. **Lossiness under lossy tags:** *For any initialization value $K \in \mathcal{K}_n$ and tag $t \in \mathcal{T}_n$ such that $(K,t) \notin \mathcal{R}_n$, for every pair (sk,pk) of keys produced by $\mathsf{KeyGen}(1^n,K)$, and for every two messages $m_0, m_1 \in \{0,1\}^{\ell(n)}$, the distributions $\mathsf{Enc}^t_{pk}(m_0)$ and $\mathsf{Enc}^t_{pk}(m_1)$ are statistically indistinguishable.*
4. **Indistinguishability of initialization values:** *For every sequence of pairs $\{(K_n, K'_n)\}_{n\in\mathbb{N}}$ such that $K_n, K'_n \in \mathcal{K}_n$, the two ensembles $\{pk : (sk,pk) \leftarrow \mathsf{KeyGen}(1^n, K_n)\}_{n\in\mathbb{N}}$ and $\{pk : (sk,pk) \leftarrow \mathsf{KeyGen}(1^n, K'_n)\}_{n\in\mathbb{N}}$ are computationally indistinguishable.*

As with the other primitives that are used in our construction, we need to be able to obliviously sample public keys in a way that is computationally indistinguishable from those produced by $\mathsf{KeyGen}(1^n,\cdot)$. Specifically, we require that there exists a sequence of initialization values $\{K_n\}_{n\in\mathbb{N}}$ such that the ensemble $\{pk : (sk,pk) \leftarrow \mathsf{KeyGen}(1^n,K_n)\}_{n\in\mathbb{N}}$ is computationally indistinguishable from the uniform distribution over $\{0,1\}^*$. Note that by the indistinguishability of initialization values property defined above, this in fact holds for every sequence $\{K_n\}_{n\in\mathbb{N}}$.

For our constructions of fully leakage-resilient signature schemes we consider two relations: the equality relation $\mathcal{R}^{\mathsf{EQ}}$, and the more general "bit-matching" relation $\mathcal{R}^{\mathsf{BM}}$ that is defined below.

The relation $\mathcal{R}^{\mathsf{EQ}}$. The relation $\mathcal{R}^{\mathsf{EQ}}$ is the equality relation for binary tags of length $\tau(n)$ bits. That is, $\mathcal{K}_n = \mathcal{T}_n = \{0,1\}^{\tau(n)}$, and $(K,t) \in \mathcal{R}^{\mathsf{EQ}}_n$ if and only if $K = t$. An $\mathcal{R}^{\mathsf{EQ}}$-lossy encryption is just an *all-but-one-lossy* (ALBO) public-key encryption scheme, a primitive discussed in the introduction. In this case there is one injective tag, corresponding to the value of K used during initialization, and all the other tags are lossy.

The relation $\mathcal{R}^{\mathsf{BM}}$. The bit-matching relation $\mathcal{R}^{\mathsf{BM}}$ is a generalization of equality, which allows for more complex partitions. For $\mathcal{K}_n = \{0,1,\perp\}^{\tau(n)}$, $\mathcal{T}_n = \{0,1\}^{\tau(n)}$ define $(K,t) \in \mathcal{R}^{\mathsf{BM}}_n \subseteq \mathcal{K}_n \times \mathcal{T}_n$ iff for every $i \in \{1,\ldots,\tau(n)\}$ it holds that $K_i = t_i$ or $K_i = \perp$. That is, given some fixed initialization value K, the set of injective tags t are exactly those whose bits match K in all positions i for which $K_i \neq \perp$. Notice that, if K does not contain any $\perp$ symbols, then there is a *single* injective tag $t = K$ and all other tags are lossy. Therefore $\mathcal{R}^{\mathsf{BM}}$-lossy encryption is a strict generalization of $\mathcal{R}^{\mathsf{EQ}}$-lossy encryption.

In our signature scheme construction, the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption will be used in combination with an *admissible hash function* (discussed in Section 2.3). The admissible hash function gives us a way to map messages to encryption tags such that, with high probability over an appropriate distribution of K, all signing queries map to lossy tags while the forgery maps to an injective tag.

Constructions. In the full version, we propose two constructions of $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption schemes[9]. Our first construction is rather generic and is based on any lossy public-key encryption scheme. In turn, this implies $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption schemes can be based on a variety of number-theoretic

[9] We note that rather straightforward variants of these constructions yield $\mathcal{R}^{\mathsf{EQ}}$-lossy public-key encryption schemes.

assumptions. Our second construction is based on a specific number-theoretic assumption (the DDH assumption[10]) and is significantly more efficient than our generic construction.

5 A Signature Scheme in the Bounded-Leakage Model

In this section we present our construction of a fully leakage-resilient signature scheme in the bounded-leakage model (see Definition 3.1). We use the following primitives in a generic manner:

- Let $\mathcal{F} = (\mathsf{KeyGen}_{\mathsf{SPR}}, \mathsf{F})$ be a family of public-coin second-preimage resistant functions $\mathsf{F}_s(\cdot) : \{0,1\}^{\mu(n)} \to \{0,1\}^{\kappa(n)}$ for some $\kappa(n) < \mu(n)$ (see Section 2.1).
- Let $\mathcal{H}$ be a public-coin admissible hash function ensemble (see Section 2.3).
- Let $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}, \mathsf{Enc}, \mathsf{Dec})$ be an $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption scheme (see Section 4).
- Let $\Pi = (\mathsf{CRSGen}, \mathsf{P}, \mathsf{V})$ be a SNIWI argument system for the language

$$L = \{(s, y, pk, t, C) : \exists x, \omega \text{ st } C = \mathsf{Enc}^t_{pk}(x; \omega) \text{ and } \mathsf{F}_s(x) = y\}$$

(see Section 2.2).

We assume that the distribution of public keys and common-reference strings produced by the algorithms $\mathsf{KeyGen}_{\mathcal{R}^{\mathsf{BM}}}$ and CRSGen, respectively, are computationally indistinguishable from the uniform distribution over $\{0,1\}^*$[11]. Define the signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- **Key generation:** On input 1^n, the algorithm KeyGen samples a uniformly distributed $x \leftarrow \{0,1\}^{\mu(n)}$, a function description $s \leftarrow \mathsf{KeyGen}_{\mathsf{SPR}}(1^n)$ from the SPR family, and computes $y = \mathsf{F}_s(x)$. Then, it samples a description of an admissible hash function $H \leftarrow \mathcal{H}_n$, and samples $pk \leftarrow \{0,1\}^*$ and $\mathsf{crs} \leftarrow \{0,1\}^*$ to be used as a public key for the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme and a common-reference string for the SNIWI argument system, respectively. It outputs the signing key $sk = x$ and the verification key $vk = (s, y, H, pk, \mathsf{crs})$.
- **Signing:** On input message m, the algorithm Sign computes an encryption $C = \mathsf{Enc}^{H(m)}_{pk}(x; \omega)$ of x under the tag $H(m)$ using fresh randomness ω. Then, it invokes the prover of the argument system to obtain a proof $\pi \leftarrow \mathsf{P}(\mathsf{crs}, (s, y, pk, H(m), C), (x, \omega))$, and outputs the signature (C, π).
- **Verifying:** On input message m and signature $\sigma = (C, \pi)$, the algorithm Verify invokes the verifier of the argument system and outputs 1 if and only if $\mathsf{V}(\mathsf{crs}, (s, y, pk, H(m), C), \pi) = 1$.

[10] Our construction easily generalizes to rely on the d-Linear assumption for any $d \geq 1$.

[11] More generally, we just require "oblivious"sampling, but we will assume uniform distribution for simplicity. See Appendix 2.

Theorem 5.1. *Assuming the existence of the schemes $\mathcal{F}$, $\mathcal{H}$, $\mathcal{E}$ and Π with properties described above, the scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is λ-fully-leakage-resilient in the bounded-leakage model for any $\lambda = \mu(n) - \kappa(n) - \omega(\log n)$. The relative leakage is given by $\lambda/|sk| \approx (1 - \kappa(n)/\mu(n)) = (1 - o(1))$ for an appropriate choice of $\kappa(n) = o(\mu(n))$.*

Due to space limitations the proof of Theorem 5.1 is left to the full version of the paper, and we only give a short proof outline here.

Proof outline. Suppose there is an adversary who breaks the security of the scheme. We can then use the adversary to break the security of the SPR function as follows. Choose a random crs for the SNIWI argument honestly, and a (pk, sk) pair for $\mathcal{R}^{\mathsf{BM}}$-lossy public-key encryption using an initialization value K sampled from an appropriate distribution (dictated by the admissible hash function, depending on the number of signing queries the adversary makes). Given a random challenge x from the SPR challenger, we embed $y = \mathsf{F}(x), \mathsf{crs}, pk$ into the verification key and then run the forging adversary, using x to answer all its signing/leakage queries. If the adversary's forgery is on a message m^* that corresponds to a injective tag of the encryption scheme, then we use sk to decrypt a (hopefully second preimage) x' from the adversary's forged signature. We argue that, with polynomial probability, we do recover a *second preimage* $x' \neq x$, using the following steps:

- Using the partitioning argument of Boneh-Boyen [5], there is a noticeable probability that the all of the adversary's signing queries correspond to "lossy" tags while the forgery corresponds to an "injective" tag. Here we rely on the property that the initialization value K is hidden by the public-key. We call an execution where the above occurs a "good execution."
- In a good execution, the adversary's forgery can be decrypted to a valid preimage $x' \in \mathsf{F}^{-1}(y)$, by the soundness of the SNIWI argument.
- Information theoretically, the probability of $x' = x$ in a good execution is negligible, since the adversary just doesn't have enough information about x. That is, the signature-query responses are independent of x, and the leakage-query responses and the verification key y are too short. This is formalized with an entropy argument.

6 Concluding Remarks and Open Problems

Deterministic leakage-resilient signatures. An alternative approach for constructing fully leakage-resilient signature schemes is constructing a signature scheme that is resilient to leakage from the signing key, and has a deterministic signing algorithm (this is indeed the idea underlying the fully leakage-resilient *one-time* signature schemes of Katz and Vaikuntanathan [26]). In general, the signing algorithm of any signature scheme can be made deterministic by using as its random coins the output of a pseudorandom function applied to the message. This requires, however, that the signing key will include also the key of the pseudorandom function, and therefore it is not clear that such a transformation can preserve leakage resilience.

Bounded leakage vs. noisy leakage. In some scenarios it is not always possible to assume that the total amount of leakage is upper bounded by λ bits. This motivated the approach of Naor and Segev [30] (later refined by Dodis et al. [13, Definition 7.2]) who considered the more general notion of *noisy leakage*, in which the leakage is not necessarily of bounded length, but is guaranteed to reduce the average min-entropy of the secret key by at most λ. Although our schemes are secure with respect to bounded leakage, they are in fact insecure with respect to noisy leakage. This seems to be the first separation between bounded leakage and noisy leakage, and this settles an open problem posed by Naor and Segev.

Specifically, in our schemes the public key for the $\mathcal{R}^{\mathsf{BM}}$-lossy encryption scheme is sampled obliviously as a uniformly random string $pk \in \{0,1\}^*$. For our specific constructions based on the DDH or Linear assumptions (see full version), this can be easily seen to imply that with an overwhelming probability all possible tags for the $\mathcal{R}^{\mathsf{BM}}$-lossy scheme are lossy. An analysis almost identical to that presented in the security proofs of our schemes then shows that a leakage function that simply outputs a signature on any message m^* is a valid leakage function with respect to noisy leakage (yet clearly invalid with respect to bounded leakage).

Modeling hard-to-invert leakage for signature schemes. In the setting of public-key encryption a more general model of leakage was formalized by only assuming that the decryption key cannot be efficiently recovered given the leakage (see [15,12,20,8] and the references therein). For signature schemes, however, it is not clear how to meaningfully formalize such an attack model. It would be interesting to formalize hard-to-invert leakage for signature schemes (especially when any intermediate value may leak, and not only the signing key), and to construct schemes that are leakage resilient in such a model.

Acknowledgements

We thank Moni Naor, Brent Waters, and the Eurocrypt '11 referees for many useful comments on this work.

References

1. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
2. Alwen, J., Dodis, Y., Naor, M., Segev, G., Walfish, S., Wichs, D.: Public-key encryption in the bounded-retrieval model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 113–134. Springer, Heidelberg (2010)
3. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)

4. Bellare, M., Goldwasser, S.: New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 194–211. Springer, Heidelberg (1990)
5. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
7. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
8. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic Residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
9. Brakerski, Z., Tauman Kalai, Y.: A framework for efficient signatures, ring signatures and identity based encryption in the standard model. Cryptology ePrint Archive, Report 2010/086 (2010)
10. Brakerski, Z., Tauman Kalai, Y., Katz, J., Vaikuntanathan, V.: Cryptography resilient to continual memory leakage. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 501–510 (2010)
11. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
12. Dodis, Y., Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Public-key encryption schemes with auxiliary inputs. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 361–381. Springer, Heidelberg (2010)
13. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, pp. 511–520 (2010)
14. Dodis, Y., Haralambiev, K., Lopez-Alt, A., Wichs, D.: Efficient public-key cryptography in the presence of key leakage. Cryptology ePrint Archive, Report 2010/154 (2010)
15. Dodis, Y., Tauman Kalai, Y., Lovett, S.: On cryptography with auxiliary input. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 621–630 (2009)
16. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
17. 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)
18. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
19. 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)
20. Goldwasser, S., Tauman Kalai, Y., Peikert, C., Vaikuntanathan, V.: Robustness of the learning with errors assumption. In: Proceedings of the 1st Symposium on Innovations in Computer Science, pp. 230–240 (2010)

21. 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)
22. 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)
23. Hohenberger, S., Waters, B.: Short and stateless signatures from the RSA assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009)
24. 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)
25. 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)
26. Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
27. Krawczyk, H., Rabin, T.: Chameleon signatures. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) (2000)
28. Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)
29. Malkin, T., Teranishi, I., Vahlis, Y., Yung, M.: Signatures resilient to continual leakage on memory and computation (2010) (manuscript)
30. Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
31. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 33–43 (1989)
32. Rompel, J.: One-way functions are necessary and sufficient for secure signatures. In: Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, pp. 387–394 (1990)
33. 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 Formal Study of Power Variability Issues and Side-Channel Attacks for Nanoscale Devices

Mathieu Renauld, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Dina Kamel, and Denis Flandre

UCL Crypto Group, Université catholique de Louvain,
Place du Levant 3, B-1348, Louvain-la-Neuve, Belgium

Abstract. Variability is a central issue in deep submicron technologies, in which it becomes increasingly difficult to produce two chips with the same behavior. While the impact of variability is well understood from the microelectronic point of view, very few works investigated its significance for cryptographic implementations. This is an important concern as 65-nanometer and smaller technologies are soon going to equip an increasing number of security-enabled devices. Based on measurements performed on 20 prototype chips of an AES S-box, this paper provides the first comprehensive treatment of variability issues for side-channel attacks. We show that technology scaling implies important changes in terms of physical security. First, common leakage models (e.g. based on the Hamming weight of the manipulated data) are no longer valid as the size of transistors shrinks, even for standard CMOS circuits. This impacts both the evaluation of hardware countermeasures and formal works assuming that independent computations lead to independent leakage. Second, we discuss the consequences of variability for profiled side-channel attacks. We study the extend to which a leakage model that is carefully profiled for one device can lead to successful attacks against another device. We also define the perceived information to quantify this context, which generalizes the notion of mutual information with possibly degraded leakage models. Our results exhibit that existing side-channel attacks are not perfectly suited to this new context. They constitute an important step in better understanding the challenges raised by future technologies for the theory and practice of leakage resilient cryptography.

Introduction

Side-channel attacks are one of the most important threats against modern cryptographic implementations. Since the apparition of power [11] and electromagnetic analysis [6,21], the design and evaluation of countermeasures allowing to withstand such physical attacks has become an increasingly important research topic. The security assessment of commercial products (such as smart cards) has also implied major developments in the industry of secure hardware devices. Various solutions purposed to increase the security against side-channel attacks have been proposed, at different abstraction levels. They range from the modification of the hardware [31] to generic techniques using the formalism of

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 109–128, 2011.

modern cryptography [20]. Significant progresses have also been made in better understanding the statistical aspects of power analysis and its connection with countermeasures such as masking and hiding, as detailed in the DPA book [14].

By contrast to classical cryptanalyis, that targets abstract mathematical objects, side-channel cryptanalysis is implementation-specific. The gain of such a specialization is a significantly increased power. Cryptographic algorithms that are assumed (or proven) secure against classical adversaries, even with intensive time and memory complexities, often turn out to be completely insecure against physical attacks, if implemented in an unprotected device. As a consequence, technological dependencies are at the core of both the theory and practice of side-channel analysis. On the one hand, solutions to attack cryptographic implementations are most efficient if they can exploit a good understanding of the underlying physics. On the other hand, solutions to (provably) ensure the security of leaking devices need to rely on assumptions that correctly capture the peculiarities of actual hardware. In this paper, we tackle this issue of technological dependency and show that some of the common assumptions used in power analysis attacks are not going to hold anymore in future cryptographic hardware.

In particular, the scaling of the CMOS technology, that is the basis of most present microelectronic devices, is a permanent trend since the apparition of integrated circuits in the late 1950s. Shrinking transistors is generally motivated by the need of increased performances and reduced energy per operation. But when reaching the nanometer scale, two major detrimental side effects also arise. First, the relative importance of so-called static currents increases (i.e. energy is consumed, even if no computation is performed) [25]. Second, device variability becomes important (i.e. it becomes increasingly difficult to engineer identical chips) [1,17]. As a consequence, the goal of this paper is to investigate the impact of these effects, with a focus on power variability, from the point of view of side-channel attacks. More precisely, our contributions are as follows.

1. A classical tool in DPA is to use Pearson's correlation coefficient in order to compare key-dependent leakage predictions with actual measurements performed on a chip [2]. These (so-called) correlation attacks are most efficient if a good leakage model is available for the predictions. And a very common solution is to use the Hamming weight (or distance) of the manipulated data for this purpose. We show that such models are not accurate anymore for 65-nanometer and smaller technologies. Hence, their use may lead to overestimate the security of a (protected or unprotected) implementation.
2. Recent works in the area of leakage resilient cryptography frequently assume that independent computations lead to independent leakage. We put forward that this assumption is not fulfilled anymore for 65-nanometer technologies. In particular, we show that linear leakage models that only depend on the input/output bits of an S-box are not able to capture parasitical effects occurring during the computations. We then discuss the consequences of this observation and highlight that they are different for works such as [5], which assume independence at the gate level, and works such as [4], which assume independence at a larger scale, e.g. between functional blocks.

3. Profiled attacks, e.g. using templates [3] or stochastic models [27], are an important class of side-channel attacks in which an adversary first characterizes a target device (in order to obtain a precise knowledge of the leakage probability distributions), and then uses this knowledge in a very powerful online phase. In this context, it is important to know whether a profile obtained from one device can be used against other similar devices. We discuss this question in light of the increased variability of recent technologies. For this purpose, we define the perceived information, which is a generalization of the mutual information that allows quantifying degraded leakage models.
4. Finally, we provide a careful empirical evaluation of both the information leakage and the success rates of various implementations and attacks. Our results are based on a set of 20 implementations of the same AES S-box in a 65-nanometer low-power CMOS technology. We use these experiments to discuss the impact of the power supply on the information leakage, and the selection of meaningful time samples in the traces. We also take advantage of this case study to compare real measurement traces with simulated ones.

Summarizing, while an important literature covers the impact of nanoscale technologies from a microelectronic point of view, e.g. [7], only a few works consider its consequences in terms of security. To the best of the authors' knowledge, the simulated experiments in [13] are the only available reference. In this paper, we extend these preliminary investigations, and show that technology scaling implies new challenges for the theory and practice of side-channel attacks, that are not completely solved by present statistical tools, proof techniques and assumptions.

1 Preliminaries

1.1 Target Implementation

Our analysis is based on simulated and actual power traces obtained from the execution of an AES Rijndael S-box, full-custom designed in a low power 65-nanometer CMOS technology, and measured under two different supply voltages: 1.2V and 0.5V. We used an area-optimized S-box architecture based on composite field arithmetic, described in [16], of which the design is detailed in [9].

Measurements were performed on 20 prototype chips implementing this S-box, each of them made of 1,530 transistors in static CMOS logic style, with a maximum logic depth of 22. The S-box delay is 3 ns at 1.2V supply voltage, meaning a maximum operating frequency of 200 MHz (taking a security margin of 2 ns). This maximum clock frequency drops down below 10 MHz when decreasing the supply to 0.5V. In our experiments, we monitored the voltage drop on a resistor introduced in the supply circuit of the chips, using a high sampling rate oscilloscope (1 Gsample/second), while running the chip at 2 MHz (motivated by interface constraints of our prototype board). Post-layout simulations were performed using Spice models provided by the same industrial foundry as for actual measurements, for the chosen technology node.

1.2 Notations

Let a power trace l be the output of a leakage function L. In our experiments, the leakage function will essentially depend on three input arguments: X, C and N. The (discrete) random variable X denotes the input value of the S-box under investigation, the (discrete) random variable C denotes the index of the chip under investigation, the (continuous) random variable N denotes the noise in the measurements. As a result, we denote the random variable representing the leakage traces as $L(.,.,.)$, where the arguments are written as capital letters if they are variable, and as small caps if they are fixed. For example, $l(x, c, n)$ is a single measurement trace, corresponding to input x and chip c; $L(x, c, N)$ is a random variable representing the noisy traces corresponding to input x and chip c. We also denote the t^{th} time sample in a leakage trace as $L_t(x, c, n)$. Finally, it is sometimes convenient to consider noise-free mean traces, that are defined as:

$$\overline{L}(X, C) = \underset{n}{\mathbf{E}}\, L(X, C, n),$$

where $\mathbf{E}$ denotes the mean operator, which is to be replaced by a sample mean operator (denoted as $\hat{\mathbf{E}}$) when applied to actual measurement traces. Simulation environments such as Spice do not directly allow parametrizing the noise level in the power traces. Therefore, they provide noise-free traces by default. In this case, and in order to analyze the impact of noise on the security of our AES S-box, our evaluations considered an additive Gaussian noise (which is a reasonable starting point for the simulated analysis of side-channel attacks). We denote with $\mathcal{N}(l|\mu, \sigma^2)$ the probability density function (pdf) of a normal random variable L with mean μ, variance σ_n^2 and evaluated on input x. It yields:

$$L_t(X, C, N) = L_t^{sim}(X, C) + N,$$

where N has mean 0 and variance σ_n^2. When considering multiple time samples in the traces (i.e. $L_{t_1:t_d}(X, C, N)$), the mean and variance are replaced by a mean vector and a covariance matrix. Our simulated evaluations assume the same noise distribution for all inputs, chips and time samples. By contrast, when considering actual power traces, the noise is directly present in the measurements obtained from the oscilloscope. In this case, our evaluations characterized its distribution, in order to take possible correlation between different time samples into account.

As an illustration, Figure 9 in Appendix A shows noise-free power traces corresponding to 4 different inputs, measured for 10 different chips, under 1.2V and 0.5V supply voltages, obtained from simulations and actual measurements.

1.3 Noise Distribution

The preliminary analysis of a set of leakage traces usually starts with the characterization of the noise. For this purpose, we first applied the filtering described in Appendix B, in order to remove some parasitic frequencies from the traces. Then, we tested the distribution of the residual noise. In recent works on side-channel attacks, this distribution is usually assumed to be normal, with mean

zero and variance σ_n^2 [14]. Using a normality test like the Pearson's chi-square test telled us that, formally, the residual noise does not exactly follow a normal distribution. However, the ratio between the entropy of the estimated normal distribution and its Kullback-Leibler divergence with the actual distribution is smaller than 0.5%, meaning that the residual noise distribution is very close to Gaussian. As will be seen in the following section, this assumption is also validated from a side-channel point of view, when comparing the information leakage computed with the actual noise distribution and with a Gaussian estimate.

1.4 Physical Variability

The power consumption traces of an electronic device can be divided into a static part and a dynamic part. These parts can be informally identified by visual inspection: the static power corresponds to the constant parts of the traces, the dynamic power corresponds to their variable parts. Dynamic power is usually the most useful in side-channel attacks, because its strong input-dependency can be used to accumulate information about a secret value manipulated by a device. As discussed in [10], physical variability of the dynamic energy in nanoscale devices can be explained by capacitance fluctuations that are magnified when the computation delays increase, because of the random glitches that are generated by variability-induced unbalanced logic paths. In the following, we will mainly be interested in two parameters that influence the physical variability.

First, the supply voltage can be scaled down, resulting in a reduced dynamic power at the cost of an increased delay, hence implying a higher variability. Second, different time samples can be selected in the traces. Because of the impact of the computation delays on the random glitches in these traces, the samples corresponding to the beginning of the computations have less variability than the ones corresponding to the end of the computations. These parameters can be illustrated by looking at the variance of the power traces, over the input plaintexts and chips, in Figure 1. One can see that the variance over the chips (caused by physical variability) increases when moving from 1.2V to 0.5V supply voltage. In addition, for the 0.5 supply, i.e. when variability becomes significant, this variance is quite localized in the late time samples. Note that this effect is particularly visible when considering the actual measurements.

1.5 Dimensionality Reduction

One difficult task when performing a side-channel attack is to select the samples of interest in the traces. Many heuristics have been proposed for this purpose. A straightforward solution is to apply the attacks to all the samples in the traces and to select the samples where they perform best. This is possible, e.g. when applying Kocher's DPA [11], correlation attacks [2] or template attacks [3] (as long as the templates are only built for a reduced number of samples). Alternatively, it is also possible to use dimensionality reduction techniques such as Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) [29]. These are linear transforms that can be used to project the traces in a subspace of small

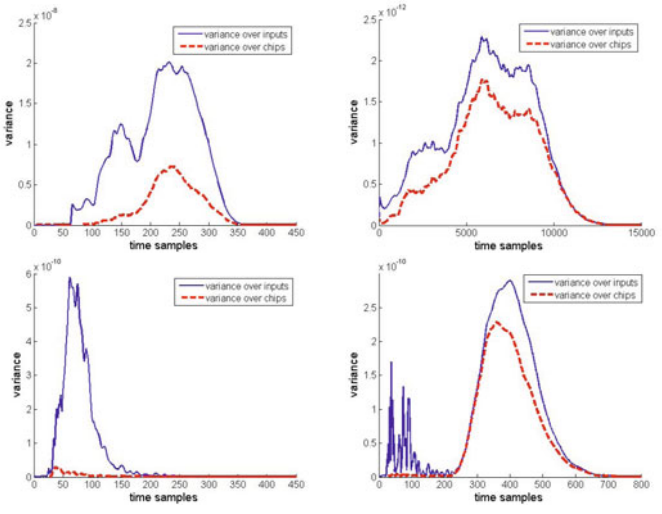


Fig. 1. Variances of the power traces over the input plaintexts and chips. Left: 1.2V power supply, Right: 0.5V power supply / Up: simulations, Down: actual measurements.

dimensionality, with the goal of "summarizing" the useful information in a few samples. PCA uses the inter-class variance as optimization criteria, while LDA uses the ratio between inter- and intra-class variance. Figure 11 in Appendix C plots the eigenvectors corresponding to the principal component produced by PCA and LDA, for simulated traces. It shows that physical variability makes the application of PCA irrelevant, as it cannot distinguish between inter-plaintext and inter-chip variances. By contrast, LDA does a good job in this case, and only selects early time samples in the traces, where inter-plaintext variance is large and inter-chip variance is small. In order to simplify the interpretation of the results, our analyzes in the following sections will reduce the dimensionality by selecting one to three meaningful time samples, with small, medium and large variability (examples are give in the upper left part of Figure 9 in Appendix A).

2 Information Theoretic Analysis

The goal of this paper is to investigate how inter-chip variability affects the application of side-channel attacks. For this purpose, we start with an information theoretic analysis. As detailed in [28], it allows to quantify the security of an implementation against an adversary who can perfectly profile the leakage pdf. In our context, we will consider the information between a secret S-box input X and the corresponding leakage L. We analyzed three types of leakage. First, we used simulations $L_t^1 = L_t^{sim}(X, C) + N$. Second, we used actual measurements $L_t^2 = L_t(X, C, N)$. Third, we considered a hybrid situation combining the average traces obtained from the oscilloscope with simulated noise: $L_t^3 = \overline{L}_t(X, C) + N$.

Interestingly, the presence of inter-chip variability implies new concerns regarding the profiling of a leakage pdf. In our 65-nanometer technology, two pieces of silicon implementing the same functionality can give rise to different power consumptions. And this variability can even occur intra-chip, e.g. two S-boxes within the same implementation of the AES can have different leakage models. As a consequence, this section will focus on two main scenarios. In the first one, the profiling and attack are performed on the same chip. This scenario reflects the classical assumption that two chips produced from the same design leak in a similar way. It corresponds to a worst case situation in which all the information leaked by an implementation can be exploited by the adversary. In the second (more realistic) one, different chips are used for profiling and attacking. We study the possibility of building templates from a set of n chips and to attack a $n{+}1^{\text{th}}$ chip, in order to infer the effect of process variability. Doing this, we introduce a new notion of "perceived information", which allows capturing the information loss that is due to the degradation of an adversary's templates.

This section will also consider two additional questions. First, we evaluate the assumption of "independent leakage" that is frequently required by formal security analyzes in physically observable cryptography, e.g. [4,5]. Then, we discuss the notion of model soundness and its relation with the scenarios of standard DPA attacks [2,3,11,15] and algebraic side-channel attacks [23,24,26].

2.1 Worst Case Scenario: Profiling and Attacking the Same Chip

Analyzing the information leakage of a cryptographic implementation first requires to choose a profiling technique, in order to estimate the leakage pdf. In this section, we use the template attacks introduced in [3], which are the most generic solution for this purpose[1]. Template attacks essentially work in two steps. In a first profiling phase, the adversary builds 256 Gaussian templates, denoted as $\hat{\Pr}_{\mathtt{model}}[L|x] = \mathcal{N}(l|\hat{\mu}_{x,c,N}, \hat{\sigma}^2_{x,c,N})$, corresponding to the 256 maximum likelihood estimates of the conditional density functions $\Pr_{\mathtt{chip}}[L|x]$. Then, in a second online phase, he uses these templates to recover information from a leaking chip, for which he will select the maximum likelihood input candidate:

$$\tilde{x} = \underset{x^*}{\operatorname{argmax}}\ \hat{\Pr}_{\mathtt{model}}[x^*|l]. \tag{1}$$

The information theoretic analysis introduced in [28] consists in evaluating the posterior probability of different inputs and computing the mutual information:

$$\mathrm{MI}(X;L) = \mathrm{H}[X] - \sum_{x\in\mathcal{X}} \Pr[x] \sum_{l\in\mathcal{L}} \Pr_{\mathtt{chip}}[l|x].\log_2 \Pr_{\mathtt{chip}}[x|l], \tag{2}$$

where $\Pr_{\mathtt{chip}}[x|l]$ is derived from $\Pr_{\mathtt{chip}}[l|x]$ using Bayes' formula and $\mathcal{X}, \mathcal{L}$ are the sets of all possible input values and leakage. In practice, the real leakage distribution is a priori unknown, both for adversaries and evaluators. Hence, the

[1] An alternative is to use stochastic models [27] and is discussed later in the paper.

probability of the leakage l conditioned on input x is replaced by a sample estimate $\hat{\Pr}_{\texttt{chip}}[l|x]$ (i.e. typically, one divided by the number of measured traces). And the probability of the input x conditioned on leakage l is replaced by the adversary's model estimate $\hat{\Pr}_{\texttt{model}}[l|x]$. In general, one assumes that the adversary's model is reasonably close to the actual chip leakages, which allows to formally compute the mutual information. As demonstrated in [30] in the context of the masking countermeasure, the mutual information provides an excellent indicator of the template adversary's success rate. However, if the adversary's model degrades for some reason, and differs from the actual chip leakage distribution, the mutual information cannot be computed anymore. In order to capture such situations, we introduce the following definition of perceived information:

$$\hat{\mathrm{PI}}(X;L) = \mathrm{H}[X] - \sum_{x\in\mathcal{X}} \Pr[x] \sum_{l\in\mathcal{L}} \hat{\Pr}_{\texttt{chip}}[l|x].\log_2 \hat{\Pr}_{\texttt{model}}[x|l]. \tag{3}$$

When profiling and attacking the same chip with sufficiently accurate templates, $\hat{\Pr}_{\texttt{chip}}[l|x]$ and $\hat{\Pr}_{\texttt{model}}[l|x]$ are the same, and the perceived information reverts to the mutual information. From a side-channel point of view, the intuitive meaning of the perceived information is close to the one of mutual information: it captures the information about a latent variable X obtained when observing leakages L, generated according to a density $\Pr_{\texttt{chip}}[L|x]$, and interpreted with the model $\hat{\Pr}_{\texttt{model}}[L|x]$. This implies that the perceived information is lower or equal to the mutual information, and may have a negative value, meaning that the leakage is misinterpreted by the adversary's model. In this case, the side-channel attacks do not converge towards their correct result (i.e. they don't output the correct key). The perceived information can also decrease with measurement noise. Such a counterintuitive behavior will be observed in the next sections: less measurement noise may increase the misinterpretations of the model, as the probability of the correct event $\hat{\Pr}_{\texttt{model}}[x|l]$ will be closer to zero in this case. Note finally that, in the case of simulations, the sum over the leakages l, in Equations (2) and (3), becomes an integral, as an analytical description of the pdf is available.

The results of our analysis for the worst case scenario where we profile and attack the same chip are displayed in Figure 2 (averaged over 20 chips), with models using 1, 2 or 3 samples (denoted as 1D, 2D and 3D in the plots). They do not exhibit deviations from previous information theoretic analyzes (e.g. the perceived information is always positive). They also confirm the intuition that reducing the power supply reduces the information leakage, and that higher dimension leakage provides more information to the adversary [22]. In fact, the most interesting observations in these experiments relate to simulations:

1. *Simulated noise.* As witnessed by the right part of the figure, average measurements plus simulated noise (i.e. L_t^3) provide an excellent approximation of actual measurements with real noise (i.e. L_t^2), from an information leakage point of view. This is in line with our observation of Section 1.3.
2. *Simulated traces.* As witnessed by the differences between the left and right parts of the figure, the information leakage of simulated traces reasonably corresponds to the one of actual traces at 1.2V, and exhibit more deviations

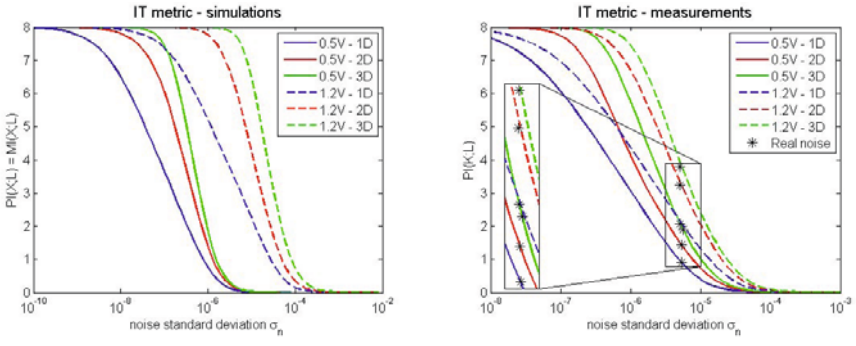


Fig. 2. Mutual information between an input X and corresponding leakage L in function of the noise std. deviation, for 1D, 2D and 3D leakages. Left: simulations. Right curves: measurements + simulated noise. Right stars: measurements.

at 0.5V (i.e. when variability increases). This can be explained by the difficulty to capture all physical effects in simulation models (including the ones related to our measurement setup). While the intuitions given by simulations are sound (e.g. decreasing the supply voltage reduces the information leakage) the numerical values they provide need to be considered with care.

In the rest of the paper, we will systematically consider averaged measurements plus simulated noise in our evaluations, since this behaves very similarly to the actual measurements while allowing modifications in noise levels[2].

2.2 A Note about the "Independent Leakage" Assumption

An important assumption found in several formal works in the area of leakage resilient cryptography is that independent computations give rise to independent leakage. Our experiments suggest that such an assumption may not hold in practice. One first reason for this, discussed in [18], is cross-talk: the current flowing in one wire of a bus may significantly influence the one of adjacent wires, both in terms of delays and power consumption. More generally, the coupling between any locally connected parts of an integrated circuit, like our S-box implementation, has an important impact in this respect. For example, the leakage traces of different chips in Figure 9 are significantly different. The main cause of these different shapes are glitches, i.e. random transitions at the gates inputs/outputs that are caused by signals arriving at different times. As these arrival times depend on all the paths of the signals before they reach a gate, glitches are a clear expression of leakage dependencies between different parts of a circuit.

In general, it is difficult to quantify the exact impact of each type of coupling that can occur within an integrated circuit. This is because only the combination of all these effects can be observed in a measurement trace. However, it is possible

[2] For each experiment, we additionally checked that simulated noise did not introduce any significant deviation from the real measurement noise, as in Figure 2.

to show that simple models that are linear combinations of the S-box input or output bits are not able to capture the full complexity of the leakage samples in our traces. The stochastic models introduced by Schindler et al. [27] are a very useful tool to quantify this claim. The principle of stochastic models is to perform a regression in order to find the function $\hat{\mathsf{L}}_t = \sum \alpha_i \cdot g_i(x)$ that will best approach the actual leakage function, with $[g_1(x), g_2(x), \ldots, g_N(x)]$ representing the basis used in the regression. If one uses the S-box output bits as base vectors, the approximated function will be linear. And by adding quadratic, cubic, ... terms in the basis, it is possible to refine the approximation. Eventually, a stochastic model using all possible terms of degree equal to or smaller than 8 has enough degrees of freedom to assign an independent value to the leakage of each S-box input, i.e. it is strictly equivalent to the exhaustive construction of 256 templates.

Note that, when evaluating the information leakage with a stochastic model using small bases, the model used by the adversary $\hat{\Pr}_{\mathtt{model}}[L|x]$ may not anymore correspond to the actual leakage pdf $\hat{\Pr}_{\mathtt{chip}}[L|x]$. This happens, e.g. if the stochastic model is not able to capture all the leakage dependencies in the traces.

The left part of Figure 3 plots the information leakage corresponding to different stochastic models. For low noise standard deviations and low degree bases, it shows that the stochastic models are not accurate, as the perceived information decreases below zero. The figure also exhibits that the impact of adding terms in the basis varies with the time samples. Again, the intuitive meaning of the non linear terms in the basis is not easy to give, as they relate to various physical effects. As illustrated in the right part of Figure 3, a combinatorial circuit connects several gates and any intermediate value may be used as an additional base vector. But our experiments at least show that, for any time sample in the traces, even late ones that are mainly influenced by the S-box output bits, various features cannot be predicted by a linear combination of those bits.

These observations have important consequences for formal works in the area of physically observable cryptography. First, they contradict the assumption in [5], where the security proof requires that different gates generate independent

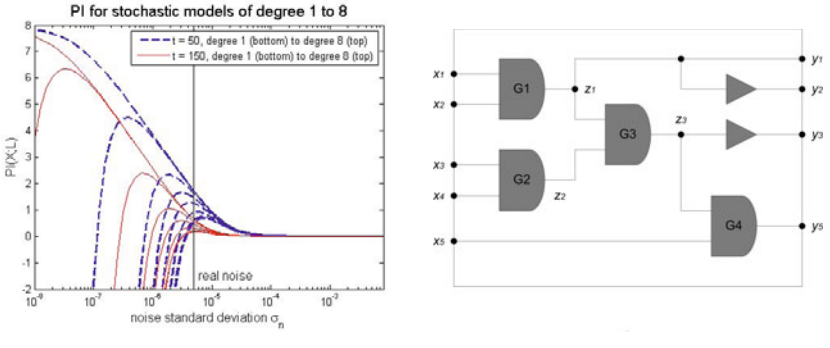


Fig. 3. Left: mutual information between an input X and corresponding leakage L, in function of the noise, obtained using stochastic models with bases containing functions of various degrees of the S-box output bits (1.2V supply). Right: gates combination.

leakage. In general, it is unlikely that this condition can hold for locally connected parts of a circuit. The integration of coupling effects (e.g. leakage functions with quadratic, cubic, ... terms) in such analyzes is an interesting scope for further research. More theoretically, our experiments suggest that the assumption in [4,19] may not always hold either. These works assume independence at a higher abstraction level, e.g. by requiring that two PRGs lead to independent leakage. In view of the importance of coupling in deep submicron technologies, fulfilling this requirement would at least require to ensure a sufficient (time or space) distance between their executions, so that local dependencies become negligible.

2.3 Realistic Scenario: Profiling and Attacking Different Chips

As inter-chip variability increases in recent CMOS technologies, the worst-case analysis in the previous section no longer corresponds to an actual attack scenario. This is because the templates profiled for one implementation may not be optimal anymore for attacking other implementations. As a consequence, we now concentrate on a more realistic situation, where one profiles and attacks different chips. The success rate of the side-channel key recovery will then essentially depend on the extent to which the templates built during profiling are sufficiently close to the actual power consumption of the target implementation. Again, this can be measured with the perceived information. But in order to build sound leakage models $\hat{\Pr}_{\mathtt{model}}[L|x]$, we first need to improve the profiling, in order to take the process variability into account. For this purpose, a natural approach is to extend the template profiling as illustrated in Figure 4. That is, classical template attacks estimate the conditional leakage distributions with $\hat{\Pr}_{\mathtt{model}}[L|x] = \mathcal{N}(l|\hat{\mu}_{x,c,N}, \hat{\sigma}^2_{x,c,N})$, where $\hat{\mu}_{x,c,N}$ (resp. $\hat{\sigma}^2_{x,c,N}$) denotes the sample mean (resp. variance) of the leakage variable, for a fixed plaintext x, chip c and a random noise N. In order to take the inter-chip variability into account, one can simply accumulate these sample means and variances, considering multiple chips rather than a single one. This means using the following estimates:

$$\hat{\Pr}_{\mathtt{model}}[L|x] = \mathcal{N}(l|\hat{\mu}_{x,C,N}, \hat{\sigma}^2_{x,C,N}),$$

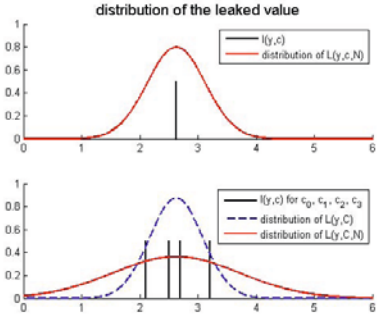


Fig. 4. Example of a template built from 1 chip (top) and 4 chips (bottom)

where $\hat{\sigma}^2_{x,C,N} = \hat{\sigma}^2_{x,c,N} + \hat{\sigma}^2_{x,C,0}$ when (additive) simulated noise is used for N. The left part of figure 5 shows the information leakage of different templates, obtained using different sets of profiling chips. It shows that variability has important consequences for the application of side-channel attacks. First, we see that profiling a large set of chips allows avoiding situations in which the perceived entropy is negative (see, e.g. the "10 vs. 1" curve). But this is at the cost of a reduced information leakage: by inferring the inter-chip variability directly into the templates, one also obtains models that are less accurate for any single chip. This can be observed by the significantly higher information curve of the worst case scenario (denoted as "1 vs. 1 (same chip)"). Second, the right part of the figure illustrates the effect of the selected time sample on the attack: the information decreases both when the input variability decreases (time sample 75 to 110) and when the chip variability increases (time sample 75 to 300).

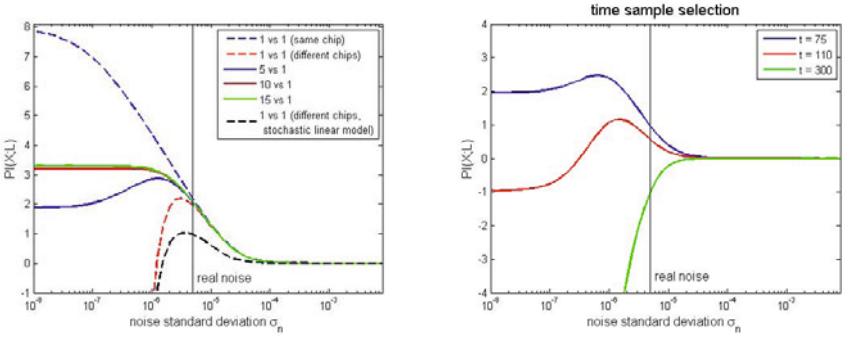


Fig. 5. Left: information theoretic analysis for various sets of learning chips (1.2V supply). Right: information theoretic analysis for various time samples (0.5V supply).

It is essential to properly understand the meaning of these different curves. What they essentially show is that modeling inter-chip variability with a straightforward extension of template attacks (as we did in this section) leads to significant information losses. Hence, it underlines the need to develop new side-channel distinguishers, that can better cope with such situations. One possible solution, discussed in the next section, is to use non-profiled distinguishers and to perform model estimation "on-the-fly", while performing the attacks.

Another important remark is that these experiments do not reduce the relevance of the worst case curve in security evaluations: perfectly profiling one chip and evaluating the perceived information in this context (i.e. the mutual information), remains useful to determine the security limits of an implementation. From a cryptographic designer point of view, the good news is that technology scaling will generally make this limit harder to reach for actual adversaries.

2.4 Model Soundness versus DPA Soundness

The previous section suggests that inter-chip variability makes the building of accurate templates a challenging task. There exist cases in which the adversary's

leakage model is not even sound, in the sense that it leads to negative perceived information values. Following [28], a leakage model is sound if the asymptotic success rate of a Bayesian adversary exploiting it in order to recover a secret target value equals one. In our present case study, the model is sound if all inputs x can be recovered thanks to their corresponding leakage.

This definition of soundness is very strict: a single inversion in the templates (*e.g.* $\hat{\mu}_{x_i,C,N} < \hat{\mu}_{x_j,C,N}$ when $\mu_{x_i,C,N} > \mu_{x_j,C,N}$) is enough for a model not to be sound. However, it is of particular interest for the application of algebraic side-channel attacks [23,24,26], in which errors in the leakage information usually makes the solving of the system of equations containing the secret key impossible. As a consequence, we now discuss solutions to obtain sound leakage models.

For this purpose, we use the notion of key class. That is, in the previous section, we always considered leakage models built for all the 256 possible S-box inputs x. However, it is also possible to build less informative models, by considering a lower number of templates. Formally, we define a function $\delta : \mathcal{X} \rightarrow \mathcal{S}$ that maps each input value x to a key class $s = \delta(x)$. The number of key classes can be equal (in the case of an identity mapping $\delta(x) = x$) or lower than the number of possible inputs: $|\mathcal{S}| \leq |\mathcal{X}|$. The mutual information between a mapping variable $S = \delta(X)$ and an input variable X, is defined as:

$$\mathrm{I}(X;S) = \mathrm{H}[S] - \mathrm{H}[S|X] = \mathrm{H}[S].$$

Given a key class variable S, it is then possible to check the soundness of the corresponding leakage model with the conditional entropy matrix defined in [28]:

$$\begin{aligned}\hat{\mathbf{H}}_{s,s^*} &= -\sum_{l \in \mathcal{L}} \hat{\Pr}_{\mathtt{chip}}[l|s] \log_2 \hat{\Pr}_{\mathtt{model}}[s^*|l], \\ &= \begin{pmatrix} \hat{h}_{1,1} & \hat{h}_{1,2} & \dots & \hat{h}_{1,|\mathcal{S}|} \\ \hat{h}_{2,2} & \hat{h}_{2,2} & \dots & \hat{h}_{2,|\mathcal{S}|} \\ \dots & \dots & \dots & \dots \\ \hat{h}_{|\mathcal{S}|,1} & \hat{h}_{|\mathcal{S}|,2} & \dots & \hat{h}_{|\mathcal{S}|,|\mathcal{S}|} \end{pmatrix},\end{aligned}$$

where s and s^* respectively denote the correct key class and a key class candidate. The model is sound if and only if the minimum value for each line of the matrix is the diagonal value $\hat{h}_{i,i}$. Having defined these tools, we can study the tradeoff between the informativeness of a key class $\mathrm{I}(X;S)$ and the soundness of the corresponding leakage model $\hat{\Pr}_{\mathtt{model}}[L|s]$. For this purpose, we considered consecutive key classes with $|\mathcal{S}| = 256, 255, \dots 1$, with the mapping function δ grouping close leakages, and the templates built from a set of 5 chips, as illustrated in the left part of Figure 6. The right part of the figure then shows how the informativeness and soundness of these successive key classes evolves with the error probability of the template attack that we define as:

$$\Pr_{\mathtt{error}} = \Pr[\underset{s^*}{\mathrm{argmax}}\ \hat{\Pr}_{\mathtt{model}}[s^*|l] \neq s]. \tag{4}$$

It illustrates that it is possible to build a key class with 6 possible values for which the model $\hat{\Pr}_{\mathtt{model}}[L|s]$ is sound. Such a key class could be directly used in

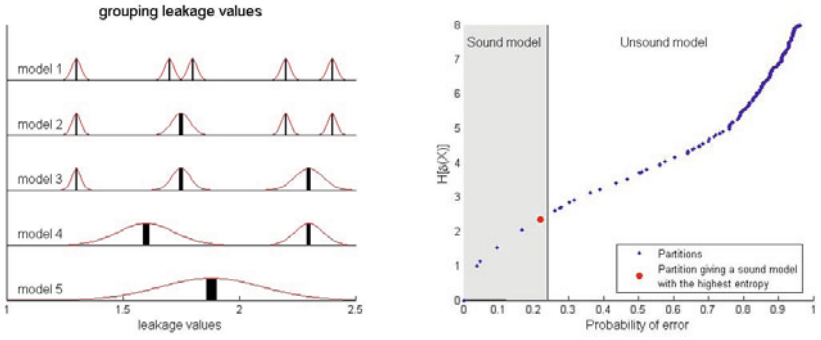


Fig. 6. Left: building more robust / less informative models. Right: maximum information provided by a model in function of the classification error rate (1.2V supply).

an algebraic side-channel attack. We note, however, that the same comment as in the previous section applies. Namely, classical template attacks are probably not the best solution to build sound and informative leakage models.

To conclude this section, we finally mention that model soundness is a necessary condition for successful algebraic side-channel attacks. But for standard DPA types of attacks [15], it is only a sufficient condition. That is, standard DPA attacks will generally exploit the leakage corresponding to the execution of the S-box for several plaintexts, i.e. $\mathsf{S}(x \oplus k)$, in order to recover the secret key k. Hence, from an information theoretic point of view, the relevant metric is no more $\hat{\mathrm{PI}}(X; L)$ but $\hat{\mathrm{PI}}(K; X, L)$, with conditional entropy matrix:

$$\hat{\mathbf{H}}_{k,k^*} = -\sum_{x \in \mathcal{X}} \Pr[x] \sum_{l \in \mathcal{L}} \hat{\Pr}_{\mathtt{chip}}[l|x,k] \log_2 \hat{\Pr}_{\mathtt{model}}[k^*|l,x],$$

where k and k^* denote the correct key and a key candidate. As each line of the matrix is computed by averaging over 256 possible inputs x, even some misclassified traces do not always prevent a successful DPA. In other words, DPA-soundness, corresponding to a matrix $\hat{\mathbf{H}}_{k,k^*}$ with minimum diagonal, is a much weaker requirement than model soundness. In our setting, even the identity mapping $\delta(x) = x$ gave rise to successful DPA attacks using templates. The analysis of this scenario will be investigated in the next section.

3 Security Analysis

The previous section provided an extensive evaluation of the information leakage of an AES S-box implemented in a 65-nanometer CMOS technology. It shows that inter-chip variability makes the straightforward application of profiled attacks (such as using templates, or stochastic models) less efficient than when variability can be neglected. In this section, we perform the second part of the evaluation framework in [28], i.e. security analysis. For this purpose, we analyze

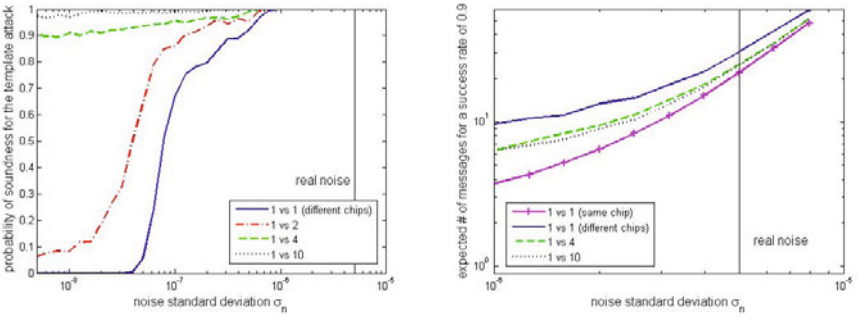


Fig. 7. Left: Probability of soundness for a template attack. Right: Expected number of messages to reach a 0.9 success rate for a sound template attack (1.2V supply).

the success rates of various distinguishers in a standard DPA setting, in order to determine the impact of variability in this context. Namely, we performed:

- Template attacks, using exactly the profiling described in Section 2.3.
- Correlation attacks [2] with a Hamming weight leakage model[3].
- Mutual Information Analysis (MIA) attacks, using a Hamming weight leakage model and an identity leakage model (corresponding to 7 out of 8 S-box output bits). Our implementation of MIA followed the guidelines given in [8]: we used histogram-based pdf estimation, with 32 (linearly-spaced) bins, which allowed us to deal with the weak accuracy of our leakage models.
- Non-profiled attacks using stochastic models generated "on-the-fly" , with the linear bases described in Section 2.2. That is, we used exactly the profiling techniques proposed in [27], but this profiling was done for each key candidate separately. The attack then proceeds as carefully described in [12]: each time the adversary gains a new trace, he repeats the profiling and tests his key dependent models directly on the set of available traces.

Figure 7 illustrates the effect of variability on template attacks, for a 1.2V supply voltage. Its left part shows that attacks may not work at all, when the profiling done for a chip is used to attack a significantly different chip. But as discussed in the previous section, one reaches DPA-soundness easily, by profiling on more than 4 chips. The right part of the figure shows that the number of traces to attack is low once a sound model is available, and lower bounded by the worst case curve corresponding to a perfect model. Moving to 0.5V supply would lead to similar conclusions, with both curves slightly translated on the right.

Figure 8 shows the results of various non-profiled attacks. Its right part contains results of a correlation attack against each of our 20 chips. It underlines that the Hamming weight leakage model only allows to reach a 100% success rate for a few of these chips. Hence, it cannot be used to evaluate the security of a

[3] This is equivalent to a Hamming distance model, corresponding to the transitions between a pre-charge of the S-box input to zero and its evaluation on input x.

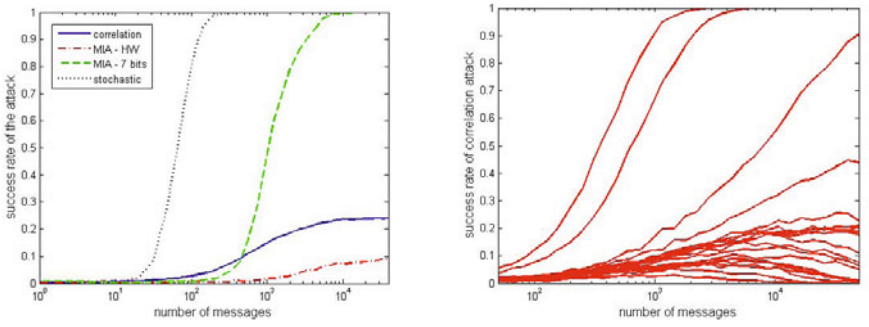


Fig. 8. Left: success rates of a various non profiled attacks averaged over 20 chips. Right: success rates of the correlation attacks for each of the 20 chips (1.2V supply).

cryptographic implementation in this case. The left part of the figure illustrates the average success rates (over the 20 chips) of our different non-profiled attacks. It suggests that attacks performing "on-the-fly" model estimation, such as using stochastic models or MIA, are a promising approach for dealing with variability.

4 Conclusions and Open Problems

Process variability in nanoscale devices raises new challenges for the theory and practice of side-channel attacks. Experiments performed on 20 prototype chips show that former DPA attacks are not perfectly adequate to evaluate the security of an implementation in this context. In the absence of variability, adversaries could first profile (or assume) a leakage model, and then exploit this model in an online attack. With the increase of process variability, it becomes necessary to infer the correct leakage model for each target implementation. The deep integration of recent microelectronic circuits also increases the coupling between their interconnected parts, with consequences on the "independent leakage" assumption that is frequently required in formal works on leakage resilience. Hence, developing new techniques to deal with this new attacks scenario is essential, in order to avoid overestimating the security levels of actual products.

Acknowledgements. M. Renauld is a PhD student funded by the Walloon region SCEPTIC project. F.-X. Standaert is an associate researcher of the Belgian fund for scientific research (FNRS - F.R.S). N. Veyrat-Charvillon is a postdoctoral researcher funded by the Walloon region SCEPTIC project. D. Kamel is a PhD student funded by the Walloon region E.USER project.

References

1. Bowman, K.A., Tang, X., Eble, J.C., Menldl, J.D.: Impact of extrinsic and intrinsic parameter fluctuations on CMOS circuit performance. IEEE Journal of Solid State Circuits 35(8), 1186–1193 (2002)

2. Brier, É., Clavier, C., Olivier, F.: Correlation Power Analysis with a Leakage Model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
3. 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)
4. Dziembowski, S., Pietrzak, K.: Leakage-Resilient Cryptography. In: Proceedings of FOCS 2008, Philadelphia, Pennsylvania, USA, pp. 293–302 (October 2008)
5. Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting Circuits from Leakage: the Computationally-Bounded and Noisy Cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
6. 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)
7. Ghosh, S., Roy, K.: Parameter Variation Tolerance and Error Resiliency: New Design Paragigm for the Nanoscale Era. The Proceedings of the IEEE 98(10), 1718–1751 (2010)
8. Gierlichs, B., Batina, L., Tuyls, P., Preneel, B.: Mutual Information Analysis. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 426–442. Springer, Heidelberg (2008)
9. Kamel, D., Standaert, F.-X., Flandre, D.: Scaling Trends of the AES S-box Lower Power Consumption in 130 and 65 nm CMOS Technology Nodes. In: The Proceedings of ISCAS 2009, Taipei, Taiwan (May 2009)
10. Kamel, D., Hocquet, C., Standaert, F.-X., Flandre, D., Bol, D.: Glicth-Induced Within Die Variations of Dynamic Energy in Voltage-Scaled Nano-CMOS Circuits. In: The Proceedings of ESSCIRC 2010, Seville, Spain (September 2010)
11. 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)
12. Lemke-Rust, K.: Models and Algorithms for Physical Cryptanalysis, PhD Thesis, Ruhr University Bochum, Germany (June 2007)
13. Lin, L., Burleson, W.: Analysis and Mitigation of Process Variation Impacts on Power-Attack Tolerance. In: The Proceedings of DAC 2009, San Francisco, CA, USA, pp. 238–243 (July 2009)
14. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks. Springer, Heidelberg (2007)
15. Mangard, S., Oswald, E., Standaert, F.-X.: One for All - All for One: Unifying Standard DPA Attacks, Cryptology ePrint Archive: Report 2009/449
16. Mentens, N., Batina, L., Preneel, B., Verbauwhede, I.: A Systematic Evaluation of Compact Hardware Implementations for the Rijndael SBOX. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 323–333. Springer, Heidelberg (2005)
17. Nassif, S., Bernstein, K., Frank, D.J., Gattiker, A., Haensch, W., Ji, B.L., Nowak, E., Pearson, D., Rohrer, N.J.: High Performance CMOS Variability in the 65nm Regime and Beyond. In: The Proceedings of IEDM 2007, Washington DC, USA, pp. 569–571 (December 2007)
18. Nieuwland, A.K., Katoch, A., Meijer, M.: Reducing Cross-Talk Induced Power Consumption and Delay. In: Macii, E., Paliouras, V., Koufopavlou, O. (eds.) PATMOS 2004. LNCS, vol. 3254, pp. 179–188. Springer, Heidelberg (2004)
19. Pietrzak, K.: A Leakage-Resilient Mode of Operation. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 462–482. Springer, Heidelberg (2009)

20. Pietrzak, K.: Provable Security for Physical Cryptography. In: The Proceedings of WEWORC 2009, Graz, Austria (July 2009)
21. Quisquater, J.-J., Samyde, D.: ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
22. Regazzoni, F., Cevrero, A., Standaert, F.-X., Badel, S., Kluter, T., Brisk, P., Leblebici, Y., Ienne, P.: A Design Flow and Evaluation Framework for DPA-Resistant Instruction Set Extensions. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 205–219. Springer, Heidelberg (2009)
23. Renauld, M., Standaert, F.-X.: Algebraic Side-Channel Attacks. In: Bao, F., Yung, M., Lin, D., Jing, J. (eds.) Inscrypt 2009. LNCS, vol. 6151, pp. 393–410. Springer, Heidelberg (2010)
24. Renauld, M., Standaert, F.-X., Veyrat-Charvillon, N.: Algebraic Side-Channel Attacks on the AES: Why Time also Matters in DPA. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 97–111. Springer, Heidelberg (2009)
25. Roy, K., Mukhopadhyay, S., Mahmoodi-Meimand, H.: Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. Proceedings of the IEEE 91(2), 305–327 (2003)
26. Oren, Y., Kirschbaum, M., Popp, T., Wool, A.: Algebraic Side-Channel Analysis in the Presence of Errors. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 428–442. Springer, Heidelberg (2010)
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. Standaert, F.-X., Malkin, T.G., 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)
29. 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)
30. Standaert, F.-X., Veyrat-Charvillon, N., Oswald, E., Gierlichs, B., Medwed, M., Kasper, M., Mangard, S.: The World is Not Enough: Another Look on Second-Order DPA. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 112–129. Springer, Heidelberg (2010)
31. Tiri, K., Verbauwhede, I.: Securing encryption algorithms against dpa at the logic level: Next generation smart card technology. In: Walter, C.D., Koç, Ç.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 125–136. Springer, Heidelberg (2003)

A Exemplary Leakage Traces

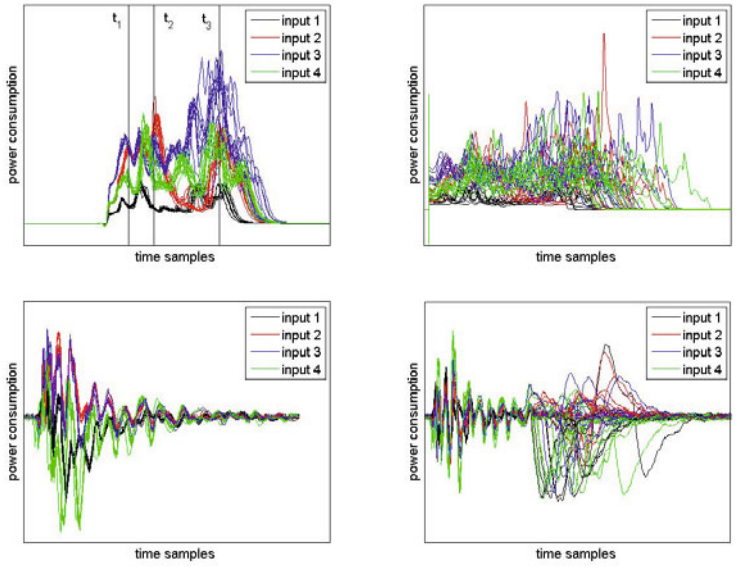


Fig. 9. Illustrative noise-free leakage traces corresponding to 4 inputs and 10 chips. Left: 1.2V supply, Right: 0.5V supply / Up: simulations, Down: actual measurements.

B Preprocessing of the Traces

Side-channel attacks exploit information about the internal state of a computing device that is leaked, e.g. through power consumption traces. These power traces also contain some noise, either due to unpredictable physical effects, or to other forms of perturbations such as measurement artifacts (outliers) or parasitic signals (interference). The influence of this second source of noise can sometimes be reduced by processing the power traces prior to the actual attack.

The top curve in the left part of Figure 10 illustrates the autocorrelation of a power trace. It measures the linear correlation between the trace and its shift by τ time samples. The autocorrelation function shows two mixed components: regularly spaced peaks (around $\tau = 1000, 2000$, etc.) and some periodic sinusoidal component with period close to 2000. Roughly speaking, the correlation peaks which correspond to successive clock cycles of the chip, correspond to the useful signal. By contrast, the periodic sinusoidal component is a parasitic that can be filtered. An example of filtered trace is given in the bottom curve of the left part of Figure 10. And the impact of this preprocessing on the distribution of the residual noise (estimated with histograms) is in the right part of the figure. It clearly illustrates the gain in terms of noise standard deviation.

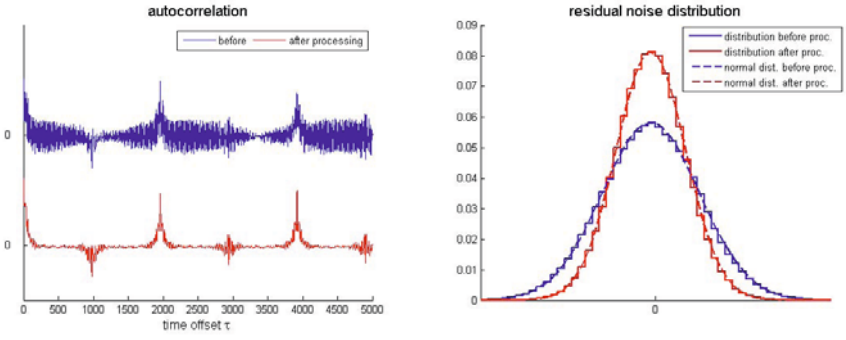


Fig. 10. Left: autocorrelation of the signal before (top) and after (bottom) preprocessing of the traces. Right: residual noise distribution with/without preprocessing.

C PCA and LDA Eigenvectors

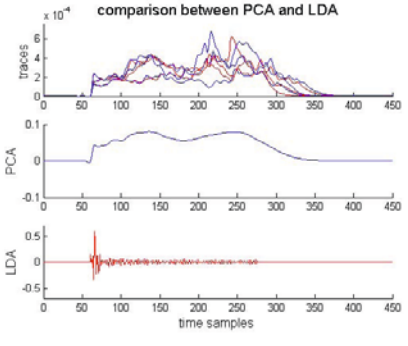


Fig. 11. PCA (middle) and LDA (below) applied to simulated traces at 1.2V (above)

Implementing Gentry's Fully-Homomorphic Encryption Scheme

Craig Gentry* and Shai Halevi*

IBM Research

Abstract. We describe a working implementation of a variant of Gentry's fully homomorphic encryption scheme (STOC 2009), similar to the variant used in an earlier implementation effort by Smart and Vercauteren (PKC 2010). Smart and Vercauteren implemented the underlying "somewhat homomorphic" scheme, but were not able to implement the bootstrapping functionality that is needed to get the complete scheme to work. We show a number of optimizations that allow us to implement all aspects of the scheme, including the bootstrapping functionality.

Our main optimization is a key-generation method for the underlying somewhat homomorphic encryption, that does not require full polynomial inversion. This reduces the asymptotic complexity from $\tilde{O}(n^{2.5})$ to $\tilde{O}(n^{1.5})$ when working with dimension-n lattices (and practically reducing the time from many hours/days to a few seconds/minutes). Other optimizations include a batching technique for encryption, a careful analysis of the degree of the decryption polynomial, and some space/time trade-offs for the fully-homomorphic scheme.

We tested our implementation with lattices of several dimensions, corresponding to several security levels. From a "toy" setting in dimension 512, to "small," "medium," and "large" settings in dimensions 2048, 8192, and 32768, respectively. The public-key size ranges in size from 70 Megabytes for the "small" setting to 2.3 Gigabytes for the "large" setting. The time to run one bootstrapping operation (on a 1-CPU 64-bit machine with large memory) ranges from 30 seconds for the "small" setting to 30 minutes for the "large" setting.

1 Introduction

Encryption schemes that support operations on encrypted data (aka homomorphic encryption) have a very wide range of applications in cryptography. This concept was introduced by Rivest et al. shortly after the discovery of public key cryptography [12], and many known public-key cryptosystems support either addition or multiplication of encrypted data. However, supporting both at the same time seems harder, and until very recently all the attempts at constructing so-called "*fully homomorphic*" encryption turned out to be insecure.

In 2009, Gentry described the first plausible construction of a fully homomorphic cryptosystem [3]. Gentry's construction consists of several steps: He first

* Supported by DARPA grant DARPA-BAA 10-81.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 129–148, 2011.

constructed a "*somewhat homomorphic*" scheme that supports evaluating low-degree polynomials on the encrypted data, next he needed to "*squash*" the decryption procedure so that it can be expressed as a low-degree polynomial which is supported by the scheme, and finally he applied a "*bootstrapping*" transformation to obtain a fully homomorphic scheme. The crucial point in this process is to obtain a scheme that can evaluate polynomials of high-enough degree, and at the same time has decryption procedure that can be expressed as a polynomial of low-enough degree. Once the degree of polynomials that can be evaluated by the scheme exceeds the degree of the decryption polynomial (times two), the scheme is called "*bootstrappable*" and it can then be converted into a fully homomorphic scheme.

Toward a bootstrappable scheme, Gentry described in [3] a somewhat homomorphic scheme, which is roughly a GGH-type scheme [6,8] over ideal lattices. Gentry later proved [4] that with an appropriate key-generation procedure, the security of that scheme can be (quantumly) reduced to the worst-case hardness of some lattice problems in ideal lattices.

This somewhat homomorphic scheme is not yet bootstrappable, so Gentry described in [3] a transformation to squash the decryption procedure, reducing the degree of the decryption polynomial. This is done by adding to the public key an additional hint about the secret key, in the form of a "*sparse subset-sum*" problem (SSSP). Namely the public key is augmented with a big set of vectors, such that there exists a very sparse subset of them that adds up to the secret key. A ciphertext of the underlying scheme can be "*post-processed*" using this additional hint, and the post-processed ciphertext can be decrypted with a low-degree polynomial, thus obtaining a bootstrappable scheme.

Stehlé and Steinfeld described in [14] two optimizations to Gentry's scheme, one that reduces the number of vectors in the SSSP instance, and another that can be used to reduce the degree of the decryption polynomial (at the expense of introducing a small probability of decryption errors). We mention that in our implementation we use the first optimization but not the second[1]. Some improvements to Gentry's key-generation procedure were discussed in [9].

1.1 The Smart-Vercauteren Implementation

The first attempt to implement Gentry's scheme was made in 2010 by Smart and Vercauteren [13]. They chose to implement a variant of the scheme using "*principal-ideal lattices*" of prime determinant. Such lattices can be represented implicitly by just two integers (regardless of their dimension), and moreover Smart and Vercauteren described a decryption method where the secret key is represented by a single integer. Smart and Vercauteren were able to implement the underlying somewhat homomorphic scheme, but they were not able to support large enough parameters to make Gentry's squashing technique go through. As a result they could not obtain a bootstrappable scheme or a fully homomorphic scheme.

[1] The reason we do not use the second optimization is that the decryption error probability is too high for our parameter settings.

One obstacle in the Smart-Vercauteren implementation was the complexity of key generation for the somewhat homomorphic scheme: For one thing, they must generate very many candidates before they find one whose determinant is prime. (One may need to try as many as $n^{1.5}$ candidates when working with lattices in dimension n.) And even after finding one, the complexity of computing the secret key that corresponds to this lattice is at least $\tilde{\Theta}(n^{2.5})$ for lattices in dimension n. For both of these reasons, they were not able to generate keys in dimensions $n > 2048$.

Moreover, Smart and Vercauteren estimated that the squashed decryption polynomial will have degree of a few hundreds, and that to support this procedure with their parameters they need to use lattices of dimension at least $n = 2^{27} (\approx 1.3 \times 10^8)$, which is well beyond the capabilities of the key-generation procedure.

1.2 Our Implementation

We continue in the same direction of the Smart-Vercauteren implementation and describe optimizations that allow us to implement also the squashing part, thereby obtaining a bootstrappable scheme and a fully homomorphic scheme.

For key-generation, we present a new faster algorithm for computing the secret key, and also eliminate the requirement that the determinant of the lattice be prime. We also present many simplifications and optimizations for the squashed decryption procedure, and as a result our decryption polynomial has degree only fifteen. Finally, our choice of parameters is somewhat more aggressive than Smart and Vercauteren (which we complement by analyzing the complexity of known attacks).

Differently from [13], we decouple the dimension n from the size of the integers that we choose during key generation. Decoupling these two parameters lets us decouple functionality from security. Namely, we can obtain bootstrappable schemes in any given dimension, but of course the schemes in low dimensions will not be secure. Our (rather crude) analysis suggests that the scheme may be practically secure at dimension $n = 2^{13}$ or $n = 2^{15}$, and we put this analysis to the test by publishing a few challenges in dimensions from 512 up to 2^{15}.

1.3 Organization

We give some background in Section 2, and then describe our implementation of the underlying "somewhat homomorphic" encryption scheme in Sections 3 through 7. A description of our optimizations that are specific to the bootstrapping functionality appears in the full version of this report [5].

2 Background

Notations. Throughout this report we use '$\cdot$' to denote scalar multiplication and '$\times$' to denote any other type of multiplication. For integers z, d, we denote the reduction of z modulo d by either $[z]_d$ or $\langle z \rangle_d$. We use $[z]_d$ when the operation

maps integers to the interval $[-d/2,\ d/2)$, and use $\langle z \rangle_d$ when the operation maps integers to the interval $[0, d)$. We use the generic "$z \bmod d$" when the specific interval does not matter (e.g., mod 2). For example we have $[13]_5 = -2$ vs. $\langle 13 \rangle_5 = 3$, but $[9]_7 = \langle 9 \rangle_7 = 2$.

For a rational number q, we denote by $\lceil q \rfloor$ the rounding of q to the nearest integer, and by $[q]$ we denote the distance between q and the nearest integer. That is, if $q = \frac{a}{b}$ then $[q] \stackrel{\text{def}}{=} \frac{[a]_b}{b}$ and $\lceil q \rfloor \stackrel{\text{def}}{=} q - [q]$. For example, $\lceil \frac{13}{5} \rfloor = 3$ and $[\frac{13}{5}] = \frac{-2}{5}$. These notations are extended to vectors in the natural way: for example if $\boldsymbol{q} = \langle q_0, q_1, \ldots, q_{n-1} \rangle$ is a rational vector then rounding is done coordinate-wise, $\lceil \boldsymbol{q} \rfloor = \langle \lceil q_0 \rfloor, \lceil q_1 \rfloor, \ldots, \lceil q_{n-1} \rfloor \rangle$.

2.1 Lattices

A full-rank n-dimensional *lattice* is a discrete subgroup of $\mathbb{R}^n$, concretely represented as the set of all integer linear combinations of some *basis* $B = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) \in \mathbb{R}^n$ of linearly independent vectors. Viewing the vectors $\boldsymbol{b}_i$ as the rows of a matrix $B \in \mathbb{R}^{n \times n}$, we have: $L = \mathcal{L}(B) = \{\boldsymbol{y} \times B \ :\ \boldsymbol{y} \in \mathbb{Z}^n\}$.

Every lattice (of dimension $n > 1$) has an infinite number of lattice bases. If B_1 and B_2 are two lattice bases of L, then there is some unimodular matrix U (i.e., U has integer entries and $\det(U) = \pm 1$) satisfying $B_1 = U \times B_2$. Since U is unimodular, $|\det(B_i)|$ is invariant for different bases of L, and we may refer to it as $\det(L)$. This value is precisely the size of the quotient group $\mathbb{Z}^n/L$ if L is an integer lattice. To basis B of lattice L we associate the half-open parallelepiped $\mathcal{P}(B) \leftarrow \{\sum_{i=1}^{n} x_i \boldsymbol{b}_i : x_i \in [-1/2, 1/2)\}$. The volume of $\mathcal{P}(B)$ is precisely $\det(L)$.

For $\boldsymbol{c} \in \mathbb{R}^n$ and basis B of L, we use $\boldsymbol{c} \bmod B$ to denote the unique vector $\boldsymbol{c'} \in \mathcal{P}(B)$ such that $\boldsymbol{c} - \boldsymbol{c'} \in L$. Given $\boldsymbol{c}$ and B, $\boldsymbol{c} \bmod B$ can be computed efficiently as $\boldsymbol{c} - \lfloor \boldsymbol{c} \times B^{-1} \rceil \times B = [\boldsymbol{c} \times B^{-1}] \times B$. (Recall that $\lfloor \cdot \rceil$ means rounding to the nearest integer and $[\cdot]$ is the fractional part.)

Every full-rank lattice has a unique Hermite normal form (HNF) basis where $b_{i,j} = 0$ for all $i < j$ (lower-triangular), $b_{j,j} > 0$ for all j, and for all $i > j$ $b_{i,j} \in [-b_{j,j}/2, +b_{j,j}/2)$. Given any basis B of L, one can compute $\mathrm{HNF}(L)$ efficiently via Gaussian elimination. The HNF is in some sense the "least revealing" basis of L, and thus typically serves as the public key representation of the lattice [8].

Short vectors and Bounded Distance Decoding. The length of the shortest nonzero vector in a lattice L is denoted $\lambda_1(L)$, and Minkowski's theorem says that for any n-dimensional lattice L ($n > 1$) we have $\lambda_1(L) < \sqrt{n} \cdot \det(L)^{1/n}$. Heuristically, for random lattices the quantity $\det(L)^{1/n}$ serves as a threshold: for $t \ll \det(L)^{1/n}$ we don't expect to find any nonzero vectors in L of size t, but for $t \gg \det(L)^{1/n}$ we expect to find exponentially many vectors in L of size t.

In the "*bounded distance decoding*" problem (BDDP), one is given a basis B of some lattice L, and a vector $\boldsymbol{c}$ that is very close to some lattice point of L, and the goal is to find the point in L nearest to $\boldsymbol{c}$. In the promise problem γ-BDDP, we have a parameter $\gamma > 1$ and the promise that $\mathrm{dist}(L, \boldsymbol{c}) \stackrel{\text{def}}{=} \min_{\boldsymbol{v} \in L}\{\|\boldsymbol{c} - \boldsymbol{v}\|\} \leq \det(L)^{1/n}/\gamma$. (BDDP is often defined with respect to λ_1 rather than with respect to $\det(L)^{1/n}$, but the current definition is more convenient in our case.)

Gama and Nguyen conducted extensive experiments with lattices in dimensions 100-400 [2], and concluded that for those dimensions it is feasible to solve γ-BDDP when $\gamma > 1.01^n \approx 2^{n/70}$. More generally, the best algorithms for solving the γ-BDDP in n-dimensional lattices take time exponential in $n/\log\gamma$. Specifically, currently known algorithms can solve dimension-n γ-BDDP in time 2^k up to $\gamma = 2^{\frac{\mu n}{k/\log k}}$, where μ is a parameter that depends on the exact details of the algorithm. (Extrapolating from the Gama-Nguyen experiments, we expect something like $\mu \in [0.1, 0.2]$.)

2.2 Ideal Lattices

Let $f(x)$ be an integer monic irreducible polynomial of degree n. In this paper, we use $f(x) = x^n + 1$, where n is a power of 2. Let R be the ring of integer polynomials modulo $f(x)$, $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/(f(x))$. Each element of R is a polynomial of degree at most $n-1$, and thus is associated to a coefficient vector in $\mathbb{Z}^n$. This way, we can view each element of R as being both a polynomial and a vector. For $\boldsymbol{v}(x)$, we let $\|\boldsymbol{v}\|$ be the Euclidean norm of its coefficient vector. For every ring R, there is an associated expansion factor $\gamma_{\mathsf{Mult}}(R)$ such that $\|\boldsymbol{u} \times \boldsymbol{v}\| \leq \gamma_{\mathsf{Mult}}(R) \cdot \|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|$, where $\times$ denotes multiplication in the ring. When $f(x) = x^n + 1$, $\gamma_{\mathsf{Mult}}(R)$ is $\sqrt{n}$. However, for "random vectors" $\boldsymbol{u}, \boldsymbol{v}$ the expansion factor is typically much smaller, and our experiments suggest that we typically have $\|\boldsymbol{u} \times \boldsymbol{v}\| \approx \|\boldsymbol{u}\| \cdot \|\boldsymbol{v}\|$.

Let I be an ideal of R – that is, a subset of R that is closed under addition and multiplication by elements of R. Since I is additively closed, the coefficient vectors associated to elements of I form a *lattice*. We call I an *ideal lattice* to emphasize this object's dual nature as an algebraic ideal and a lattice[2]. Ideals have additive structure as lattices, but they also have multiplicative structure. The *product* IJ of two ideals I and J is the additive closure of the set $\{\boldsymbol{v} \times \boldsymbol{w} : \boldsymbol{v} \in I, \boldsymbol{w} \in J\}$, where '$\times$' is ring multiplication. To simplify things, we will use *principal* ideals of R – i.e., ideals with a single generator. The ideal $(\boldsymbol{v})$ generated by $\boldsymbol{v} \in R$ corresponds to the lattice generated by the vectors $\{\boldsymbol{v}_i \stackrel{\text{def}}{=} \boldsymbol{v} \times x^i \bmod f(x) : i \in [0, n-1]\}$; we call this the *rotation basis* of the ideal lattice $(\boldsymbol{v})$.

Let K be a field containing the ring R (in our case $K = \mathbb{Q}[x]/(f(x))$). The *inverse* of an ideal $I \subseteq R$ is $I^{-1} = \{\boldsymbol{w} \in K : \forall \boldsymbol{v} \in I, \boldsymbol{v} \times \boldsymbol{w} \in R\}$. The inverse of a principal ideal $(\boldsymbol{v})$ is given by $(\boldsymbol{v}^{-1})$, where the inverse $\boldsymbol{v}^{-1}$ is taken in the field K.

2.3 GGH-Type Cryptosystems

We briefly recall Micciancio's "cleaned-up version" of GGH cryptosystems [6,8]. The secret and public keys are "good" and "bad" bases of some lattice L. More specifically, the key-holder generates a good basis by choosing B_{sk} to be a basis of

[2] Alternative representations of an ideal lattice are possible – e.g., see [11,7].

short, "nearly orthogonal" vectors. Then it sets the public key to be the Hermite normal form of the same lattice, $B_{pk} \stackrel{\text{def}}{=} \text{HNF}(\mathcal{L}(B_{sk}))$.

A ciphertext in a GGH-type cryptosystem is a vector $\boldsymbol{c}$ close to the lattice $\mathcal{L}(B_{pk})$, and the message which is encrypted in this ciphertext is somehow embedded in the distance from $\boldsymbol{c}$ to the nearest lattice vector. To encrypt a message m, the sender chooses a short "error vector" $\boldsymbol{e}$ that encodes m, and then computes the ciphertext as $\boldsymbol{c} \leftarrow \boldsymbol{e} \bmod B_{pk}$. Note that if $\boldsymbol{e}$ is short enough (i.e., less than $\lambda_1(L)/2$), then it is indeed the distance between $\boldsymbol{c}$ and the nearest lattice point.

To decrypt, the key-holder uses its "good" basis B_{sk} to recover $\boldsymbol{e}$ by setting $\boldsymbol{e} \leftarrow \boldsymbol{c} \bmod B_{sk}$, and then recovers m from $\boldsymbol{e}$. The reason decryption works is that, if the parameters are chosen correctly, then the parallelepiped $\mathcal{P}(B_{sk})$ of the secret key will be a "plump" parallelepiped that contains a sphere of radius bigger than $\|\boldsymbol{e}\|$, so that $\boldsymbol{e}$ is the point inside $\mathcal{P}(B_{sk})$ that equals $\boldsymbol{c}$ modulo L. On the other hand, the parallelepiped $\mathcal{P}(B_{pk})$ of the public key will be very skewed, and will not contain a sphere of large radius, making it useless for solving BDDP.

2.4 Gentry's Somewhat-Homomorphic Cryptosystem

Gentry's somewhat homomorphic encryption scheme [3] can be seen as a GGH-type scheme over ideal lattices. The public key consists of a "bad" basis B_{pk} of an ideal lattice J, along with some basis B_I of a "small" ideal I (which is used to embed messages into the error vectors). For example, the small ideal I can be taken to be $I = (2)$, the set of vectors with all even coefficients.

A ciphertext in Gentry's scheme is a vector close to a J-point, with the message being embedded in the distance to the nearest lattice point. More specifically, the plaintext space is $\{0,1\}$, which is embedded in $R/I = \{0,1\}^n$ by encoding 0 as 0^n and 1 as $0^{n-1}1$. For an encoded bit $\boldsymbol{m} \in \{0,1\}^n$ we set $\boldsymbol{e} = 2\boldsymbol{r} + \boldsymbol{m}$ for a random small vector $\boldsymbol{r}$, and then output the ciphertext $\boldsymbol{c} \leftarrow \boldsymbol{e} \bmod B_{pk}$.

The secret key in Gentry's scheme (that plays the role of the "good basis" of J) is just a short vector $\boldsymbol{w} \in J^{-1}$. Decryption involves computing the fractional part $[\boldsymbol{w} \times \boldsymbol{c}]$. Since $\boldsymbol{c} = \boldsymbol{j} + \boldsymbol{e}$ for some $j \in J$, then $\boldsymbol{w} \times \boldsymbol{c} = \boldsymbol{w} \times \boldsymbol{j} + \boldsymbol{w} \times \boldsymbol{e}$. But $\boldsymbol{w} \times \boldsymbol{j}$ is in R and thus an integer vector, so $\boldsymbol{w} \times \boldsymbol{c}$ and $\boldsymbol{w} \times \boldsymbol{e}$ have the same fractional part, $[\boldsymbol{w} \times \boldsymbol{c}] = [\boldsymbol{w} \times \boldsymbol{e}]$. If $\boldsymbol{w}$ and $\boldsymbol{e}$ are short enough – in particular, if we have the guarantee that all of the coefficients of $\boldsymbol{w} \times \boldsymbol{e}$ have magnitude less than $1/2$ – then $[\boldsymbol{w} \times \boldsymbol{e}]$ equals $\boldsymbol{w} \times \boldsymbol{e}$ exactly. From $\boldsymbol{w} \times \boldsymbol{e}$, the decryptor can multiply by $\boldsymbol{w}^{-1}$ to recover $\boldsymbol{e}$, and then recover $\boldsymbol{m} \leftarrow \boldsymbol{e} \bmod 2$. The actual decryption procedure from [3] is slightly different, however. Specifically, $\boldsymbol{w}$ is "tweaked" so that decryption can be implemented as $\boldsymbol{m} \leftarrow \boldsymbol{c} - [\boldsymbol{w} \times \boldsymbol{c}] \bmod 2$ (when $I = (2)$).

The reason that this scheme is somewhat homomorphic is that for two ciphertexts $\boldsymbol{c}_1 = \boldsymbol{j}_1 + \boldsymbol{e}_1$ and $\boldsymbol{c}_2 = \boldsymbol{j}_2 + \boldsymbol{e}_2$, their sum is $\boldsymbol{j}_3 + \boldsymbol{e}_3$ where $\boldsymbol{j}_3 = \boldsymbol{j}_1 + \boldsymbol{j}_2 \in J$ and $\boldsymbol{e}_3 = \boldsymbol{e}_1 + \boldsymbol{e}_2$ is small. Similarly, their product is $\boldsymbol{j}_4 + \boldsymbol{e}_4$ where $\boldsymbol{j}_4 = \boldsymbol{j}_1 \times (\boldsymbol{j}_2 + \boldsymbol{e}_2) + \boldsymbol{e}_1 \times \boldsymbol{j}_2 \in J$ and $\boldsymbol{e}_4 = \boldsymbol{e}_1 \times \boldsymbol{e}_2$ is still small. If fresh encrypted ciphertexts are very very close to the lattice, then it is possible to add and

multiply ciphertexts for a while before the error grows beyond the decryption radius of the secret key.

The Smart-Vercauteren Variant. Smart and Vercauteren [13] work over the ring $R = \mathbb{Z}[x]/f_n(x)$, where $f_n(x) = x^n + 1$ and n is a power of two. The ideal J is set as a principal ideal by choosing a vector $\boldsymbol{v}$ at random from some n-dimensional cube, subject to the condition that the determinant of $(\boldsymbol{v})$ is prime, and then setting $J = (\boldsymbol{v})$. It is known that such ideals can be implicitly represented by only two integers, namely the determinant $d = \det(J)$ and a root r of $f_n(x)$ modulo d. (An easy proof of this fact "from first principles" can be derived from our Lemma 1 below.) Specifically, the Hermite normal form of this ideal lattice is

$$\mathrm{HNF}(J) = \begin{bmatrix} d & 0 & 0 & 0 & 0 \\ -r & 1 & 0 & 0 & 0 \\ -[r^2]_d & 0 & 1 & 0 & 0 \\ -[r^3]_d & 0 & 0 & 1 & 0 \\ & & & & \ddots \\ -[r^{n-1}]_d & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

It is easy to see that reducing a vector $\boldsymbol{a}$ modulo $\mathrm{HNF}(J)$ consists of evaluating the associated polynomial $a(x)$ at the point r modulo d, then outputting the vector $\langle [a(r)]_d, 0, 0, \ldots, 0\rangle$ (see Section 5). Hence encryption of a vector $\langle m, 0, 0, \ldots, 0\rangle$ with $m \in \{0, 1\}$ can be done by choosing a random small polynomial $u(x)$ and evaluating it at r, then outputting the integer $c \leftarrow [2u(r) + m]_d$.

Smart and Vercauteren also describe a decryption procedure that uses a single integer w as the secret key, setting $m \leftarrow (c - \lceil cw/d \rfloor) \bmod 2$. Jumping ahead, we note that our decryption procedure from Section 6 is very similar, except that for convenience we replace the rational division cw/d by modular multiplication $[cw]_d$.

3 Key Generation

We adopt the Smart-Vercauteren approach [13], in that we also use principal-ideal lattices in the ring of polynomials modulo $f_n(x) \stackrel{\text{def}}{=} x^n + 1$ with n a power of two. We do not require that these principal-ideal lattices have prime determinant, instead we only need the Hermite normal form to have the same form as in Equation (1). During key-generation we choose $\boldsymbol{v}$ at random in some cube, verify that the HNF has the right form, and work with the principal ideal $(\boldsymbol{v})$. We have two parameters: the dimension n, which must be a power of two, and the bit-size t of coefficients in the generating polynomial. Key-generation consists of the following steps:

1. Choose a random n-dimensional integer lattice $\boldsymbol{v}$, where each entry v_i is chosen at random as a t-bit (signed) integer. With this vector $\boldsymbol{v}$ we associate the formal polynomial $v(x) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} v_i x^i$, as well as the rotation basis:

$$V = \begin{bmatrix} v_0 & v_1 & v_2 & v_{n-1} \\ -v_{n-1} & v_0 & v_1 & v_{n-2} \\ -v_{n-2} & -v_{n-1} & v_0 & v_{n-3} \\ & & & \ddots \\ -v_1 & -v_2 & -v_3 & v_0 \end{bmatrix} \tag{2}$$

The i'th row is a cyclic shift of $\boldsymbol{v}$ by i positions to the right, with the "overflow entries" negated. Note that the i'th row corresponds to the coefficients of the polynomial $v_i(x) = v(x) \times x^i \pmod{f_n(x)}$. Note that just like V itself, the entire lattice $\mathcal{L}(V)$ is also closed under "rotation": Namely, for any vector $\langle u_0, u_1, \ldots, u_{n-1}\rangle \in \mathcal{L}(V)$, also the vector $\langle -u_{n-1}, u_0, \ldots, u_{n-2}\rangle$ is in $\mathcal{L}(V)$.

2. Next we compute the scaled inverse of $v(x)$ modulo $f_n(x)$, namely an integer polynomial $w(x)$ of degree at most $n-1$, such that $w(x) \times v(x) = \mathsf{constant} \pmod{f_n(x)}$. Specifically, this constant is the determinant of the lattice $\mathcal{L}(V)$, which must be equal to the resultant of the polynomials $v(x)$ and $f_n(x)$ (since f_n is monic). Below we denote the resultant by d, and denote the coefficient-vector of $w(x)$ by $\boldsymbol{w} = \langle w_0, w_1, \ldots, w_{n-1}\rangle$. It is easy to check that the matrix

$$W = \begin{bmatrix} w_0 & w_1 & w_2 & w_{n-1} \\ -w_{n-1} & w_0 & w_1 & w_{n-2} \\ -w_{n-2} & -w_{n-1} & w_0 & w_{n-3} \\ & & & \ddots \\ -w_1 & -w_2 & -w_3 & w_0 \end{bmatrix} \tag{3}$$

is the scaled inverse of V, namely $W \times V = V \times W = d \cdot I$. One way to compute the polynomial $w(x)$ is by applying the extended Euclidean-GCD algorithm (for polynomials) to $v(x)$ and $f_n(x)$. See Section 4 for a more efficient method of computing $w(x)$.

3. We also check that this is a good generating polynomial. We consider $\boldsymbol{v}$ to be good if the Hermite-Normal-form of V has the same form as in Equation (1), namely all except the leftmost column equal to the identity matrix. See below for a simple check that the $\boldsymbol{v}$ is good, in our implementation we test this condition while computing the inverse.

   It was observed by Nigel Smart that the HNF has the correct form whenever the determinant is odd and square-free. Indeed, in our tests this condition was met with probability roughly 0.5, irrespective of the dimension and bit length, with the failure cases usually due to the determinant of V being even.

Checking the HNF. In Lemma 1 below we prove that the HNF of the lattice $\mathcal{L}(V)$ has the right form if and only if the lattice contains a vector of the form $\langle -r, 1, 0, \ldots, 0\rangle$. Namely, if and only if there exists an integer vector $\boldsymbol{y}$ and another integer r such that

$$\boldsymbol{y} \times V = \langle -r, 1, 0, \ldots, 0\rangle$$

Multiplying the last equation on the right by W, we get the equivalent condition

$$\begin{aligned} & \boldsymbol{y} \times V \times W = \langle -r, 1, 0 \ldots, 0 \rangle \times W \qquad (4) \\ \Leftrightarrow \quad & \boldsymbol{y} \times (dI) = d \cdot \boldsymbol{y} = -r \cdot \langle w_0, w_1, w_2, \ldots, w_{n-1} \rangle + \langle -w_{n-1}, w_0, w_1, \ldots, w_{n-2} \rangle \end{aligned}$$

In other words, there must exists an integer r such that the second row of W minus r times the first row yields a vector of integers that are all divisible by d:

$$\begin{aligned} & -r \cdot \langle w_0, w_1, w_2, \ldots, w_{n-1} \rangle + \langle -w_{n-1}, w_0, w_1, \ldots, w_{n-2} \rangle \;=\; 0 \pmod d \\ \Leftrightarrow\; & -r \cdot \langle w_0, w_1, w_2, \ldots, w_{n-1} \rangle \;=\; \langle w_{n-1}, -w_0, -w_1, \ldots, -w_{n-2} \rangle \pmod d \end{aligned}$$

The last condition can be checked easily: We compute $r := w_0/w_1 \bmod d$ (assuming that w_1 has an inverse modulo d), then check that $r \cdot w_{i+1} = w_i \pmod d$ holds for all $i = 1, \ldots, n-2$ and also $-r \cdot w_0 = w_{n-1} \pmod d$. Note that this means in particular that $r^n = -1 \pmod d$. (In our implementation we actually test only that last condition, instead of testing all the equalities $r \cdot w_{i+1} = w_i \pmod d$.)

Lemma 1. *The Hermite normal form of the matrix V from Equation (2) is equal to the identity matrix in all but the leftmost column, if and only if the lattice spanned by the rows of V contains a vector of the form $\boldsymbol{r} = \langle -r, 1, 0 \ldots, 0 \rangle$.*

Proof. Let B be the Hermite normal form of V. Namely, B is lower triangular matrix with non-negative diagonal entries, where the rows of B span the same lattice as the rows of V, and the absolute value of every entry under the diagonal in B is no more than half the diagonal entry above it. This matrix B can be obtained from V by a sequence of elementary row operations, and it is unique. It is easy to see that the existence of a vector $\boldsymbol{r}$ of this form is necessary: indeed the second row of B must be of this form (since B is equal the identity in all except the leftmost column). We now prove that this condition is also sufficient.

It is clear that the vector $d \cdot \boldsymbol{e_1} = \langle d, 0, \ldots, 0 \rangle$ belongs to $\mathcal{L}(V)$: in particular we know that $\langle w_0, w_1, \ldots, w_{n-1} \rangle \times V = \langle d, 0, \ldots, 0 \rangle$. Also, by assumption we have $\boldsymbol{r} = -r \cdot \boldsymbol{e_1} + \boldsymbol{e_2} \in \mathcal{L}(V)$, for some integer r. Note that we can assume without loss of generality that $-d/2 \le r < d/2$, since otherwise we could subtract from $\boldsymbol{r}$ multiples of the vector $d \cdot \boldsymbol{e_1}$ until this condition is satisfied:

$$\begin{aligned} & \langle -r \quad 1\ 0\ \ldots\ 0 \rangle \\ -\kappa \cdot & \langle \quad d \quad 0\ 0\ \ldots\ 0 \rangle \\ = \; & \overline{\langle [-r]_d \ 1\ 0\ \ldots\ 0 \rangle} \end{aligned}$$

For $i = 1, 2, \ldots, n-1$, denote $r_i \stackrel{\text{def}}{=} [r^i]_d$. Below we will prove by induction that for all $i = 1, 2, \ldots, n-1$, the lattice $\mathcal{L}(V)$ contains the vector:

$$\boldsymbol{r_i} \stackrel{\text{def}}{=} -r_i \cdot \boldsymbol{e_1} + \boldsymbol{e}_{i+1} \;=\; \underbrace{\langle -r_i, 0 \ldots 0, 1, 0 \ldots 0 \rangle}_{1 \text{ in the } i+1\text{'st position}}.$$

Placing all these vectors $\boldsymbol{r_i}$ at the rows of a matrix, we got exactly the matrix B that we need:

$$B = \begin{bmatrix} d & 0 & 0 & \cdots & 0 \\ -r_1 & 1 & 0 & \cdots & 0 \\ -r_2 & 0 & 1 & \cdots & 0 \\ & & & \ddots & \\ -r_{n-1} & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{5}$$

B is equal to the identity except in the leftmost column, its rows are all vectors in $\mathcal{L}(V)$ (so they span a sub-lattice), and since B has the same determinant as V then it cannot span a proper sub-lattice, it must therefore span $\mathcal{L}(V)$ itself.

It is left to prove the inductive claim. For $i = 1$ we set $\boldsymbol{r_1} \stackrel{\text{def}}{=} \boldsymbol{r}$ and the claim follow from our assumption that $\boldsymbol{r} \in \mathcal{L}(V)$. Assume now that it holds for some $i \in [1, n-2]$ and we prove for $i+1$. Recall that the lattice $\mathcal{L}(V)$ is closed under rotation, and since $\boldsymbol{r_i} = -r_i\boldsymbol{e_1} + \boldsymbol{e}_{i+1} \in \mathcal{L}(V)$ then the right-shifted vector $\boldsymbol{s}_{i+1} \stackrel{\text{def}}{=} -r_i\boldsymbol{e_2} + \boldsymbol{e}_{i+2}$ is also in $\mathcal{L}(V)$[3]. Hence $\mathcal{L}(V)$ contains also the vector

$$\boldsymbol{s}_{i+1} + r_i \cdot \boldsymbol{r} \;=\; (-r_i\boldsymbol{e_2} + \boldsymbol{e_{i+2}}) + r_i(-r\boldsymbol{e_1} + \boldsymbol{e_2}) = \;= -r_i r \cdot \boldsymbol{e_1} + \boldsymbol{e}_{i+2}$$

We can now reduce the first entry in this vector modulo d, by adding/subtracting the appropriate multiple of $d \cdot \boldsymbol{e_1}$ (while still keeping it in the lattice), thus getting the lattice vector

$$[-r \cdot r_i]_d \cdot \boldsymbol{e_1} + \boldsymbol{e_{i+2}} \;=\; -[r^{i+1}]_d \cdot \boldsymbol{e_1} + \boldsymbol{e_{i+2}} \;=\; \boldsymbol{r}_{i+1} \;\in\; \mathcal{L}(V)$$

This concludes the proof.

Remark 1. Note that the proof of Lemma 1 shows in particular that if the Hermite normal form of V is equal to the identity matrix in all but the leftmost column, then it must be of the form specified in Equation (5). Namely, the first column is $\langle d, -r_1, -r_2, \ldots, -r_{n-1}\rangle^t$, with $r_i = [r^i]_d$ for all i. Hence this matrix can be represented implicitly by the two integers d and r.

3.1 The Public and Secret Keys

In principle the public key is the Hermite normal form of V, but as we explain in Remark 1 and Section 5 it is enough to store for the public key only the two integers d, r. Similarly, in principle the secret key is the pair $(\boldsymbol{v}, \boldsymbol{w})$, but as we explain in Section 6.1 it is sufficient to store only a single (odd) coefficient of $\boldsymbol{w}$ and discard $\boldsymbol{v}$ altogether.

4 Inverting the Polynomial $v(x)$

The fastest known methods for inverting the polynomial $v(x)$ modulo $f_n(x) = x^n + 1$ are based on FFT: We can evaluate $v(x)$ at all the roots of $f_n(x)$ (either

[3] This is a circular shift, since $i \leq n-2$ and hence the rightmost entry in $\boldsymbol{r_i}$ is zero.

over the complex field or over some finite field), then compute $w^*(\rho) = 1/v(\rho)$ (where inversion is done over the corresponding field), and then interpolate $w^* = v^{-1}$ from all these values. If the resultant of v and f_n has N bits, then this procedure will take $O(n \log n)$ operations over $O(N)$-bit numbers, for a total running time of $\tilde{O}(nN)$. This is close to optimal in general, since just writing out the coefficients of the polynomial w^* takes time $O(nN)$. However, in Section 6.1 we show that it is enough to use for the secret key only one of the coefficients of $w = d \cdot w^*$ (where $d = \mathsf{resultant}(v, f_n)$). This raises the possibility that we can compute this one coefficient in time quasi-linear in N rather than quasi-linear in nN. Although polynomial inversion is very well researched, as far as we know this question of computing just one coefficient of the inverse was not tackled before. Below we describe an algorithm for doing just that.

The approach for the procedure below is to begin with the polynomial v that has n small coefficients, and proceed in steps where in each step we halve the number of coefficients to offset the fact that the bit-length of the coefficients approximately doubles. Our method relies heavily on the special form of $f_n(x) = x^n + 1$, with n a power of two. Let $\rho_0, \rho_1, \ldots, \rho_{n-1}$ be roots of $f_n(x)$ over the complex field: That is, if ρ is some primitive $2n$'th root of unity then $\rho_i = \rho^{2i+1}$. Note that the roots r_i satisfy that $\rho_{i+\frac{n}{2}} = -\rho_i$ for all i, and more generally for every index i (with index arithmetic modulo n) and every $j = 0, 1, \ldots, \log n$, if we denote $n_j \stackrel{\text{def}}{=} n/2^j$ then it holds that

$$\left(\rho_{i+\,n_j/2}\right)^{2^j} \;=\; \left(\rho^{2i+n_j+1}\right)^{2^j} \;=\; \left(\rho^{2i+1}\right)^{2^j} \cdot \rho^n \;=\; -\left(\rho_i^{2^j}\right) \tag{6}$$

The method below takes advantage of Equation (6), as well as a connection between the coefficients of the scaled inverse w and those of the formal polynomial

$$g(z) \stackrel{\text{def}}{=} \prod_{i=0}^{n-1} \big(v(\rho_i) - z\big).$$

We invert $v(x) \bmod f_n(x)$ by computing the lower two coefficients of $g(z)$, then using them to recover both the resultant and (one coefficient of) the polynomial $w(x)$, as described next.

Step one: the polynomial $g(z)$. Note that although the polynomial $g(z)$ is defined via the complex numbers ρ_i, the coefficients of $g(z)$ are all integers. We begin by showing how to compute the lower two coefficients of $g(z)$, namely the polynomial $g(z) \bmod z^2$. We observe that since $\rho_{i+\frac{n}{2}} = -\rho_i$ then we can write $g(z)$ as

$$\begin{aligned} g(z) &= \prod_{i=0}^{\frac{n}{2}-1} (v(\rho_i) - z)(v(-\rho_i) - z) \\ &= \prod_{i=0}^{\frac{n}{2}-1} \Big(\underbrace{v(\rho_i)v(-\rho_i)}_{a(\rho_i)} - z\underbrace{(v(\rho_i) + v(-\rho_i))}_{b(\rho_i)} + z^2 \Big) = \prod_{i=0}^{\frac{n}{2}-1} \Big(a(\rho_i) - z b(\rho_i) \Big) \pmod{z^2} \end{aligned}$$

We observe further that for both the polynomials $a(x) = v(x)v(-x)$ and $b(x) = v(x) + v(-x)$, all the odd powers of x have zero coefficients. Moreover, the same equalities as above hold if we use $A(x) = a(x) \bmod f_n(x)$ and $B(x) = b(x) \bmod f_n(x)$ instead of $a(x)$ and $b(x)$ themselves (since we only evaluate these polynomials in roots of f_n), and also for A, B all the odd powers of x have zero coefficients (since we reduce modulo $f_n(x) = x^n + 1$ with n even).

Thus we can consider the polynomials $\hat{v}, \tilde{v}$ that have half the degree and only use the nonzero coefficients of A, B, respectively. Namely they are defined via $\hat{v}(x^2) = A(x)$ and $\tilde{v}(x^2) = B(x)$. Thus we have reduced the task of computing the n-product involving the degree-n polynomial $v(x)$ to computing a product of only $n/2$ terms involving the degree-$n/2$ polynomials $\hat{v}(x), \tilde{v}(x)$. Repeating this process recursively, we obtain the polynomial $g(z) \bmod z^2$. The details of this process are described in Section 4.1 below.

Step two: recovering d and w_0. Recall that if $v(x)$ is square free then $d = \mathsf{resultant}(v, f_n) = \prod_{i=0}^{n-1} v(\rho_i)$, which is exactly the free term of $g(z)$, $g_0 = \prod_{i=0}^{n-1} v(\rho_i)$.

Recall also that the linear term in $g(z)$ has coefficient $g_1 = \sum_{i=0}^{n-1} \prod_{j \neq i} v(\rho_i)$. We next show that the free term of $w(x)$ is $w_0 = g_1/n$. First, we observe that g_1 equals the sum of w evaluated in all the roots of f_n, namely

$$g_1 = \sum_{i=0}^{n-1} \prod_{j \neq i} v(\rho_j) = \sum_{i=0}^{n-1} \frac{\prod_{j=0}^{n-1} v(\rho_j)}{v(\rho_i)} \overset{\text{(a)}}{=} \sum_{i=0}^{n-1} \frac{d}{v(\rho_i)} \overset{\text{(b)}}{=} \sum_{i=0}^{n-1} w(\rho_i)$$

where Equality (a) follows since $v(x)$ is square free and $d = \mathsf{resultant}(v, f_n)$, and Equality (b) follows since $v(\rho_i) = d/w(\rho_i)$ holds in all the roots of f_n. It is left to show that the constant term of $w(x)$ is $w_0 = n \sum_{i=0}^{n-1} w(\rho_i)$. To show this, we write

$$\sum_{i=0}^{n-1} w(\rho_i) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} w_j \rho_i^j = \sum_{j=0}^{n-1} w_j \sum_{i=0}^{n-1} \rho_i^j \overset{(\star)}{=} \sum_{j=0}^{n-1} w_j \sum_{i=0}^{n-1} (\rho^j)^{2i+1} \tag{7}$$

where the Equality $(\star)$ holds since the i'th root of f_n is $\rho_i = \rho^{2i+1}$ where ρ is a $2n$-th root of unity. Clearly, the term corresponding to $j = 0$ in Equation (7) is $w_0 \cdot n$, it is left to show that all the other terms are zero. This follows since ρ^j is a $2n$-th root of unity different from ± 1 for all $j = 1, 2, \ldots, n-1$, and summing over all odd powers of such root of unity yields zero.

Step three: recovering the rest of w. We can now use the same technique to recover all the other coefficients of w: Note that since we work modulo $f_n(x) = x^n + 1$, then the coefficient w_i is the free term of the scaled inverse of $x^i \times v \pmod{f_n}$.

In our case we only need to recover the first two coefficients, however, since we are only interested in the case where $w_1/w_0 = w_2/w_1 = \cdots = w_{n-1}/w_{n-2} = -w_0/w_{n-1} \pmod{d}$, where $d = \mathsf{resultant}(v, f_n)$. After recovering w_0, w_1 and

$d = \mathsf{resultant}(v, f_n)$, we therefore compute the ratio $r = w_1/w_0 \bmod d$ and verify that $r^n = -1 \pmod d$. Then we recover as many coefficients of w as we need (via $w_{i+1} = [w_i \cdot r]_d$), until we find one coefficient which is an odd integer, and that coefficient is the secret key.

4.1 The Gory Details of Step One

We denote $U_0(x) \equiv 1$ and $V_0(x) = v(x)$, and for $j = 0, 1, \ldots, \log n$ we denote $n_j = n/2^j$. We proceed in $m = \log n$ steps to compute the polynomials $U_j(x), V_j(x)$ $(j = 1, 2, \ldots, m)$, such that the degrees of U_j, V_j are at most $n_j - 1$, and moreover the polynomial $g_j(z) = \prod_{i=0}^{n_j-1}(V_j(\rho_i^{2^j}) - zU_j(\rho_i^{2^j}))$ has the same first two coefficients as $g(z)$. Namely,

$$g_j(z) \stackrel{\text{def}}{=} \prod_{i=0}^{n_j-1}\left(V_j(\rho_i^{2^j}) - zU_j(\rho_i^{2^j})\right) = g(z) \pmod{z^2}. \tag{8}$$

Equation (8) holds for $j = 0$ by definition. Assume that we computed U_j, V_j for some $j < m$ such that Equation (8) holds, and we show how to compute U_{j+1} and V_{j+1}. From Equation (6) we know that $\left(\rho_{i+n_j/2}\right)^{2^j} = -\rho_i^{2^j}$, so we can express g_j as

$$\begin{aligned} g_j(z) &= \prod_{i=0}^{n_j/2-1}\left(V_j(\rho_i^{2^j}) - zU_j(\rho_i^{2^j})\right)\left(V_j(-\rho_i^{2^j}) - zU_j(-\rho_i^{2^j})\right) \\ &= \prod_{i=0}^{n_j/2-1}\left(\underbrace{V_j(\rho_i^{2^j})V_j(-\rho_i^{2^j})}_{=A_j(\rho_i^{2^j})} - z\underbrace{\left(U_j(\rho_i^{2^j})V_j(-\rho_i^{2^j}) + U_j(-\rho_i^{2^j})V_j(\rho_i^{2^j})\right)}_{=B_j(\rho_i^{2^j})}\right) \pmod{z^2} \end{aligned}$$

Denoting $f_{n_j}(x) \stackrel{\text{def}}{=} x^{n_j} + 1$ and observing that ${\rho_i}^{2^j}$ is a root of f_{n_j} for all i, we next consider the polynomials:

$$A_j(x) \stackrel{\text{def}}{=} V_j(x)V_j(-x) \bmod f_{n_j}(x) \quad \text{(with coefficients } a_0, \ldots, a_{n_j-1})$$

$$B_j(x) \stackrel{\text{def}}{=} U_j(x)V_j(-x) + U_j(-x)V_j(x) \bmod f_{n_j}(x) \quad \text{(with coefficients } b_0, \ldots, b_{n_j-1})$$

and observe the following:

- Since ${\rho_i}^{2^j}$ is a root of f_{n_j}, then the reduction modulo f_{n_j} makes no difference when evaluating A_j, B_j on ${\rho_i}^{2^j}$. Namely we have $A_j(\rho_i^{2^j}) = V_j(\rho_i^{2^j})V_j(-\rho_i^{2^j})$ and similarly $B_j(\rho_i^{2^j}) = U_j(\rho_i^{2^j})V_j(-\rho_i^{2^j}) + U_j(-\rho_i^{2^j})V_j(\rho_i^{2^j})$ (for all i).
- The odd coefficients of A_j, B_j are all zero. For A_j this is because it is obtained as $V_j(x)V_j(-x)$ and for B_j this is because it is obtained as $R_j(x) + R_j(-x)$ (with $R_j(x) = U_j(x)V_j(-x)$). The reduction modulo $f_{n_j}(x) = x^{n_j} + 1$ keeps the odd coefficients all zero, because n_j is even.

We therefore set

$$U_{j+1}(x) \stackrel{\text{def}}{=} \sum_{t=0}^{n_j/2-1} b_{2t} \cdot x^t, \quad \text{and} \quad V_{j+1}(x) \stackrel{\text{def}}{=} \sum_{t=0}^{n_j/2-1} a_{2t} \cdot x^t,$$

so the second bullet above implies that $U_{j+1}(x^2) = B_j(x)$ and $V_{j+1}(x^2) = A_j(x)$ for all x. Combined with the first bullet, we have that

$$\begin{aligned} g_{j+1}(z) &\stackrel{\text{def}}{=} \prod_{i=0}^{n_j/2-1} \left(V_{j+1}(\rho_i^{2^{j+1}}) - z \cdot U_{j+1}(\rho_i^{2^{j+1}})\right) \\ &= \prod_{i=0}^{n_j/2-1} \left(A_j(\rho_i^{2^j}) - z \cdot B_j(\rho_i^{2^j})\right) \;=\; g_j(z) \pmod{z^2}. \end{aligned}$$

By the induction hypothesis we also have $g_j(z) = g(z) \pmod{z^2}$, so we get $g_{j+1}(z) = g(z) \pmod{z^2}$, as needed.

5 Encryption

To encrypt a bit $b \in \{0,1\}$ with the public key B (which is implicitly represented by the two integers d, r), we first choose a random $0, \pm 1$ "noise vector" $\boldsymbol{u} \stackrel{\text{def}}{=} \langle u_0, u_1, \ldots, u_{n-1}\rangle$, with each entry chosen as 0 with some probability q and as ± 1 with probability $(1-q)/2$ each. We then set $\boldsymbol{a} \stackrel{\text{def}}{=} 2\boldsymbol{u} + b \cdot \boldsymbol{e}_1 = \langle 2u_0 + b, 2u_1, \ldots, 2u_{n-1}\rangle$, and the ciphertext is the vector

$$\boldsymbol{c} \;=\; \boldsymbol{a} \bmod B \;=\; \boldsymbol{a} - \left(\lfloor \boldsymbol{a} \times B^{-1} \rceil \times B\right) \;=\; \underbrace{\left[\boldsymbol{a} \times B^{-1}\right]}_{[\cdot] \text{ is fractional part}} \times B$$

We now show that also $\boldsymbol{c}$ can be represented implicitly by just one integer. Recall that B (and therefore also B^{-1}) are of a special form

$$B = \begin{bmatrix} d & 0\,0\,0 & 0 \\ -r & 1\,0\,0 & 0 \\ -[r^2]_d & 0\,1\,0 & 0 \\ -[r^3]_d & 0\,0\,1 & 0 \\ & & \ddots \\ -[r^{n-1}]_d & 0\,0\,0 & 1 \end{bmatrix}, \text{ and } B^{-1} = \frac{1}{d} \cdot \begin{bmatrix} 1 & 0\,0\,0 & 0 \\ r & d\,0\,0 & 0 \\ [r^2]_d & 0\,d\,0 & 0 \\ [r^3]_d & 0\,0\,d & 0 \\ & & \ddots \\ [r^{n-1}]_d & 0\,0\,0 & d \end{bmatrix}.$$

Denote $\boldsymbol{a} = \langle a_0, a_1, \ldots, a_{n-1}\rangle$, and also denote by $a(\cdot)$ the integer polynomial $a(x) \stackrel{\text{def}}{=} \sum_{i=0}^{n-1} a_i x^i$. Then we have $\boldsymbol{a} \times B^{-1} = \left\langle \frac{s}{d}, a_1, \ldots, a_{n-1}\right\rangle$ for some integer s that satisfies $s = a(r) \pmod{d}$. Hence the fractional part of $\boldsymbol{a} \times B^{-1}$ is $\left[\boldsymbol{a} \times B^{-1}\right] = \left\langle \frac{[a(r)]_d}{d}, 0, \ldots, 0\right\rangle$, and the ciphertext vector is $\boldsymbol{c} = \left\langle \frac{[a(r)]_d}{d}, 0, \ldots, 0\right\rangle \times B = \langle [a(r)]_d, 0, \ldots, 0\rangle$. Clearly, this vector can

be represented implicitly by the integer $c \stackrel{\text{def}}{=} \left[a(r)\right]_d = \left[b + 2\sum_{i=1}^{n-1} u_i r^i\right]_d$. Hence, to encrypt the bit b, we only need to evaluate the noise-polynomial $u(\cdot)$ at the point r, then multiply by two and add the bit b (everything modulo d). We now describe an efficient procedure for doing that.

5.1 An Efficient Encryption Procedure

The most expensive operation during encryption is evaluating the degree-$(n-1)$ polynomial u at the point r. Polynomial evaluation using Horner's rule takes $n-1$ multiplications, but it is known that for small coefficients we can reduce the number of multiplications to only $O(\sqrt{n})$, see [1,10]. Moreover, we observe that it is possible to batch this fast evaluation algorithm, and evaluate k such polynomials in time $O(\sqrt{kn})$.

We begin by noting that evaluating many $0, \pm 1$ polynomials at the same point x can be done about as fast as a naive evaluation of a single polynomial. Indeed, once we compute all the powers $(1, x, x^2, \ldots, x^{n-1})$ then we can evaluate each polynomial just by taking a subset-sum of these powers. As addition is much faster than multiplication, the dominant term in the running time will be the computation of the powers of x, which we only need to do once for all the polynomials.

Next, we observe that evaluating a single degree-$(n-1)$ polynomial at a point x can be done quickly given a subroutine that evaluates two degree-$(n/2-1)$ polynomials at the same point x. Namely, given $u(x) = \sum_{i=0}^{n-1} u_i x^i$, we split it into a "bottom half" $u^{\mathsf{bot}}(x) = \sum_{i=0}^{n/2-1} u_i x^i$ and a "top half" $u^{\mathsf{top}}(x) = \sum_{i=0}^{n/2-1} u_{i+n/2} x^i$. Evaluating these two smaller polynomials we get $y^{\mathsf{bot}} = u^{\mathsf{bot}}(x)$ and $y^{\mathsf{top}} = u^{\mathsf{top}}(x)$, and then we can compute $y = u(x)$ by setting $y = x^{n/2} y^{\mathsf{top}} + y^{\mathsf{bot}}$. If the subroutine for evaluating the two smaller polynomials also returns the value of $x^{n/2}$, then we need just one more multiplication to get the value of $y = u(x)$.

These two observations suggest a recursive approach to evaluating the $0, \pm 1$ polynomial u of degree $n-1$. Namely, we repeatedly cut the degree in half at the price of doubling the number of polynomials, and once the degree is small enough we use the "trivial implementation" of just computing all the powers of x. Analyzing this approach, let us denote by $M(k, n)$ the number of multiplications that it takes to evaluate k polynomials of degree $(n-1)$. Then we have $M(k, n) \leq \min(n-1,\ M(2k, n/2) + k + 1)$. To see the bound $M(k, n) \leq M(2k, n/2) + k + 1$, note that once we evaluated the top- and bottom-halves of all the k polynomials, we need one multiplication per polynomial to put the two halves together, and one last multiplication to compute x^n (which is needed in the next level of the recursion) from $x^{n/2}$ (which was computed in the previous level). Obviously, making the recursive call takes less multiplications than the "trivial implementation" whenever $n - 1 > (n/2 - 1) + k + 1$. Also, an easy inductive argument shows that the "trivial implementation" is better when $n - 1 < (n/2 - 1) + k + 1$. We thus get the recursive formula

$$M(k,n) \;=\; \begin{cases} M(2k,n/2)+k+1 & \text{when } n/2 > k+1 \\ n-1 & \text{otherwise.} \end{cases}$$

Solving this formula we get $M(k,n) \le \min(n-1, \sqrt{2kn})$. In particular, the number of multiplications needed for evaluating a single degree-$(n-1)$ polynomial is $M(1,n) \le \sqrt{2n}$.

We comment that this "more efficient" batch procedure relies on the assumption that we have enough memory to keep all these partially evaluated polynomials at the same time. In our experiments we were only able to use it in dimensions up to $n = 2^{15}$, trying to use it in higher dimension resulted in the process being killed after it ran out of memory. A more sophisticated implementation could take the available amount of memory into account, and stop the recursion earlier to preserve space at the expense of more running time. An alternative approach, of course, is to store partial results to disk. More experiments are needed to determine what approach yields better performance for which parameters.

5.2 The Euclidean Norm of Fresh Ciphertexts

When choosing the noise vector for a new ciphertext, we want to make it as sparse as possible, i.e., increase as much as possible the probability q of choosing each entry as zero. The only limitation is that we need q to be bounded sufficiently below 1 to make it hard to recover the original noise vector from c. There are two types of attacks that we need to consider: lattice-reduction attacks that try to find the closest lattice point to c, and exhaustive-search/birthday attacks that try to guess the coefficients of the original noise vector (and a combination thereof). Pure lattice-reduction attacks should be thwarted by working with lattices with high-enough dimension, so we concentrate here on exhaustive-search attacks.

Roughly, if the noise vector has ℓ bits of entropy, then we expect birthday-type attacks to be able to recover it in $2^{\ell/2}$ time, so we need to ensure that the noise has at least 2λ bits of entropy for security parameter λ. Namely, for dimension n we need to choose q sufficiently smaller than one so that $2^{(1-q)n} \cdot \binom{n}{qn} > 2^{2\lambda}$.

Another "hybrid" attack is to choose a small random subset of the powers of r (e.g., only 200 of them) and "hope" that they include all the noise coefficients. If this holds then we can now search for a small vector in this low-dimension lattice (e.g., dimension 200). For example, if we work in dimension $n = 2048$ and use only 16 nonzero entries for noise, then choosing 200 of the 2048 entries, we have probability of about $(200/2048)^{16} \approx 2^{54}$ of including all of them (hence we can recover the original noise by solving 2^{54} instances of SVP in dimension 200). The same attack will have success probability only $\approx 2^{-80}$ if we use 24 nonzero entries.

For our public challenges we chose a (somewhat aggressive) setting where the number of nonzero entries in the noise vector is between 15 and 20. We note that increasing the noise will have only moderate effect on the performance numbers of our fully-homomorphic scheme, for example using 30 nonzero entries is likely to increase the size of the key (and the running time) by only about 5-10%.

6 Decryption

The decryption procedure takes the ciphertext c (which implicitly represents the vector $\boldsymbol{c} = \langle c, 0, \ldots, 0\rangle$) and in principle it also has the two matrices V, W. The vector $\boldsymbol{a} = 2\boldsymbol{u} + b \cdot \boldsymbol{e}_1$ that was used during encryption is recovered as $\boldsymbol{a} \leftarrow \boldsymbol{c} \bmod V = \lfloor \boldsymbol{c} \times W/d \rceil$, and then outputs the least significant bit of the first entry of $\boldsymbol{a}$, namely $b := a_0 \bmod 2$.

The reason that this decryption procedure works is that the rows of V (and therefore also of W) are close to being orthogonal to each other, and hence the operator l_∞-norm of W is small. Namely, for any vector $\boldsymbol{x}$, the largest entry in $\boldsymbol{x} \times W$ (in absolute value) is not much larger than the largest entry in $\boldsymbol{x}$ itself. Specifically, the procedure from above succeeds when all the entries of $\boldsymbol{a} \times W$ are smaller than $d/2$ in absolute value. To see that, note that $\boldsymbol{a}$ is the distance between $\boldsymbol{c}$ and some point in the lattice $\mathcal{L}(V)$, namely we can express $\boldsymbol{c}$ as $\boldsymbol{c} = \boldsymbol{y} \times V + \boldsymbol{a}$ for some integer vector $\boldsymbol{y}$. Hence we have

$$\lfloor \boldsymbol{c} \times W/d \rceil \times V = \lfloor \boldsymbol{y} \times V \times W/d + \boldsymbol{a} \times W/d \rceil \overset{(\star)}{=} \lfloor \boldsymbol{a} \times W/d \rceil \times V$$

where the equality $(\star)$ follows since $\boldsymbol{y} \times V \times W/d$ is an integer vector. The vector $\lfloor \boldsymbol{a} \times W/d \rceil \times V$ is supposed to be $\boldsymbol{a}$ itself, namely we need $\lfloor \boldsymbol{a} \times W/d \rceil \times V = \boldsymbol{a} = (\boldsymbol{a} \times W/d) \times V$. But this last condition holds if and only if $\lfloor \boldsymbol{a} \times W/d \rceil = (\boldsymbol{a} \times W/d)$, i.e., $\boldsymbol{a} \times W/d$ is equal to its fractional part, which means that every entry in $\boldsymbol{a} \times W/d$ must be less than $1/2$ in absolute value.

6.1 An Optimized Decryption Procedure

We next show that the encrypted bit b can be recovered by a significantly cheaper procedure: Recall that the (implicitly represented) ciphertext vector $\boldsymbol{c}$ is decrypted to the bit b when the distance from $\boldsymbol{c}$ to the nearest vector in the lattice $\mathcal{L}(V)$ is of the form $\boldsymbol{a} = 2\boldsymbol{u} + b\boldsymbol{e_1}$, and moreover all the entries in $\boldsymbol{a} \times W$ are less than $d/2$ in absolute value. As we said above, in this case we have $[\boldsymbol{c} \times W/d] = [\boldsymbol{a} \times W/d] = \boldsymbol{a} \times W/d$, which is equivalent to the condition $[\boldsymbol{c} \times W]_d = [\boldsymbol{a} \times W]_d = \boldsymbol{a} \times W$. Recall now that $\boldsymbol{c} = \langle c, 0, \ldots, 0\rangle$, hence

$$[\boldsymbol{c} \times W]_d = [c \cdot \langle w_0, w_1, \ldots, w_{n-1}\rangle]_d = \langle [cw_0]_d, [cw_1]_d, \ldots, [cw_{n-1}]_d \rangle .$$

On the other hand, we have

$$[\boldsymbol{c} \times W]_d = \boldsymbol{a} \times W = 2\boldsymbol{u} \times W + b\boldsymbol{e_1} \times W = 2\boldsymbol{u} \times W + b \cdot \langle w_0, w_1, \ldots, w_{n-1}\rangle .$$

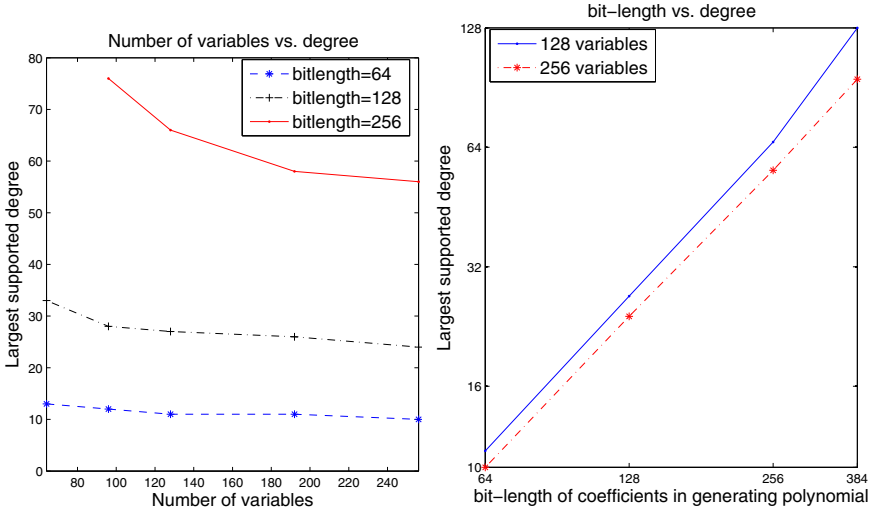
Putting these two equations together, we get that any decryptable ciphertext c must satisfy the relation

$$\langle [cw_0]_d, [cw_1]_d, \ldots, [cw_{n-1}]_d \rangle = b \cdot \langle w_0, w_1, \ldots, w_{n-1}\rangle \pmod 2$$

In other words, for every i we have $[c \cdot w_i]_d = b \cdot w_i \pmod 2$. It is therefore sufficient to keep only one of the w_i's (which must be odd), and then recover the bit b as $b := [c \cdot w_i]_d \bmod 2$.

7 How Homomorphic Is This Scheme?

We ran some experiments to get a handle on the degree and number of monomials that the somewhat homomorphic scheme can handle, and to help us choose the parameters. In these experiments we generated key pairs for parameters n (dimension) and t (bit-length), and for each key pair we encrypted many bits, evaluated on the ciphertexts many elementary symmetric polynomials of various degrees and number of variables, decrypted the results, and checked whether or not we got back the same polynomials in the plaintext bits.



m =#-of-variables t =bit-length	$m = 64$	$m = 96$	$m = 128$	$m = 192$	$m = 256$
$t = 64$	13	12	11	11	10
$t = 128$	33	28	27	26	24
$t = 256$	64	76	66	58	56
$t = 384$	64	96	128	100	95

Cells contain the largest supported degree for every m, t combination

Fig. 1. Supported degree vs. number of variables and bit-length of the generating polynomial, all tests were run in dimension $n = 128$

More specifically, for each key pair we tested polynomials on 64 to 256 variables. For every fixed number of variables m we ran 12 tests. In each test we encrypted m bits, evaluated all the elementary symmetric polynomials in these variables (of degree up to m), decrypted the results, and compared them to the results of applying the same polynomials to the plaintext bits. For each setting of m, we recorded the highest degree for which all 12 tests were decrypted to the correct value. We call this the "*largest supported degree*" for those parameters.

In these experiments we used fresh ciphertexts of expected Euclidean length roughly $2 \cdot \sqrt{20} \approx 9$, regardless of the dimension. This was done by choosing each entry of the noise vector $\boldsymbol{u}$ as 0 with probability $1 - \frac{20}{n}$, and as ± 1 with probability $\frac{10}{n}$ each. With that choice, the degree of polynomials that the somewhat-homomorphic scheme could evaluate did not depend on the dimension n: We tested various dimensions from 128 to 2048 with a few settings of t and m, and the largest supported degree was nearly the same in all these dimensions. Thereafter we tested all the other settings only in dimension $n = 128$.

The results are described in Figure 1. As expected, the largest supported degree grows linearly with the bit-length parameter t, and decreases slowly with the number of variables (since more variables means more terms in the polynomial).

These results can be more or less explained by the assumptions that the decryption radius of the secret key is roughly 2^t, and that the noise in an evaluated ciphertext is roughly $c^{\text{degree}} \times \sqrt{\#\text{-of-monomials}}$, where c is close to the Euclidean norm of fresh ciphertexts (i.e., $c \approx 9$). For elementary symmetric polynomials, the number of monomials is exactly $\binom{m}{\text{deg}}$. Hence to handle polynomials of degree deg with m variables, we need to set t large enough so that $2^t \geq c^{\text{deg}} \times \sqrt{\binom{m}{\text{deg}}}$, in order for the noise in the evaluated ciphertexts to still be inside the decryption radius of the secret key.

Trying to fit the data from Figure 1 to this expression, we observe that c is not really a constant, rather it gets slightly smaller when t gets larger. For $t = 64$ we have $c \in [9.14, 11.33]$, for $t = 128$ we have $c \in [7.36, 8.82]$, for $t = 256$ we get $c \in [7.34, 7.92]$, and for $t = 384$ we have $c \in [6.88, 7.45]$. We speculate that this small deviation stems from the fact that the norm of the individual monomials is not exactly c^{deg} but rather has some distribution around that size, and as a result the norm of the sum of all these monomials differs somewhat from $\sqrt{\#\text{-of-monomials}}$ times the expected c^{deg}.

Acknowledgments. We thank Nigel Smart for many excellent comments. We also thank the CRYPTO reviewers for their helpful comments and Tal Rabin, John Gunnels, and Grzegorz Swirszcz for interesting discussions.

References

1. Avanzi, R.M.: Fast evaluation of polynomials with small coefficients modulo an integer. Web document (2005), http://caccioppoli.mac.rub.de/website/papers/trick.pdf
2. Gama, N., Nguyen, P.Q.: Predicting lattice reduction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 31–51. Springer, Heidelberg (2008)
3. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st ACM Symposium on Theory of Computing – STOC 2009, pp. 169–178. ACM, New York (2009)
4. Gentry, C.: Toward basing fully homomorphic encryption on worst-case hardness. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 116–137. Springer, Heidelberg (2010)

5. Gentry, C., Halevi, S.: Implementing gentry's fully-homomorphic encryption scheme. Cryptology ePrint Archive, Report 2010/520 (2010), http://eprint.iacr.org/
6. Goldreich, O., Goldwasser, S., Halevi, S.: Public-key cryptosystems from lattice reduction problems. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 112–131. Springer, Heidelberg (1997)
7. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)
8. Micciancio, D.: Improving lattice based cryptosystems using the hermite normal form. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 126–145. Springer, Heidelberg (2001)
9. Ogura, N., Yamamoto, G., Kobayashi, T., Uchiyama, S.: An improvement of key generation algorithm for gentry's homomorphic encryption scheme. In: Echizen, I., Kunihiro, N., Sasaki, R. (eds.) IWSEC 2010. LNCS, vol. 6434, pp. 70–83. Springer, Heidelberg (2010)
10. Paterson, M.S., Stockmeyer, L.J.: On the number of nonscalar multiplications necessary to evaluate polynomials. SIAM Journal on Computing 2(1), 60–66 (1973)
11. Peikert, C., Rosen, A.: Lattices that admit logarithmic worst-case to average-case connection factors. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing – STOC 2007, pp. 478–487. ACM, New York (2007)
12. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–177. Academic Press, London (1978)
13. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
14. Stehlé, D., Steinfeld, R.: Faster fully homomorphic encryption. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 377–394. Springer, Heidelberg (2010)

Homomorphic Signatures for Polynomial Functions

Dan Boneh* and David Mandell Freeman**

Stanford University
{dabo,dfreeman}@cs.stanford.edu

Abstract. We construct the first homomorphic signature scheme that is capable of evaluating multivariate polynomials on signed data. Given the public key and a signed data set, there is an efficient algorithm to produce a signature on the mean, standard deviation, and other statistics of the signed data. Previous systems for computing on signed data could only handle linear operations. For polynomials of constant degree, the length of a derived signature only depends logarithmically on the size of the data set.

Our system uses ideal lattices in a way that is a "signature analogue" of Gentry's fully homomorphic encryption. Security is based on hard problems on ideal lattices similar to those in Gentry's system.

Keywords: Homomorphic signatures, ideals, lattices.

1 Introduction

While recent groundbreaking work has shown how to compute arbitrary functions on *encrypted* data [17,33,12], far less is known about computing functions on *signed* data.

Informally, the problem of computing on signed data is as follows. Alice has a numerical data set $m_1, \ldots, m_k$ of size k (e.g., final grades in a course with k students). She independently signs each datum m_i, but before signing she augments m_i with a tag and an index. More precisely, Alice signs the triple ("grades", m_i, i) for $i = 1, \ldots, k$ and obtains k independent signatures $\sigma_1, \ldots, \sigma_k$. Here i is the index of m_i in the data set and the tag "grades" serves as a label that names the data set and binds its members together. For convenience we write $\vec{\sigma} := (\sigma_1, \ldots, \sigma_k)$. The data set and the k signatures are stored on some untrusted remote server.

Later, the server is asked to compute authenticated functions of the data, such as the mean or standard deviation of subsets of the data. To compute a function f, the server uses an algorithm $\mathsf{Evaluate}(\mathsf{pk}, \cdot, f, \vec{\sigma})$ that uses $\vec{\sigma}$ and f to derive a signature σ on the triple

$$\big(\text{ "grades", } m := f(m_1, \ldots, m_k), \langle f \rangle \big), \tag{1.1}$$

where $\langle f \rangle$ is an encoding of the function f, i.e., a string that uniquely describes the function. Note that $\mathsf{Evaluate}$ does not need the original messages — it only acts on signatures. Now the pair (m, σ) can be published and anyone can check that the server correctly applied f to the data set by verifying that σ is a signature on the triple (1.1).

* Supported by NSF, DARPA, and AFOSR.
** Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 149–168, 2011.

The derived signature authenticates both the function f and the result of applying f to the data. The pair (m, σ) can be used further to derive signatures on functions of m and other signed data. We give precise definitions of the system's syntax and security below.

Our focus here is on functions that perform arithmetic operations on the data set, such as mean, standard deviation, and other data mining algorithms. Current methods for computing on signed data can handle only *linear* functions [23, 11, 38, 7, 16, 6]. In these systems, given k independently signed vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ defined over some finite field $\mathbb{F}_p$, anyone can compute a signature on any vector $\mathbf{v}$ in the $\mathbb{F}_p$-linear span of $\{\mathbf{v}_1, \ldots, \mathbf{v}_k\}$, and no one without the secret key can compute a valid signature on a vector $\mathbf{v}$ outside this span. The original motivation for these linear schemes comes from the *network coding* routing mechanism [14].

In this paper we present the first signature system that supports computing *polynomial functions* on signed data. Specifically, our system supports multivariate polynomials of bounded degree. For a constant-degree polynomial with bounded coefficients, the length of a derived signature only depends logarithmically on the size of the data set. Thus, for example, given a signed data set as input, anyone can compute a short signature on the mean, standard deviation, least squares fit, and other functions of arbitrary subsets of the data. Note that computing standard deviation requires only a quadratic multivariate polynomial; other applications, discussed in Section 2.4, may require cubic or higher degree polynomials. While our system intrinsically computes on data defined over a finite field $\mathbb{F}_p$, it can be used to compute on data defined over the integers by choosing a sufficiently large field size p.

Our system's functionality and security derive from properties of certain *integer lattices*. As a "warm-up" to our main result, we describe in Section 4 a linearly homomorphic scheme built from random integer lattices that uses the same underlying ideas as our polynomial scheme. Interestingly, this construction gives a homomorphic system over $\mathbb{F}_2$ that allows linear functions of many more inputs than the best previous such system [6]. In Section 6 we show how replacing the random lattices in the linear scheme with *ideal lattices* leads to a polynomially homomorphic scheme — our main result.

We note that a trivial solution to computing on signed data is to have the server send the entire data set to the client along with all the signatures and have the client compute the function itself. With our constructions only the output of the function is sent to the client along with a short signature. Beyond saving bandwidth, this approach also limits the amount of information revealed to the client about the data set, as formalized in Section 2.2.

Related work. Before delving into the details of our construction, we mention the related work on non-interactive proofs [26, 37, 20] where the prover's goal is to output a certificate that convinces the verifier that a certain statement is correct. Micali's computationally sound (CS) proofs [26] can solve the problem discussed above as follows: Alice signs the pair (τ, D) where D is a data set and τ is a short tag used to name D. She sends D, τ and the signature σ to the server. Later, for some function f, the server publishes $\big(\tau, \sigma,\ t := f(D),\ \pi\big)$ where π is a short proof that there exists a data set D such that $t = f(D)$ and that σ is a valid signature by Alice on (τ, D). This tuple convinces anyone that t is the result of applying f to the original data set D labeled τ by Alice. Security is proved using Valiant's witness extractor [37] to extract a signature

forgery from a cheating server. The construction of π uses the full machinery of the PCP theorem and soundness is in the random oracle model. Note that computational soundness is sufficient in these settings since the server is given signed data and is therefore already assumed to be computationally bounded.

Our approach eliminates the proof π. The server only publishes $(\tau, \sigma',\ t := f(D))$, where σ' is derived from σ and authenticates both t and f. Constructing σ' is straightforward and takes about the same amount of work as computing $f(D)$. Moreover, anyone can further compute $t' := g(t) = g(f(D))$ for some function g and use σ' to derive a signature on t' and the function $g(f(\cdot))$. While further computation can also be done with CS proofs [37], it is much simpler with homomorphic signatures.

More recently Goldwasser, Kalai, and Rothblum [20] and Gennaro, Gentry, and Parno [15] show how to outsource computation securely. In both [15] and [20] (in the non-interactive setting) the interaction between the server and the client is tailored to the client and the client uses a secret key to verify the results. In our settings the server constructs a publicly verifiable signature on the result and anyone can verify that signature using Alice's public key.

We also mention another line of related work that studies "redactable" signatures [35, 22, 21, 4, 29, 28, 10, 9, 8, 1, 31]. These schemes have the property that given a signature on a message, anyone can derive signatures on subsets of the message. Our focus here is quite different — we look at computing arithmetic functions on independently authenticated data, rather than computing on a subset of a single message. We also require that the derived signature explicitly authenticate the computed function f.

1.1 Overview of Our Techniques

The intersection method. Our system uses two n-dimensional integer lattices Λ_1 and Λ_2. The lattice Λ_1 is used to sign the data (e.g., a student's grade or the result of a computation), while the lattice Λ_2 is used to sign a description of the function f applied to the data. The message space for these signatures is $\mathbb{Z}^n/\Lambda_1$, which for the lattices we consider is simply a vector space over the finite field $\mathbb{F}_p$ for some prime p.

A signature in our system is a short vector σ in $\mathbb{Z}^n$ in the intersection of $\Lambda_1 + \mathbf{u}_1$ and $\Lambda_2 + \mathbf{u}_2$ for certain $\mathbf{u}_1, \mathbf{u}_2 \in \mathbb{Z}^n$. In other words, we have $\sigma = \mathbf{u}_1 \bmod \Lambda_1$ and $\sigma = \mathbf{u}_2 \bmod \Lambda_2$. Loosely speaking, this single signature σ "binds" $\mathbf{u}_1$ and $\mathbf{u}_2$ — an attacker cannot generate a new *short* vector σ' from σ such that $\sigma = \sigma' \mod \Lambda_1$ but $\sigma \neq \sigma' \mod \Lambda_2$. We refer to this method of jointly signing two vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ as the *intersection method*.

More precisely, let τ be a tag, m be a message, and $\langle f \rangle$ be an encoding of a function f. A signature σ on a triple $(\tau, m, \langle f \rangle)$ is a short vector in $\mathbb{Z}^n$ satisfying $\sigma = m \bmod \Lambda_1$ and $\sigma = \omega_\tau(\langle f \rangle) \bmod \Lambda_2$. Here ω_τ is a hash function defined by the tag τ that maps (encodings of) functions to vectors in $\mathbb{Z}^n/\Lambda_2$. This ω_τ not only must preserve the homomorphic properties of the system, but also must enable simulation against a chosen-message adversary. Note that the Λ_1 component of the signature σ is essentially the same as a Gentry-Peikert-Vaikuntanathan signature [19] on the (unhashed) message m.

It is not difficult to see that these signatures are *additively* homomorphic. That is, let σ_1 be a signature on $(\tau, m_1, \langle f_1 \rangle)$ and let σ_2 be a signature on $(\tau, m_2, \langle f_2 \rangle)$. With an appropriate hash function ω_τ, we can ensure that $\sigma_1 + \sigma_2$ is a signature on

$\big(\tau,\ m_1 + m_2,\ \langle f_1 + f_2 \rangle\ \big)$. If we set $\Lambda_1 = (2\mathbb{Z})^n$, we obtain a more efficient linearly homomorphic signature over $\mathbb{F}_2$ than previously known [6].

Now let $g \in \mathbb{Z}[x]$ be a polynomial of degree n and let R be the ring $\mathbb{Z}[x]/(g)$. Then R is isomorphic to $\mathbb{Z}^n$ and ideals in R correspond to integer lattices in $\mathbb{Z}^n$ under the "coefficient embedding." We choose our two lattices Λ_1 and Λ_2 to be prime ideals $\mathfrak{p}$ and $\mathfrak{q}$ in R and a signature on the triple (τ, m, f) to be a short element in R such that $\sigma = m \bmod \mathfrak{p}$ and $\sigma = \omega_\tau(\langle f \rangle) \bmod \mathfrak{q}$. With this setup, let σ_1 and σ_2 be signatures on $(\tau, m_1, \langle f_1 \rangle)$ and $(\tau, m_2, \langle f_2 \rangle)$ respectively. Then for an appropriate hash function ω_τ,

$$\begin{aligned} \sigma_1 + \sigma_2 \quad &\text{is a signature on} \quad \big(\ \tau,\ m_1 + m_2,\ \langle f_1 + f_2 \rangle\ \big) \quad \text{and} \\ \sigma_1 \cdot \sigma_2 \quad &\text{is a signature on} \quad \big(\ \tau,\ m_1 \cdot m_2,\ \langle f_1 \cdot f_2 \rangle\ \big)\ . \end{aligned}$$

More generally, we can evaluate any bounded degree polynomial with small coefficients on signatures. In particular, the quadratic polynomial $v(m_1, \ldots, m_k) := \sum_{i=1}^{k} (k m_i - \sum_{i=1}^{k} m_i)^2$ that computes a fixed multiple of the variance can easily be evaluated this way. Anyone can calculate the standard deviation from $v(m_1, \ldots, m_k)$ and k by taking a square root and dividing by k.

Our use of ideal lattices is a signature analogue of Gentry's "somewhat homomorphic" encryption system [17]. Ideal lattices also appear in the lattice-based public key encryption schemes of Stehle, Steinfeld, Tanaka, and Xagawa [34] and Lyubashevsky, Peikert, and Regev [25] and in the hash functions of Lyubashevsky and Micciancio [24].

Unforgeability. Loosely speaking, a forgery under a chosen message attack is a valid signature σ on a triple $(\tau, m, \langle f \rangle)$ such that $m \neq f(m_1, \ldots, m_k)$, where $m_1, \ldots, m_k$ is the data set signed using tag τ. We show that a successful forger can be used to solve the Small Integer Solution (SIS) problem in the lattice Λ_2, which for random q-ary lattices and suitable parameters is as hard as standard worst-case lattice problems [27]. When Λ_2 is an ideal lattice we can then use the ideal structure to obtain a solution to the Shortest Independent Vectors Problem (SIVP) for the (average case) distribution of lattices produced by our key generation algorithm. As is the case with existing linearly homomorphic signature schemes, our security proofs are set in the random oracle model, where the random oracle is used to simulate signatures for a chosen message attacker.

Privacy. For some applications it is desirable that derived signatures be private. That is, if σ is a signature on a message $m := f(m_1, \ldots, m_k)$ derived from signatures on messages $m_1, \ldots, m_k$, then σ should reveal no information about $m_1, \ldots, m_k$ beyond what is revealed by m and f. Using similar techniques to those in [6], it is not difficult to show that our linearly homomorphic signatures satisfy a privacy property (also defined in [6]) called *weak context hiding*. Demonstrating this property amounts to proving that the distribution obtained by summing independent discrete Gaussians depends only on the coset of the sum.

Interestingly, we can show that our polynomially homomorphic signature is not private. It is an open problem either to design a polynomially homomorphic signature scheme that is also private, or to modify our scheme to make it private.

Length efficiency. We require that derived signatures be not much longer than the original signatures from which they were derived; we define this requirement precisely in Section 2.3. All of our constructions are length efficient.

2 Homomorphic Signatures: Definitions and Applications

Informally, a homomorphic signature scheme consists of the usual algorithms KeyGen, Sign, Verify as well as an additional algorithm Evaluate that "translates" functions on messages to functions on signatures. If $\vec{\sigma}$ is a valid set of signatures on messages $\vec{m}$, then $\mathsf{Evaluate}(f,\ \vec{\sigma})$ should be a valid signature for $f(\vec{m})$.

To prevent mixing of data from different data sets when evaluating functions, the Sign, Verify, and Evaluate algorithms take an additional short "tag" as input. The tag serves to bind together messages from the same data set. One could avoid the tag by requiring that a new public key be generated for each data set, but simply requiring a new tag for each data set is more convenient.

Formally, a homomorphic signature scheme is as follows:

Definition 2.1. A *homomorphic signature scheme* is a tuple of probabilistic, polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Evaluate})$ as follows:

- $\mathsf{Setup}(1^n, k)$. Takes a security parameter n and a maximum data set size k. Outputs a public key pk and a secret key sk. The public key pk defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of "admissible" functions $f\colon \mathcal{M}^k \to \mathcal{M}$.
- $\mathsf{Sign}(\mathsf{sk}, \tau, m, i)$. Takes a secret key sk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathcal{M}$ and an index $i \in \{1, \ldots, k\}$, and outputs a signature $\sigma \in \Sigma$.
- $\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, f)$. Takes a public key pk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathcal{M}$, a signature $\sigma \in \Sigma$, and a function $f \in \mathcal{F}$, and outputs either 0 (reject) or 1 (accept).
- $\mathsf{Evaluate}(\mathsf{pk}, \tau, f, \vec{\sigma})$. Takes a public key pk, a tag $\tau \in \{0,1\}^n$, a function $f \in \mathcal{F}$, and a tuple of signatures $\vec{\sigma} \in \Sigma^k$, and outputs a signature $\sigma' \in \Sigma$.

Let $\pi_i\colon \mathcal{M}^k \to \mathcal{M}$ be the function $\pi_i(m_1, \ldots, m_k) = m_i$ that projects onto the ith component. We require that $\pi_1, \ldots, \pi_k \in \mathcal{F}$ for all pk output by $\mathsf{Setup}(1^n, k)$.

For correctness, we require that for each $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}(1^n, k)$, we have:

1. For all tags $\tau \in \{0,1\}^n$, all $m \in \mathcal{M}$, and all $i \in \{1, \ldots, k\}$, if $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m, i)$, then with overwhelming probability $\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, \pi_i) = 1$.
2. For all $\tau \in \{0,1\}^n$, all tuples $\vec{m} = (m_1, \ldots, m_k) \in \mathcal{M}^k$, and all functions $f \in \mathcal{F}$, if $\sigma_i \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau, m_i, i)$ for $i = 1, \ldots, k$, then with overwhelming probability

$$\mathsf{Verify}\big(\mathsf{pk},\ \tau,\ f(\vec{m}),\ \mathsf{Evaluate}\big(\mathsf{pk}, \tau, f, (\sigma_1, \ldots, \sigma_k)\big),\ f\big) = 1.$$

We say that a signature scheme as above is *$\mathcal{F}$-homomorphic*.

While the Evaluate algorithm in our schemes can take as input derived signatures themselves produced by Evaluate, doing so for a large number of iterations may eventually reach a point where the input signatures to Evaluate are valid, but the output signature is not. Therefore, to simplify the discussion we limit the correctness property to require only that Evaluate produce valid output when given as input signatures $\vec{\sigma}$ produced by the Sign algorithm.

For ease of exposition we describe our systems as if all data sets consist of exactly k items. It is straightforward to apply the systems to data sets of size ℓ for any $\ell \leq k$, simply by interpreting a function on ℓ variables as a function on k variables that ignores the last $k-\ell$ inputs. The definitions of unforgeability and privacy below can be adapted accordingly.

2.1 Unforgeability

The security model for homomorphic signatures allows an adversary to make adaptive signature queries on data sets of his choosing, each containing (up to) k messages, with the signer randomly choosing the tag τ for each data set queried. Eventually the adversary produces a message-signature pair (m^*, σ^*) as well as an admissible function f and a tag τ^*. The winning condition captures the fact that there are two distinct types of forgeries. In a *type 1 forgery*, the pair (m^*, σ^*) verifies for some data set *not* queried to the signer; this corresponds to the usual notion of signature forgery. In a *type 2 forgery*, the pair (m^*, σ^*) verifies for some data set that *was* queried to the signer, but for which m^* does not equal f applied to the messages queried; in other words, the signature authenticates m^* as $f(\vec{m})$ but in fact this is not the case.

Our security model requires that all data in a data set be signed at once; that is, the adversary cannot request signatures on new messages after seeing signatures on other messages in the same data set.

Definition 2.2. A homomorphic signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Evaluate})$ is *unforgeable* if for all k the advantage of any probabilistic, polynomial-time adversary $\mathcal{A}$ in the following game is negligible in the security parameter n:

Setup: The challenger runs $\mathsf{Setup}(1^n, k)$ to obtain $(\mathsf{pk}, \mathsf{sk})$ and gives pk to $\mathcal{A}$. The public key defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of admissible functions $f\colon \mathcal{M}^k \to \mathcal{M}$.

Queries: Proceeding adaptively, $\mathcal{A}$ specifies a sequence of data sets $\vec{m}_i \in \mathcal{M}^k$. For each i, the challenger chooses τ_i uniformly from $\{0,1\}^n$ and gives to $\mathcal{A}$ the tag τ_i and the signatures $\sigma_{ij} \leftarrow \mathsf{Sign}(\mathsf{sk}, \tau_i, m_{ij}, j)$ for $j = 1, \ldots, k$.

Output: $\mathcal{A}$ outputs a tag $\tau^* \in \{0,1\}^n$, a message $m^* \in \mathcal{M}$, a function $f \in \mathcal{F}$, and a signature $\sigma^* \in \Sigma$.

The adversary *wins* if $\mathsf{Verify}(\mathsf{pk}, \tau^*, m^*, \sigma^*, f) = 1$ and either

(1) $\tau^* \neq \tau_i$ for all i (a *type 1 forgery*), or
(2) $\tau^* = \tau_i$ for some i but $m^* \neq f(\vec{m}_i)$ (a *type 2 forgery*).

The *advantage* of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the security game.

2.2 Privacy

As in [6] we define privacy for homomorphic signatures using a variation of a definition of Brzuska et al. [9]. The definition captures the idea that given signatures on a number

of messages derived from two different data sets, the attacker cannot tell which data set the derived signatures came from, and furthermore that this property holds even if the secret key is leaked. We call signatures with this privacy property *weakly context hiding*. The reason for "weak" is that we assume the original signatures on the data set are not public. The concept is similar to that of witness indistinguishability [13], where in our setting we treat the original data set as the witness.

Ahn et al. [1] define a stronger notion of privacy, called *strong context hiding*, that requires derived signatures to be distributed as independent fresh signatures on the same message; this requirement ensures privacy even if the original signatures are exposed.

Definition 2.3. A homomorphic signature scheme $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Evaluate})$ is *weakly context hiding* if for all k, the advantage of any probabilistic, polynomial-time adversary $\mathcal{A}$ in the following game is negligible in the security parameter n:

Setup: The challenger runs $\mathsf{Setup}(1^n, k)$ to obtain $(\mathsf{pk}, \mathsf{sk})$ and gives pk and sk to $\mathcal{A}$. The public key defines a message space $\mathcal{M}$, a signature space Σ, and a set $\mathcal{F}$ of admissible functions $f\colon \mathcal{M}^k \to \mathcal{M}$.

Challenge: $\mathcal{A}$ outputs $(\vec{m}_0^*, \vec{m}_1^*, f_1, \ldots, f_s)$ with $\vec{m}_0^*, \vec{m}_1^* \in \mathcal{M}^k$. The functions $f_1, \ldots, f_s$ are in $\mathcal{F}$ and satisfy

$$f_i(\vec{m}_0^*) = f_i(\vec{m}_1^*) \qquad \text{for all } i = 1, \ldots, s.$$

In response, the challenger generates a random bit $b \in \{0,1\}$ and a random tag $\tau \in \{0,1\}^n$. It signs the messages in $\vec{m}_b^*$ using the tag τ to obtain a vector $\vec{\sigma}$ of k signatures. Next, for $i = 1, \ldots, s$ the challenger computes a signature $\sigma_i := \mathsf{Evaluate}(\mathsf{pk}, \tau, f_i, \vec{\sigma})$ on $f_i(\vec{m}_b^*)$. It sends the tag τ and the signatures $\sigma_1, \ldots, \sigma_s$ to $\mathcal{A}$. Note that the functions $f_1, \ldots, f_s$ can be output adaptively after $\vec{m}_0^*, \vec{m}_1^*$ are output.

Output: $\mathcal{A}$ outputs a bit b'.

The adversary $\mathcal{A}$ wins the game if $b = b'$. The *advantage* of $\mathcal{A}$ is the probability that $\mathcal{A}$ wins the game.

Winning the weak context hiding game means that the attacker was able to determine whether the challenge signatures were derived from signatures on $\vec{m}_0^*$ or from signatures on $\vec{m}_1^*$. We say that the signature scheme is *s-weakly context hiding* if the attacker cannot win the privacy game after seeing at most s signatures derived from $\vec{m}_0^*$ or $\vec{m}_1^*$.

2.3 Length Efficiency

We say that a homomorphic signature scheme is *length efficient* if for a fixed security parameter n, the length of derived signatures depends only logarithmically on the size k of the data set. More precisely, we have the following:

Definition 2.4. Let $\mathcal{S} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Evaluate})$ be a homomorphic signature scheme. We say that $\mathcal{S}$ is *length efficient* if there is some function $\mu\colon \mathbb{N} \to \mathbb{R}$ such that for all $(\mathsf{pk}, \mathsf{sk})$ output by $\mathsf{Setup}(1^n, k)$, all $\vec{m} = (m_1, \ldots, m_k) \in \mathcal{M}^k$, all tags $\tau \in \{0,1\}^n$, and all functions $f \in \mathcal{F}$, if

$$\sigma_i \leftarrow \mathsf{Sign}(\mathsf{pk}, \tau, m_i, i) \quad \text{for } i = 1, \ldots, k,$$

then for all $k > 0$, the derived signature $\sigma := \mathsf{Evaluate}\big(\mathsf{pk}, \tau, f, (\sigma_1, \ldots, \sigma_k)\big)$ has bit length at most $\mu(n) \cdot \log k$ with overwhelming probability.

2.4 Applications

Before describing our constructions we first examine a few applications of computing on signed data. In the Introduction we discussed applications to computing statistics on signed data, and in particular the examples of mean and standard deviation. Here we discuss more complex data mining algorithms.

Least squares fits. Recall that given an integer $d \geq 0$ and a data set $\{(x_i, y_i)\}_{i=1}^k$ consisting of k pairs of real numbers, the *degree d least squares fit* is a polynomial $f \in \mathbb{R}[x]$ of degree d that minimizes the quantity $\sum_{i=1}^k (y_i - f(x_i))^2$. The vector of coefficients of f is denoted by $\vec{f}$ and is given by the formula

$$\vec{f} = (X^T X)^{-1} X^T \vec{y} \in \mathbb{R}^{d+1},$$

where $X \in \mathbb{R}^{k \times (d+1)}$ is the *Vandermonde matrix* of the x_i, whose jth column is the vector $(x_1^{j-1}, \ldots, x_k^{j-1})$, and $\vec{y}$ is the column vector $(y_1, \ldots, y_k)$. The degree d is usually small, e.g. $d = 1$ for a least squares fit with a line.

Using homomorphic signatures, a server can be given a set of individually signed data points and derive from it a signature on the least squares fit f (or more precisely, a signature on the vector of coefficients $\vec{f}$). If the signature is length efficient, then for fixed d the length of the derived signature depends only logarithmically on the number of data points k. If the signatures are private, then the derived signature on f reveals nothing about the original data set beyond what is revealed by f.

We consider two types of data sets. In the first type, the x-coordinates are universal constants in $\mathbb{Z}$ and need not be signed. For example, the data set might contain the temperature on each day of the year, in which case the x-coordinates are simply the days of the year and need not be explicitly included in the data. Only the y-coordinates are signed. Then the least squares fit is simply a linear function of $\vec{y}$, namely $\vec{f} := A \cdot \vec{y}$ for some fixed matrix A. The signature on $\vec{f}$ can thus be derived using any *linearly* homomorphic signature scheme. To handle fractional entries in A we can pre-multiply A by a known scalar to cancel denominators.

The second type of data set is one in which both the x-coordinate and the y-coordinate are signed. More precisely, an integer data set $\{(x_i, y_i)\}_{i=1}^k$ is signed by signing all $2k$ values separately (but with the same tag) to obtain $2k$ signatures. The server is given the data set and these $2k$ signatures. In the full version of this paper [5] we show that a homomorphic signature scheme for polynomials of degree d^2+d+1 over the integers is sufficient for deriving a signature on the least squares fit. In particular, for a least squares fit using a line it suffices for the signature to be homomorphic for cubic polynomials.

In summary, linearly homomorphic signatures are sufficient when the x-coordinates are absolute constants and polynomially homomorphic signatures are needed for general data sets.

More advanced data mining. If we had fully homomorphic signatures (i.e., supporting arbitrary computation on signed data), then an untrusted server could run more

complex data mining algorithms on the given data set. For example, given a signed data set, the server could publish a signed decision tree (e.g., as generated by the ID3 algorithm [32]). Length efficiency means that the length of the resulting signature depends only logarithmically on the size of the data set. If the signatures were private, then publishing the signed decision tree would leak no other information about the original data set.

3 Preliminaries

Notation. For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q. If q is prime, $\mathbb{Z}_q$ is a field and is denoted by $\mathbb{F}_q$. We let $\mathbb{Z}_q^{\ell \times n}$ denote the set of $\ell \times n$ matrices with entries in $\mathbb{Z}_q$. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$. The function $\lg x$ is the base 2 logarithm of x.

Lattices. An *n-dimensional lattice* is a full-rank discrete subgroup of $\mathbb{R}^n$. Standard results on lattices that we use appear in the full version of this paper [5]. Here we note briefly that our schemes will use an algorithm $\mathsf{SamplePre}$ [19, Theorem 5.9] that takes as input a basis $\mathbf{T}$ of an n-dimensional lattice Λ, a parameter ν, and a vector $\mathbf{t} \in \mathbb{Z}^n$, and outputs a vector in the coset $\Lambda + \mathbf{t}$ sampled from a Gaussian distribution. $\mathsf{SamplePre}$ is itself built from an algorithm $\mathsf{SampleGaussian}$ [19, Theorem 4.1] that outputs a vector in the lattice Λ sampled from a Gaussian distribution.

Our linearly homomorphic schemes will use "q-ary" lattices defined as follows: for any integer $q \geq 2$ and any $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$, we define $\Lambda_q^{\perp}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^n : \mathbf{A} \cdot \mathbf{e} = \mathbf{0} \bmod q\}$. Our schemes will use an algorithm $\mathsf{TrapGen}$ [3, Theorem 3.2] that samples an (almost) uniformly random matrix $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$ along with a "short" basis for $\Lambda_q^{\perp}(\mathbf{A})$.

Complexity assumption. We define a generalization of the now-standard *Small Integer Solution* (SIS) problem, which is to find a short nonzero vector in a certain class of lattices.

Definition 3.1. Let $\mathcal{L} = \{\mathcal{L}_n\}$ be a distribution ensemble of integer lattices, where lattices in $\mathcal{L}_n$ have dimension n. An instance of the $\mathcal{L}$-$\mathsf{SIS}_{n,\beta}$ problem is a lattice $\Lambda \leftarrow \mathcal{L}_n$. A solution to the problem is a nonzero vector $\mathbf{v} \in \Lambda$ with $\|\mathbf{v}\| \leq \beta$.

If $\mathcal{B}$ is an algorithm that takes as input a lattice Λ, we define the *advantage* of $\mathcal{B}$, denoted $\mathcal{L}\text{-}\mathsf{SIS}\text{-}\mathrm{Adv}[\mathcal{B}, (n, \beta)]$, to be the probability that $\mathcal{B}$ outputs a solution to an $\mathcal{L}$-$\mathsf{SIS}_{n,\beta}$ problem instance Λ chosen according to the distribution $\mathcal{L}_n$.

We say that the $\mathcal{L}$-$\mathsf{SIS}_{n,\beta}$ problem is *infeasible* if for all polynomial-time algorithms $\mathcal{B}$, the advantage $\mathcal{L}\text{-}\mathsf{SIS}\text{-}\mathrm{Adv}[\mathcal{B}, (n, \beta)]$ is a negligible function of n.

When $\mathcal{L}_n$ consists of $\Lambda_q^{\perp}(\mathbf{A})$ for uniformly random $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$, the $\mathcal{L}$-$\mathsf{SIS}_{n,\beta}$ problem is the standard $\mathsf{SIS}_{q,n,\beta}$ problem defined by Micciancio and Regev [27]. For this distribution of lattices, an algorithm that solves the $\mathcal{L}$-$\mathsf{SIS}_{n,\beta}$ problem can be used to solve worst-case problems on arbitrary ℓ-dimensional lattices [27, §5].

4 Homomorphic Signatures for Linear Functions over Small Fields

As a "warm-up" to our polynomially homomorphic scheme, we describe a signature scheme that can authenticate any linear function of signed vectors defined over small fields $\mathbb{F}_p$. Previous constructions can only achieve this functionality for vectors defined over large fields [11,7] or for a small number of vectors [6]. In particular, our scheme easily accommodates binary data ($p = 2$). Linearly homomorphic signatures over $\mathbb{F}_2$ are an example of a primitive that can be built from lattices, but cannot currently be built from discrete-log or RSA-type assumptions. In the full version of this paper [5] we describe a variant of the scheme in which the data can take values in large fields $\mathbb{F}_p$.

Security is based on the SIS problem on q-ary lattices for some prime q; Micciancio and Regev [27], building on the work of Ajtai [2], show that this problem is as hard as standard worst-case problems on arbitrary lattices of dimension approximately $n/\lg q$. The system in this section is only secure for small p, specifically $p = \mathrm{poly}(n)$ with $p \leq \sqrt{q}/nk$ for data sets of size k.

Overview of the scheme. Since our system builds on the "hash-and-sign" signatures of Gentry, Peikert, and Vaikuntanathan [19], let us recall how GPV signatures work in an abstract sense. The public key is a lattice $\Lambda \subset \mathbb{Z}^n$ and the secret key is a short basis of Λ. To sign a message m, the secret key holder hashes m to an element $H(m) \in \mathbb{Z}^n/\Lambda$ and samples a short vector σ from the coset of Λ defined by $H(m)$. To verify σ, one checks that σ is short and that $\sigma \bmod \Lambda = H(m)$.

Recall that in a homomorphic signature scheme we wish to authenticate triples $(\tau, m, \langle f \rangle)$, where τ is a "tag" attached to a data set, m is a message in $\mathbb{F}_p^n$, and $\langle f \rangle$ is an encoding of a function f acting on k-tuples of messages. We encode a linear function $f : (\mathbb{F}_p^n)^k \to \mathbb{F}_p^n$ defined by $f(m_1, \ldots, m_k) = \sum_{i=1}^k c_i m_i$ by interpreting the c_i as integers in $(-p/2, p/2]$ and defining $\langle f \rangle := (c_1, \ldots, c_k) \in \mathbb{Z}^k$.

To authenticate both the message and the function as well as bind them together, we compute a single GPV signature that is *simultaneously* a signature on the (unhashed) message $m \in \mathbb{F}_p^n$ and a signature on a hash of $\langle f \rangle$.

This dual-role signature is computed via what we call the "intersection method," which works as follows. Let Λ_1 and Λ_2 be n-dimensional integer lattices with $\Lambda_1 + \Lambda_2 = \mathbb{Z}^n$. Suppose $m \in \mathbb{Z}^n/\Lambda_1$ is a message and ω_τ is a hash function (depending on the tag τ) that maps encodings of functions f to elements of $\mathbb{Z}^n/\Lambda_2$. Since the message m defines a coset of Λ_1 in $\mathbb{Z}^n$ and the hash $\omega_\tau(\langle f \rangle)$ defines a coset of Λ_2 in $\mathbb{Z}^n$, by the Chinese remainder theorem the pair $\big(\, m,\ \omega_\tau(\langle f \rangle) \,\big)$ defines a unique coset of $\Lambda_1 \cap \Lambda_2$ in $\mathbb{Z}^n$. We can thus use a short basis of $\Lambda_1 \cap \Lambda_2$ to compute a short vector in this coset; i.e., a short vector σ with the property that $\sigma \bmod \Lambda_1 = m$ and $\sigma \bmod \Lambda_2 = \omega_\tau(\langle f \rangle)$. The vector σ is a signature on $(\tau, m, \langle f \rangle)$.

The $\mathsf{Sign}(\mathsf{sk}, \tau, m, i)$ algorithm uses the procedure above to generate a fresh signature on the triple $(\tau, m, \langle \pi_i \rangle)$ where π_i is the ith *projection function* defined by $\pi_i(m_1, \ldots, m_k) = m_i$ and encoded as $\langle \pi_i \rangle = \mathbf{e}_i$, the ith unit vector in $\mathbb{Z}^k$.

The homomorphic property is now obtained as follows. To authenticate the linear combination $m = \sum_{i=1}^k c_i m_i$ for integers c_i, we compute the signature $\sigma := \sum_{i=1}^k c_i \sigma_i$. If k and p are sufficiently small, then σ is a short vector. Furthermore, we have

$$\sigma \bmod \Lambda_1 = \sum_{i=1}^{k} c_i m_i = m\,, \quad \text{and}$$
$$\sigma \bmod \Lambda_2 = \sum_{i=1}^{k} c_i \omega_\tau(\langle \pi_i \rangle) = \sum_{i=1}^{k} c_i \omega_\tau(\mathbf{e}_i).$$

Now suppose that ω_τ is linear, namely $\sum_{i=1}^{k} c_i \omega_\tau(\mathbf{e}_i) = \omega_\tau\big((c_1, \ldots, c_k)\big)$ for all $c_1, \ldots, c_k$ in $\mathbb{Z}$. Then since $(c_1, \ldots, c_k)$ is exactly the encoding of the function f defined by $f(m_1, \ldots, m_k) = \sum_{i=1}^{k} c_i m_i$, the signature σ authenticates both the message m and the fact that it was computed correctly (i.e., via f) from the original messages $m_1, \ldots, m_k$.

The linearly homomorphic scheme. We now describe the scheme.

$\mathsf{Setup}(1^n, k)$. On input a security parameter n and a maximum data set size k, do the following:

1. Choose two primes $p, q = \mathrm{poly}(n)$ with $q \geq (nkp)^2$. Define $\ell := \lfloor n/6 \log q \rfloor$.
2. Set $\Lambda_1 := p\mathbb{Z}^n$.
3. Use $\mathsf{TrapGen}(q, \ell, n)$ to generate a matrix $\mathbf{A} \in \mathbb{F}_q^{\ell \times n}$ along with a short basis $\mathbf{T}_q$ of $\Lambda_q^\perp(\mathbf{A})$. Define $\Lambda_2 := \Lambda_q^\perp(\mathbf{A})$ and $\mathbf{T} := p \cdot \mathbf{T}_q$.
4. Set $\nu := p \cdot \sqrt{n \log q} \cdot \log n$.
5. Let $H\colon \{0,1\}^* \to \mathbb{F}_q^\ell$ be a hash function (modeled as a random oracle).
6. Output the public key $\mathsf{pk} := (\Lambda_1, \Lambda_2, \nu, k, H)$ and the secret key $\mathsf{sk} = \mathbf{T}$.

The public key pk defines the following system parameters:

- The message space is $\mathbb{F}_p^n$ and signatures are short vectors in $\mathbb{Z}^n$.
- The set of admissible functions $\mathcal{F}$ is all $\mathbb{F}_p$-linear functions on k-tuples of messages in $\mathbb{F}_p^n$.
- For a function $f \in \mathcal{F}$ defined by $f(m_1, \ldots, m_k) = \sum_{i=1}^{k} c_i m_i$, we encode f by interpreting the c_i as integers in $(-p/2, p/2]$ and defining $\langle f \rangle = (c_1, \ldots, c_k) \in \mathbb{Z}^k$.
- To evaluate the hash function ω_τ on an encoded function $\langle f \rangle = (c_1, \ldots, c_k) \in \mathbb{Z}^k$, do the following:
  (a) For $i = 1, \ldots, k$, compute $\alpha_i \leftarrow H(\tau \| i)$ in $\mathbb{F}_q^\ell$.
  (b) Define $\omega_\tau(\langle f \rangle) := \sum_{i=1}^{k} c_i \alpha_i \in \mathbb{F}_q^\ell$.

$\mathsf{Sign}(\mathsf{sk}, \tau, m, i)$. On input a secret key sk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, and an index i, do:

1. Compute $\alpha_i := H(\tau \| i) \in \mathbb{F}_q^\ell$. Then, by definition, $\omega_\tau(\langle \pi_i \rangle) = \alpha_i$.
2. Compute $\mathbf{t} \in \mathbb{Z}^n$ such that $\mathbf{t} \bmod p = m$ and $\mathbf{A} \cdot \mathbf{t} \bmod q = \alpha_i$.
3. Output $\sigma \leftarrow \mathsf{SamplePre}(\Lambda_1 \cap \Lambda_2, \mathbf{T}, \mathbf{t}, \nu) \in (\Lambda_1 \cap \Lambda_2) + \mathbf{t}$.

$\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, f)$. On input a public key pk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p^n$, a signature $\sigma \in \mathbb{Z}^n$, and a function $f \in \mathcal{F}$, do:

1. If all of the following conditions hold, output 1 (accept); otherwise output 0 (reject):
   (a) $\|\sigma\| \leq k \cdot \frac{p}{2} \cdot \nu\sqrt{n}$.
   (b) $\sigma \bmod p = m$.
   (c) $\mathbf{A} \cdot \sigma \bmod q = \omega_\tau(\langle f \rangle)$.

$\mathsf{Evaluate}(\mathsf{pk}, \tau, f, \vec{\sigma})$. On input a public key pk, a tag $\tau \in \{0,1\}^n$, a function $f \in \mathcal{F}$ encoded as $\langle f \rangle = (c_1, \ldots, c_k) \in \mathbb{Z}^k$, and a tuple of signatures $\sigma_1, \ldots, \sigma_k \in \mathbb{Z}^n$, output $\sigma := \sum_{i=1}^{k} c_i \sigma_i$.

In the full version of this paper [5], we show that this linearly homomorphic signature scheme is correct with overwhelming probability and that it is length efficient; i.e., the bit length of a derived signature depends logarithmically on the data set size k.

Unforgeability. We will show that an adversary that forges a signature for the linearly homomorphic scheme can be used to compute a short vector in the lattice Λ_2 chosen in Step 3 of Setup. By [3, Theorem 3.2], the distribution of matrices $\mathbf{A}$ used to define Λ_2 is statistically close to uniform over $\mathbb{F}_q^{\ell \times n}$. Thus the distribution of lattices Λ_2 output by Setup is statistically close to the distribution of challenges for the $\mathsf{SIS}_{q,n,\beta}$ problem (for any β). Our security theorem is as follows.

Theorem 4.1. *If $\mathsf{SIS}_{q,n,\beta}$ is infeasible for $\beta = k \cdot p^2 \cdot n \log n \sqrt{\log q}$, then the linearly homomorphic signature scheme above is unforgeable in the random oracle model.*

Sketch of Proof. Let $\mathcal{A}$ be an adversary that plays the security game of Definition 2.1. Given a challenge lattice $\Lambda_2 := \Lambda_q^\perp(\mathbf{A})$ for $\mathbf{A} \in \mathbb{Z}_q^{\ell \times n}$, we simulate the Sign algorithm on input (τ, m, i) for random τ by sampling a short vector σ from a Gaussian distribution on $\Lambda_1 + m$ and defining $H(\tau \| i) := \mathbf{A} \cdot \sigma \bmod q$. Then σ is a valid signature on (τ, m, i). Other queries to H are answered similarly, but with a random message m. The Gaussian parameter ν is large enough so that $H(\tau \| i)$ is statistically close to uniform in $\mathbb{F}_q^\ell$.

Eventually $\mathcal{A}$ outputs a tag τ^*, a message m^*, a function f encoded as $\langle f \rangle = (c_1, \ldots, c_k) \in \mathbb{Z}^k$, and a signature σ^*. Let σ_i be the short vector chosen when programming $H(\tau^* \| i)$, and let $\sigma_f := \sum_i c_i \sigma_i$. We claim that if $\mathcal{A}$ outputs a valid forgery, then with high probability the vector $\sigma^* - \sigma_f$ is a nonzero vector in Λ_2 of length at most β; i.e., a solution to the $\mathsf{SIS}_{q,n,\beta}$ problem.

Suppose $\mathcal{A}$ outputs a type 2 forgery, so the simulator has generated signatures $\vec{\sigma} = (\sigma_1, \ldots, \sigma_k)$ on messages $\vec{m} = (m_1, \ldots, m_k)$ using the tag τ^*. First observe that the verification condition (1a) implies that $\|\sigma^*\|$ and $\|\sigma_f\|$ are both less than $k \cdot \frac{p}{2} \cdot \nu \sqrt{n}$, and therefore $\|\sigma^* - \sigma_f\| \leq \beta$. Next observe that if the forgery is valid, then $m^* \neq f(\vec{m})$. The verification condition (1b) implies that $(\sigma^* - \sigma_f) \bmod p = m^* - f(\vec{m}) \neq 0$, and thus $\sigma^* - \sigma_f \neq 0$. On the other hand, verification condition (1c) implies that $\mathbf{A} \cdot \sigma^* \bmod q = \mathbf{A} \cdot \sigma_f \bmod q$, and thus $\sigma^* - \sigma_f \in \Lambda_2$. The argument for a type 1 forgery is similar, using random messages $\vec{m}$ instead of queried ones. □

Worst-case connections. By [19, Proposition 5.7], if $q \geq \beta \cdot \omega(\sqrt{n \log n})$, then the $\mathsf{SIS}_{q,m,\beta}$ problem is as hard as approximating the SIVP problem in the worst case to within $\beta \cdot \tilde{O}(\sqrt{n})$ factors. Our requirement in the Setup algorithm that $q \geq (nkp)^2$ guarantees that q is sufficiently large for this theorem to apply. While the exact worst-case approximation factor will depend on the parameters k and p, it is polynomial in n in any case.

Comparison with prior work. Boneh and Freeman [6] describe a linearly homomorphic signature scheme that can authenticate vectors over $\mathbb{F}_p$ for small p, with unforgeability also depending on the SIS problem. However, for their system to securely sign k messages, the $\mathsf{SIS}_{q,2n-k,\beta}$ problem must be difficult for $\beta = \tilde{O}(k^{3/2} \cdot k! \cdot (n/\lg q)^{k/2+1})$, and therefore their system is designed to only sign a constant number of vectors per

data set ($k = O(1)$) while maintaining a polynomial connection to worst-case lattice problems. On the other hand, for the same value of q our system remains secure when signing $k = \text{poly}(n)$ vectors per data set.

Privacy. We now show that our linearly homomorphic signature scheme is weakly context hiding. Specifically, we show that a derived signature on a linear combination $m' = \sum_{i=1}^{k} c_i m_i$ depends (up to negligible statistical distance) only on m' and the c_i, and not on the initial messages m_i. Consequently, even an unbounded adversary cannot win the privacy game of Definition 2.3. The proof of the following theorem can be found in the full version of this paper [5].

Theorem 4.2. *Suppose that ν defined in the* Setup *algorithm satisfies $\nu > p^{s+1} \cdot k^s \cdot \omega(\sqrt{\log n})$. Then the linearly homomorphic signature scheme described above is s-weakly context hiding for data sets of size k.*

5 Background on Ideal Lattices

A *number field* is a finite-degree algebraic extension of the rational numbers $\mathbb{Q}$. Any number field K can be represented as $\mathbb{Q}[x]/(f(x))$ for some monic, irreducible polynomial $f(x)$ with integer coefficients (and for each K there are infinitely many such f). The *degree* of a number field K is its dimension as a vector space over $\mathbb{Q}$, and is also the degree of any polynomial f defining K. For any given f, the set $\{1, x, x^2, \ldots, x^{\deg f - 1}\}$ is a $\mathbb{Q}$-basis for K, and we can therefore identify K with $\mathbb{Q}^n$ by mapping a polynomial of degree less than n to its vector of coefficients. By identifying K with $\mathbb{Q}^n$ using this "coefficient embedding," we can define a length function $\|\cdot\|$ on elements of K simply by using any norm on $\mathbb{Q}^n$. This length function is non-canonical — it depends explicitly on the choice of f used to represent K. (Here all norms will be the ℓ_2 norm unless otherwise stated.)

Our identification of $K = \mathbb{Q}[x]/(f(x))$ with $\mathbb{Q}^n$ induces a *multiplicative* structure on $\mathbb{Q}^n$ in addition to the usual *additive* structure. We define a parameter $\gamma_f := \sup_{u,v \in K} \frac{\|u \cdot v\|}{\|u\| \cdot \|v\|}$. This parameter bounds how much multiplication can increase the length of the product, relative to the product of the length of the factors. For our applications we will need to have $\gamma_f = \text{poly}(n)$. If n is a power of 2, then the function $f(x) = x^n + 1$ has $\gamma_f \leq \sqrt{n}$ (cf. [17, Lemma 7.4.3]). We will think of this $f(x)$ as our "preferred" choice for applications.

Number rings and ideals. A *number ring* is a ring whose field of fractions is a number field K. A survey of arithmetic in number rings can be found in [36]; here we summarize the key points.

Every number field has a subring, called the *ring of integers* and denoted by $\mathcal{O}_K$, that plays the same role with respect to K as the integers $\mathbb{Z}$ do with respect to $\mathbb{Q}$. The ring of integers consists of all elements of K whose characteristic polynomials have integer coefficients. Under the identification of K with $\mathbb{Q}^n$, the ring $\mathcal{O}_K$ forms a full-rank discrete subgroup of $\mathbb{Q}^n$; i.e., a lattice. Inside $\mathcal{O}_K$ is the subring $R = \mathbb{Z}[x]/(f(x))$. Under our identification of K with $\mathbb{Q}^n$, the ring R corresponds to $\mathbb{Z}^n$. In general R is a proper sublattice of $\mathcal{O}_K$.

An *(integral) ideal* of R is an additive subgroup $I \subset R$ that is closed under multiplication by elements of R. By our identification of R with $\mathbb{Z}^n$, the ideal I is a sublattice of R and is therefore also called an *ideal lattice*. Note that this usage of "ideal lattice" to refer to a rank one R-module differs from that of [34,24], which use the terminology to refer to R-modules of arbitrary rank.

An ideal $I \subset R$ is *prime* if for $x, y \in R$, $xy \in I$ implies either $x \in I$ or $y \in I$. If $\mathfrak{p}$ is a prime ideal, then $R/\mathfrak{p}$ is a finite field $\mathbb{F}_{p^e}$; the integer e is the *degree* of $\mathfrak{p}$ and the prime p is the *characteristic* of $\mathfrak{p}$. An ideal I is *principal* if it can be written as $\alpha \cdot R$ for some $\alpha \in R$. In general most ideals are not principal; the proportion of principal ideals is 1 over the size of the *class group* of R, which is exponential in n. The *norm* of an ideal I is the size of the (additive) group R/I.

If $\mathfrak{p}$ is a prime ideal of R, then by a theorem of Kummer and Dedekind [36, Theorem 8.2] we can write $\mathfrak{p} = p \cdot R + h(x) \cdot R$ for some polynomial $h(x)$ whose reduction mod p is an irreducible factor of $f(x) \bmod p$. Writing $\mathfrak{p}$ in this "two-element representation" makes it easy to compute the corresponding quotient map $\mathbb{Z}[x]/(f(x)) \to \mathbb{F}_{p^e}$; we simply reduce a polynomial in $\mathbb{Z}[x]$ modulo both p and $h(x)$. In particular, if $\mathfrak{p}$ is a degree-one prime, then $h(x) = x - \alpha$ for some integer α and the quotient map is given by $z(x) \mapsto z(\alpha) \bmod p$.

Generating ideals with a short basis. If we are to use ideals as the lattices Λ_1 and Λ_2 in our abstract signature scheme, we will need a method for generating ideals $\mathfrak{p}$ and $\mathfrak{q}$ in R along with a short basis for $\mathfrak{p} \cap \mathfrak{q}$ (which is equal to $\mathfrak{p} \cdot \mathfrak{q}$ if $\mathfrak{p}$ and $\mathfrak{q}$ are relatively prime ideals). Furthermore, our security proof requires that given $\mathfrak{q}$ *without* a short basis, we can still compute a prime $\mathfrak{p}$ with a short basis.

In our construction we generate ideals using an algorithm of Smart and Vercauteren [33]. This algorithm generates a *principal* prime ideal $\mathfrak{p}$ along with a short generator g of $\mathfrak{p}$. We can multiply g by powers of x to generate a full-rank set of vectors $\{g, xg, x^2g, \ldots, x^{n-1}g\}$ that spans $\mathfrak{p}$. Since $\|x\| = 1$, we have $\|x^i g\| \leq \gamma_f \cdot \|g\|$, so if γ_f is small then these vectors are all short.

Theorem 5.1 ([33, §3.1]). *There is an algorithm* PrincGen *that takes input a monic irreducible polynomial $f(x) \in \mathbb{Z}[x]$ of degree n and a parameter δ, and outputs a principal degree-one prime ideal $\mathfrak{p} = (p, x - a)$ in $K := \mathbb{Q}[x]/(f(x))$, along with a generator g of $\mathfrak{p}$ satisfying $\|g\| \leq \delta\sqrt{n}$.*

The algorithm works by sampling a random g with low norm and seeing if it generates a prime ideal in $\mathcal{O}_K$. Smart and Vercauteren do not give a rigorous analysis of the algorithm's running time, but heuristically we expect that by the number field analogue of the Prime Number Theorem [30, Theorem 8.9], we will find a prime ideal after trying $O(n \log n \log \delta)$ values of g.

6 Homomorphic Signatures for Polynomial Functions

In this section we describe our main construction, a signature scheme that authenticates polynomial functions on signed messages.

Recall the basic idea of our linearly homomorphic scheme from Section 4: messages are elements of $\mathbb{Z}^n \bmod \Lambda_1$, functions are mapped (via the hash function ω_τ) to elements of $\mathbb{Z}^n \bmod \Lambda_2$, and a signature on $(\tau, m, \langle f \rangle)$ is a short vector in the coset of $\Lambda_1 \cap \Lambda_2$ defined by m and $\omega_\tau(\langle f \rangle)$. To verify a signature σ, we simply confirm that σ is a short vector and that $\sigma \bmod \Lambda_1 = m$ and $\sigma \bmod \Lambda_2 = \omega_\tau(\langle f \rangle)$. The homomorphic property follows from the fact that the maps $\mathbf{x} \mapsto (\mathbf{x} \bmod \Lambda_i)$ are linear maps — i.e., *vector space homomorphisms* — and therefore adding signatures corresponds to adding the corresponding messages and (encoded) functions.

Our polynomial system is based on the following idea: what if the lattice $\mathbb{Z}^n$ has a *ring* structure and the lattices Λ_1, Λ_2 are *ideals*? Then the maps $\mathbf{x} \mapsto (\mathbf{x} \bmod \Lambda_i)$ are *ring homomorphisms*, and therefore adding *or multiplying* signatures corresponds to adding *or multiplying* the corresponding messages and functions. Since any polynomial can be computed by repeated additions and multiplications, adding this structure to our lattices allows us to authenticate polynomial functions on messages.

Concretely, we let $F(x) \in \mathbb{Z}[x]$ be a monic, irreducible polynomial of degree n. We define the number field $K = \mathbb{Q}[x]/(F(x))$ and let $\mathcal{O}_K$ be the lattice in $\mathbb{Q}^n$ corresponding (via the coefficient embedding) to the ring of integers of K. We now let Λ_1 and Λ_2 be (degree one) prime ideals $\mathfrak{p}, \mathfrak{q} \subset \mathcal{O}_K$ of norm p, q respectively. We fix an isomorphism from $\mathcal{O}_K/\mathfrak{p}$ to $\mathbb{F}_p$ by representing $\mathfrak{p}$ as $p\mathcal{O}_K + (x-a)\mathcal{O}_K$ and mapping $h(x) \in \mathcal{O}_K$ to $h(a) \bmod p \in \mathbb{F}_p$, and similarly for $\mathcal{O}_K/\mathfrak{q} \cong \mathbb{F}_q$. We can now sign messages exactly as in the linearly homomorphic scheme.

In our linearly homomorphic scheme we used the projection functions π_i as a generating set for admissible functions, and we encoded the function $f = \sum c_i \pi_i$ by its coefficient vector $(c_1, \ldots, c_k)$ (with the c_i interpreted as integers in $(-p/2, p/2]$). When we consider polynomial functions on $\mathbb{F}_p[x_1, \ldots, x_k]$, the projection functions π_i are exactly the linear monomials x_i, and we can obtain any (non-constant) polynomial function by adding and multiplying monomials. If we fix an ordering on all monomials of the form $x_1^{e_1} \cdots x_k^{e_k}$, then we can encode any polynomial function as its vector of coefficients, with the unit vectors $\mathbf{e}_i$ representing the linear monomials x_i for $i = 1, \ldots, k$.

The hash function ω_τ is defined exactly as in our linear scheme: for a function f in $\mathbb{F}_p[x_1, \ldots, x_k]$ whose encoding is $\langle f \rangle = (c_1, \ldots, c_\ell) \in \mathbb{Z}^\ell$, we define a polynomial $\hat{f} \in \mathbb{Z}[x_1, \ldots, x_k]$ that reduces to $f \bmod p$. We then define $\omega_\tau(\langle f \rangle) = \hat{f}(\alpha_1, \ldots, \alpha_k)$, where $\alpha_i \in \mathbb{F}_q$ are defined to be $H(\tau, i)$ for some hash function H.

We use the same lifting of f to $\hat{f} \in \mathbb{Z}[x_1, \ldots, x_k]$ to evaluate polynomials on signatures; specifically, given a polynomial f and signatures $\sigma_1, \ldots, \sigma_k \in K$ on messages $m_1, \ldots, m_k \in \mathbb{F}_p$, the signature on $f(m_1, \ldots, m_k)$ is given by $\hat{f}(\sigma_1, \ldots, \sigma_k)$.

Recall that for $\mathbf{v}_1, \mathbf{v}_2 \in \mathcal{O}_K$, the length of $\mathbf{v}_1 \cdot \mathbf{v}_2$ is bounded by $\gamma_F \cdot \|\mathbf{v}_1\| \cdot \|\mathbf{v}_2\|$. Thus if we choose $F(x)$ so that γ_F is polynomial in n, then multiplying together a constant number of vectors of length $\mathrm{poly}(n)$ produces a vector of length $\mathrm{poly}(n)$. It follows that the derived signature $f(\sigma_1, \ldots, \sigma_k)$ is short as long as the degree of f is bounded and the coefficients of f are small (when lifted to the integers). The system therefore can support polynomial computations on messages for polynomials with small coefficients and bounded degree.

The polynomially homomorphic scheme. We now describe the scheme formally.

$\mathsf{Setup}(1^n, k)$. On input a security parameter n and a maximum data set size k, do the following:

1. Choose a monic irreducible polynomial $F(x) \in \mathbb{Z}[x]$ of degree n with $\gamma_F = \mathrm{poly}(n)$.
   Let $K := \mathbb{Q}[x]/(F(x))$ be embedded in $\mathbb{Q}^n$ via the coefficient embedding.
   Let $R = \mathbb{Z}^n$ be the lattice corresponding $\mathbb{Z}[x]/(F(x)) \subset \mathcal{O}_K$.
2. Run the PrincGen algorithm twice on inputs F, n to produce distinct principal degree-one prime ideals $\mathfrak{p} = (p, x-a)$ and $\mathfrak{q} = (q, x-b)$ of R with generators $g_\mathfrak{p}, g_\mathfrak{q}$, respectively.
3. Let $\mathbf{T}$ be the basis $\{g_\mathfrak{p} g_\mathfrak{q}, g_\mathfrak{p} g_\mathfrak{q} x, \ldots, g_\mathfrak{p} g_\mathfrak{q} x^{n-1}\}$ of $\mathfrak{p} \cdot \mathfrak{q}$.
4. Define $\nu := \gamma_F^2 \cdot n^3 \log n$. Choose integers $y = \mathrm{poly}(n)$ and $d = O(1)$.
5. Let $H\colon \{0,1\}^* \to \mathbb{F}_q$ be a hash function (modeled as a random oracle).
6. Output the public key $\mathsf{pk} = (F, p, q, a, b, \nu, y, d, H)$ and secret key $\mathsf{sk} = \mathbf{T}$.

The public key pk defines the following system parameters:

- The message space is $\mathbb{F}_p$ and signatures are short vectors in R.
- The set of admissible functions $\mathcal{F}$ is all polynomials in $\mathbb{F}_p[x_1, \ldots, x_k]$ with coefficients in $\{-y, \ldots, y\}$, degree at most d, and constant term zero. The quantity y is only used in algorithm Verify.
- Let $\ell = \binom{k+d}{d} - 1$. Let $\{Y_j\}_{j=1}^{\ell}$ be the set of all non-constant monomials $x_1^{e_1} \cdots x_k^{e_k}$ of degree $\sum e_i \le d$, ordered lexicographically. Then any polynomial function $f \in \mathcal{F}$ is defined by $f(\vec{m}) = \sum_{j=1}^{\ell} c_j Y_j(\vec{m})$ for $c_j \in \mathbb{F}_p$. We interpret the c_j as integers in $[-y, y]$ and encode f as $\langle f \rangle = (c_1, \ldots, c_\ell) \in \mathbb{Z}^\ell$.
- To evaluate the hash function ω_τ on an encoded function $\langle f \rangle = (c_1, \ldots, c_\ell) \in \mathbb{Z}^\ell$, do the following:
  (a) For $i = 1, \ldots, k$, compute $\alpha_i \leftarrow H(\tau \| i)$.
  (b) Define $\omega_\tau(\langle f \rangle) := \sum_{j=1}^{\ell} c_j Y_j(\alpha_1, \ldots, \alpha_k) \in \mathbb{F}_q$.

$\mathsf{Sign}(\mathsf{sk}, \tau, m, i)$. On input a secret key sk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p$, and an index i, do:

1. Compute $\alpha_i := H(\tau \| i) \in \mathbb{F}_q$.
2. Compute $h = h(x) \in R$ such that $h(a) \bmod p = m$ and $h(b) \bmod q = \alpha_i$.
3. Output $\sigma \leftarrow \mathsf{SamplePre}(\mathfrak{p} \cdot \mathfrak{q}, \mathbf{T}, h, \nu) \quad \in (\mathfrak{p} \cdot \mathfrak{q}) + h$.

$\mathsf{Verify}(\mathsf{pk}, \tau, m, \sigma, f)$. On input a public key pk, a tag $\tau \in \{0,1\}^n$, a message $m \in \mathbb{F}_p$, a signature $\sigma = \sigma(x) \in R$, and a function $f \in \mathcal{F}$, do:

1. If all of the following conditions hold, output 1 (accept); otherwise output 0 (reject):
   (a) $\|\sigma\| \le \ell \cdot y \cdot \gamma_F^{d-1} \cdot (\nu\sqrt{n})^d$.
   (b) $\sigma(a) \bmod p = m$.
   (c) $\sigma(b) \bmod q = \omega_\tau(\langle f \rangle)$.

$\mathsf{Evaluate}(\mathsf{pk}, \tau, f, \vec{\sigma})$. On input a public key pk, a tag $\tau \in \{0,1\}^n$, a function $f \in \mathcal{F}$ encoded as $\langle f \rangle = (c_1, \ldots, c_\ell) \in \mathbb{Z}^\ell$, and a tuple of signatures $\sigma_1, \ldots, \sigma_k \in \mathbb{Z}^n$, do:

1. Lift $f \in \mathbb{F}_p[x_1, \ldots, x_k]$ to $\mathbb{Z}[x_1, \ldots, x_k]$ by setting $\hat{f} := \sum_{j=1}^{\ell} c_j Y_j(x_1, \ldots, x_k)$.
2. Output $\hat{f}(\sigma_1, \ldots, \sigma_k)$.

In the full version of this paper [5], we show that this polynomially homomorphic signature scheme is correct with overwhelming probability and that it is length efficient; i.e., the bit length of a derived signature depends logarithmically on the data set size k.

Unforgeability. As in our linearly homomorphic scheme from Section 4, an adversary that can forge a signature in the above system can be used to find a short vector in the lattice used to authenticate functions, which in this case is the ideal $\mathfrak{q}$.

Theorem 6.1. *For fixed n, let F_n be the polynomial chosen in Step (1) of the* Setup *algorithm above, and let $\mathcal{L}_n$ be the distribution of ideals $\mathfrak{q}$ output by the Smart-Vercauteren algorithm when given input polynomial $F_n(x)$ and parameter $\delta = n$. Let $\mathcal{L}_F$ be the ensemble $\{\mathcal{L}_n\}$. If $\mathcal{L}_F$-$\mathsf{SIS}_{n,\beta}$ is infeasible for*

$$\beta = 2 \cdot \binom{k+d}{d} \cdot y \cdot \gamma_{F_n}^{3d-1} \left(n^3 \log n\right)^d,$$

then the polynomially homomorphic signature scheme defined above is unforgeable in the random oracle model.

The proof of this theorem uses the same ideas as that of Theorem 4.1; details are in the full version of this paper [5]. While Theorem 6.1 gives a concrete security result for our system, the distribution $\mathcal{L}_F$ of prime ideals output by the Smart-Vercauteren algorithm is not well understood. It is an open problem to modify the system to use ideals sampled from a distribution that admits a random self-reduction.

Privacy. The homomorphic signature scheme described above is not weakly context hiding in the sense of Definition 2.3. To see why, consider an attacker that outputs two data sets $\vec{m}_0^* := (0,0)$ and $\vec{m}_1^* := (0,1)$, each containing two messages in $(R/\mathfrak{p}) \cong \mathbb{F}_p$. The attacker also outputs the function $f(x,y) := x \cdot y$, which is a valid function to request since $f(\vec{m}_0^*) = f(\vec{m}_1^*)$.

The challenger chooses a random bit b in $\{0,1\}$ and generates signatures σ_1, σ_2 in R for the two messages in $\vec{m}_b^*$. It gives the attacker $\sigma := \sigma_1 \cdot \sigma_2$.

Now, when $b = 0$ both σ_1 and σ_2 are in $\mathfrak{p}$ and therefore the derived siganture $\sigma = \sigma_1\sigma_2$ is in $\mathfrak{p}^2$. However, when $b = 1$ we know that $\sigma_2 \notin \mathfrak{p}$ and therefore $\sigma \in \mathfrak{p}^2$ only if $\sigma_1 \in \mathfrak{p}^2$. But σ_1 is in $\mathfrak{p}^2$ with probability at most $1/2$. In other words, $\Pr[\sigma \in \mathfrak{p}^2] = 1$ when $b = 0$, but $\Pr[\sigma \in \mathfrak{p}^2] \leq 1/2$ when $b = 1$. Therefore, an adversary that outputs $b' = 0$ if $\sigma \in \mathfrak{p}^2$ and $b' = 1$ otherwise has advantage at least $1/2$ in distinguishing $\vec{m}_0^*$ from $\vec{m}_1^*$ just given σ. Consequently the scheme is not weakly context hiding.

Using small fields. The signature scheme described above signs messages defined over a finite field $\mathbb{F}_p$, where p is exponential in n. In the full version [5], we show how to authenticate polynomial functions of data defined over a field where p is constant or polynomial in n, as we can for linear computations using the scheme of Section 4.

7 Conclusions and Open Problems

We have presented a homomorphic signature scheme that authenticates polynomial functions of bounded degree on signed data.

There are many open problems that remain in this area. First, as we explained in the introduction, we may desire that derived signatures not leak information about the original data set. This privacy property can be achieved for linear functions (e.g. as in [6] and in this paper), but is an open problem for quadratic and higher degree polynomials.

Second, the security of our scheme could be strengthened by removing the random oracle from our construction. All current linearly homomorphic signature schemes use the random oracle to simulate signatures during a chosen message attack. New ideas are needed to eliminate the random oracle while preserving the homomorphic properties.

Third, it is an open problem to base the security of our system on *worst case* problems on ideal lattices. In particular, we wish to generate ideals for our polynomially homomorphic signature scheme from a distribution that admits a random self-reduction. While Gentry [18] has achieved this result for homomorphic encryption, his key generation algorithm is not suitable for our scheme: it produces an ideal $\mathfrak{q}$ and a short vector in $\mathfrak{q}^{-1}$, whereas we require a short vector in $\mathfrak{q}$. One direction for future work is to construct a homomorphic signature scheme that uses Gentry's key generation algorithm; another is to construct an algorithm that samples a uniformly random ideal $\mathfrak{q}$ along with a short vector in $\mathfrak{q}$.

Finally, our construction can be seen as a first step on the road to a *fully homomorphic signature scheme*, which could authenticate the computation of *any* function on signed data. A fully homomorphic signature scheme would be a useful parallel to existing fully homomorphic encryption systems. Current constructions of fully homomorphic encryption are obtained by applying a "bootstrapping" process to a scheme that allows a limited amount of computation on encrypted data. It is unclear whether Gentry's bootstrapping process [17] can be applied to signature schemes such as ours. We leave this as a beautiful open problem. Even if a fully homomorphic scheme cannot be immediately realized, it would be useful to enlarge the set of admissible functions $\mathcal{F}$.

Acknowledgments. The authors thank Rosario Gennaro, Craig Gentry, Hugo Krawczyk, Gil Segev, and Brent Waters for helpful discussions about this work.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data (2010) (manuscript)
2. Ajtai, M.: Generating hard instances of the short basis problem. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 1–9. Springer, Heidelberg (1999)
3. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS, pp. 75–86 (2009), full version http://www.cc.gatech.edu/~cpeikert/pubs/shorter.pdf
4. Ateniese, G., Chou, D.H., de Medeiros, B., Tsudik, G.: Sanitizable signatures. In: di Vimercati, S.D.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 159–177. Springer, Heidelberg (2005)
5. Boneh, D., Freeman, D.: Homomorphic signatures for polynomial functions. Cryptology ePrint Archive, report 2011/018 (2011), http://eprint.iacr.org/2011/018
6. Boneh, D., Freeman, D.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures. In: Gennaro, R. (ed.) PKC 2011. LNCS, vol. 6571, pp. 1–16. Springer, Heidelberg (2011), full version http://eprint.iacr.org/2010/453

7. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
8. Brzuska, C., Busch, H., Dagdelen, Ö., Fischlin, M., Franz, M., Katzenbeisser, S., Manulis, M., Onete, C., Peter, A., Poettering, B., Schröder, D.: Redactable signatures for tree-structured data: Definitions and constructions. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 87–104. Springer, Heidelberg (2010)
9. Brzuska, C., Fischlin, M., Freudenreich, T., Lehmann, A., Page, M., Schelbert, J., Schröder, D., Volk, F.: Security of sanitizable signatures revisited. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 317–336. Springer, Heidelberg (2009)
10. Chang, E.C., Lim, C.L., Xu, J.: Short redactable signatures using random trees. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 133–147. Springer, Heidelberg (2009)
11. Charles, D., Jain, K., Lauter, K.: Signatures for network coding. International Journal of Information and Coding Theory 1(1), 3–14 (2009)
12. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
13. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proc. of the 22nd ACM Symposium on Theory of Computing, pp. 416–426 (1990)
14. Fragouli, C., Soljanin, E.: Network coding fundamentals. Found. Trends Netw. 2(1), 1–133 (2007)
15. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
16. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
17. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009), http://crypto.stanford.edu/craig
18. Gentry, C.: Toward basing fully homomorphic encryption on worst-case hardness. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 116–137. Springer, Heidelberg (2010)
19. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: 40th ACM Symposium on Theory of Computing—STOC 2008, pp. 197–206. ACM, New York (2008)
20. Goldwasser, S., Kalai, Y., Rothblum, G.: Delegating computation: Interactive proofs for muggles. In: 40th ACM Symposium on Theory of Computing — STOC 2008, pp. 113–122 (2008)
21. Haber, S., Hatano, Y., Honda, Y., Horne, W., Miyazaki, K., Sander, T., Tezoku, S., Yao, D.: Efficient signature schemes supporting redaction, pseudonymization, and data deidentification. In: ASIACCS 2008, pp. 353–362. ACM, New York (2008)
22. Johnson, R., Molnar, D., Song, D., Wagner, D.: Homomorphic signature schemes. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 244–262. Springer, Heidelberg (2002)
23. Krohn, M., Freedman, M., Mazieres, D.: On-the-fly verification of rateless erasure codes for efficient content distribution. In: Proc. of IEEE Symposium on Security and Privacy, pp. 226–240 (2004)
24. Lyubashevsky, V., Micciancio, D.: Generalized compact knapsacks are collision resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 144–155. Springer, Heidelberg (2006)
25. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 1–23. Springer, Heidelberg (2010)

26. Micali, S.: Computationally sound proofs. SIAM J. of Computing 30(4), 1253–1298 (2000); extended abstract in FOCS 1994
27. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2004, pp. 372–381. IEEE Computer Society, Washington, DC, USA (2004)
28. Miyazaki, K., Hanaoka, G., Imai, H.: Digitally signed document sanitizing scheme based on bilinear maps. In: ACM Symposium on Information, Computer and Communications Security — ASIACCS 2006, pp. 343–354 (2006)
29. Miyazaki, K., Iwamura, M., Matsumoto, T., Sasaki, R., Yoshiura, H., Tezuka, S., Imai, H.: Digitally signed document sanitizing scheme with disclosure condition control. IEICE Transactions on Fundamentals E88-A(1), 239–246 (2005)
30. Montgomery, H.L., Vaughan, R.C.: Multiplicative number theory. I. Classical theory. Cambridge Studies in Advanced Mathematics, vol. 97. Cambridge University Press, Cambridge (2007)
31. Naccache, D.: Is theoretical cryptography any good in practice? CHES 2010 invited talk (2010), http://www.iacr.org/workshops/ches/ches2010
32. Quinlan, J.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
33. Smart, N.P., Vercauteren, F.: Fully homomorphic encryption with relatively small key and ciphertext sizes. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 420–443. Springer, Heidelberg (2010)
34. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 617–635. Springer, Heidelberg (2009)
35. Steinfeld, R., Bull, L., Zheng, Y.: Content extraction signatures. In: Kim, K. (ed.) ICISC 2001. LNCS, vol. 2288, pp. 285–304. Springer, Heidelberg (2002)
36. Stevenhagen, P.: The arithmetic of number rings. In: Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography. Math. Sci. Res. Inst. Publ., vol. 44, pp. 209–266. Cambridge Univ. Press, Cambridge (2008)
37. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 1–18. Springer, Heidelberg (2008)
38. Zhao, F., Kalker, T., Médard, M., Han, K.: Signatures for content distribution with network coding. In: Proc. Intl. Symp. Info. Theory (ISIT) (2007)

Semi-homomorphic Encryption and Multiparty Computation

Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias

Department of Computer Science, Aarhus University and CFEM*

Abstract. An additively-homomorphic encryption scheme enables us to compute linear functions of an encrypted input by manipulating only the ciphertexts. We define the relaxed notion of a *semi-homomorphic* encryption scheme, where the plaintext can be recovered as long as the computed function does not increase the size of the input "too much". We show that a number of existing cryptosystems are captured by our relaxed notion. In particular, we give examples of semi-homomorphic encryption schemes based on lattices, subset sum and factoring. We then demonstrate how semi-homomorphic encryption schemes allow us to construct an *efficient* multiparty computation protocol for arithmetic circuits, UC-secure against a dishonest majority. The protocol consists of a preprocessing phase and an online phase. Neither the inputs nor the function to be computed have to be known during preprocessing. Moreover, the online phase is extremely efficient as it requires *no cryptographic operations*: the parties only need to exchange additive shares and verify information theoretic MACs. Our contribution is therefore twofold: from a theoretical point of view, we can base multiparty computation on a variety of different assumptions, while on the practical side we offer a protocol with better efficiency than any previous solution.

1 Introduction

The fascinating idea of computing on encrypted data can be traced back at least to a seminal paper by Rivest, Adleman and Dertouzos [RAD78] under the name of *privacy homomorphism*. A privacy homomorphism, or *homomorphic encryption scheme* in more modern terminology, is a public-key encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ for which it holds that $\mathsf{D}(\mathsf{E}(a) \otimes \mathsf{E}(b)) = a \oplus b$, where $(\otimes, \oplus)$ are some group operation in the ciphertext and plaintext space respectively. For instance, if $\oplus$ represents modular addition in some ring, we call such a scheme *additively-homomorphic*. Intuitively a homomorphic encryption scheme enables two parties, say Alice and Bob, to perform *secure computation*: as an example, Alice could encrypt her input a under her public key, send the ciphertext $\mathsf{E}(a)$ to Bob; now by the homomorphic property, Bob can compute a ciphertext containing, e.g., $\mathsf{E}(a \cdot b + c)$ and send it back to Alice, who can decrypt and learn the

* Center for Research in the Foundations of Electronic Markets, supported by the Danish Strategic Research Council.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 169–188, 2011.

result. Thus, Bob has computed a non trivial function of the input a. However, Bob only sees an encryption of a which leaks no information on a itself, assuming that the encryption scheme is secure. Informally we will say that a set of parties $P_1, \ldots, P_n$ holding private inputs $x_1, \ldots, x_n$ *securely compute* a function of their inputs $y = f(x_1, \ldots, x_n)$ if, by running some cryptographic protocol, the honest parties learn the correct output of the function y. In addition, even if (up to) $n-1$ parties are corrupt and cooperate, they are not able to learn any information about the honest parties' inputs, no matter how they deviate from the specifications of the protocol.

Building *secure multiparty computation (MPC) protocols* for this case of *dishonest majority* is essential for several reasons: First, it is notoriously hard to handle dishonest majority efficiently and it is well known that unconditionally secure solutions do not exist. Therefore, we cannot avoid using some form of public-key technology which is typically much more expensive than the standard primitives used for honest majority (such as secret sharing). Secondly, security against dishonest majority is often the most natural to shoot for in applications, and is of course the only meaningful goal in the significant 2-party case. Thus, finding practical solutions for dishonest majority under reasonable assumptions is arguably the most important research goal with respect to applications of multiparty computation.

While *fully-homomorphic* encryption [Gen09] allows for significant improvement in communication complexity, it would incur a huge computational overhead with current state of the art. In this paper we take a different road: in a nutshell, we relax the requirements of homomorphic encryption so that we can implement it under a variety of assumptions, and we show how this weaker primitive is sufficient for efficient MPC. Our main contributions are:

A framework for semi-homomorphic encryption: we define the notion of a *semi-homomorphic encryption modulo p*, for a modulus p that is input to the key generation. Abstracting from the details, the encryption function is additively homomorphic and will accept any integer x as input plaintext. However, in contrast to what we usually require from a homomorphic cryptosystem, decryption returns the correct result modulo p only if x is numerically small enough. We demonstrate the generality of the framework by giving several examples of known cryptosystems that are semi-homomorphic or can be modified to be so by trivial adjustments. These include: the Okamoto-Uchiyama cryptosystem [OU98]; Paillier cryptosystem [Pai99] and its generalization by Damgård and Jurik [DJ01]; Regev's LWE based cryptosystem [Reg05]; the scheme of Damgård, Geisler and Krøigaard [DGK09] based on a subgroup-decision problem; the subset-sum based scheme by Lyubashevsky, Palacio and Segev [LPS10]; Gentry, Halevi and Vaikuntanathan's scheme [GHV10] based on LWE, and van Dijk, Gentry, Halevi and Vaikuntanathan's scheme [DGHV10] based on the approximate gcd problem. We also show a zero-knowledge protocol for any semi-homomorphic cryptosystem, where a prover, given ciphertext C and public key pk, demonstrates that he knows plaintext x and randomness r such that $C = \mathsf{E}_{pk}(x, r)$, and that x furthermore is numerically less than a given bound. We show that using a twist

of the amortization technique of Cramer and Damgård [CD09], one can give u such proofs in parallel where the soundness error is 2^{-u} and the cost per instance proved is essentially 2 encryption operations for both parties. The application of the technique from [CD09] to prove that a plaintext is bounded in size is new and of independent interest.

Information-theoretic "online" MPC: we propose a UC secure [Can01] protocol for arithmetic multiparty computation that, in the presence of a trusted dealer who does not know the inputs, offers information-theoretic security against an adaptive, malicious adversary that corrupts any dishonest majority of the parties. The main idea of the protocol is that the parties will be given additive sharing of multiplicative triples [Bea91], together with information theoretic MACs of their shares – forcing the parties to use the correct shares during the protocol. This online phase is essentially optimal, as no symmetric or public-key cryptography is used, matching the efficiency of passive protocols for honest majority like [BOGW88, CCD88]. Concretely, each party performs $O(n^2)$ multiplications modulo p to evaluate a secure multiplication. This improves on the previous protocol of Damgård and Orlandi (DO) [DO10] where a Pedersen commitment was published for every shared value. Getting rid of the commitments we improve on efficiency (a factor of $\Omega(\kappa)$, where κ is the security parameter) and security (information theoretic against computational). Implementation results for the two-party case indicate about 6 msec per multiplication (see the full version [BDOZ10]), at least an order of magnitude faster than that of DO on the same platform. Moreover, in DO the modulus p of the computation had to match the prime order of the group where the commitments live. Here, we can, however, choose p freely to match the application which typically allows much smaller values of p.

An efficient implementation of the offline phase: we show how to replace the share dealer for the online phase by a protocol based solely on semi-homomorphic encryption[1]. Our offline phase is UC-secure against any dishonest majority, and it matches the lower bound for secure computation with dishonest majority of $O(n^2)$ public-key operations per multiplication gate [HIK07]. In the most efficient instantiation, the offline phase of DO requires security of Paillier encryption and hardness of discrete logarithms. Our offline phase only has to assume security of Paillier cryptosystem and achieves similar efficiency: A count of operations suggests that our offline phase is as efficient as DO up to a small constant factor (about 2-3). Preliminary implementation results indicate about 2-3 sec to prepare a multiplication. Since we generalize to any semi-homomorphic scheme including Regev's scheme, we get the first potentially practical solution for dishonest majority that is believed to withstand a quantum attack. It is not possible to achieve UC security for dishonest majority without set-up assumptions, and our protocol works in the registered public-key model of [BCNP04] where we

[1] The trusted dealer could be implemented using any existing MPC protocol for dishonest majority, but we want to show how we can do it *efficiently* using semi-homomorphic encryption.

assume that public keys for all parties are known, and corrupted parties know their own secret keys.

Related Work: It was shown by Canetti, Lindell, Ostrovsky and Sahai [CLOS02] that secure computation is possible under general assumptions even when considering any corrupted number of parties in a concurrent setting (the UC framework). Their solution is, however, very far from being practical. For computation over Boolean circuits efficient solutions can be constructed from Yao's garbled circuit technique, see e.g. Pinkas, Schneider, Smart and Williams [PSSW09]. However, our main interest here is arithmetic computation over larger fields or rings, which is a much more efficient approach for applications such as benchmarking or some auction variants. A more efficient solution for the arithmetic case was shown by Cramer, Damgård and Nielsen [CDN01], based on threshold homomorphic encryption. However, it requires distributed key generation and uses heavy public-key machinery throughout the protocol. More recently, Ishai, Prabhakaran and Sahai [IPS09] and the aforementioned DO protocol show more efficient solutions. Although the techniques used are completely different, the asymptotic complexities are similar, but the constants are significantly smaller in the DO solution, which was the most practical protocol proposed so far.

Notation: We let U_S denote the uniform distribution over the set S. We use $x \leftarrow X$ to denote the process of sampling x from the distribution X or, if X is a set, a uniform choice from it.

We say that a function $f : \mathbb{N} \to \mathbb{R}$ is negligible if $\forall c, \exists n_c$ s.t. if $n > n_c$ then $f(n) < n^{-c}$. We will use $\varepsilon(\cdot)$ to denote an unspecified negligible function.

For $p \in \mathbb{N}$, we represent $\mathbb{Z}_p$ by the numbers $\{-\lfloor (p-1)/2 \rfloor, \ldots, \lceil (p-1)/2 \rceil\}$. If $\mathbf{x}$ is an m-dimensional vector, $||\mathbf{x}||_\infty := \max(|x_1|, \ldots, |x_m|)$. Unless differently specified, all the logarithms are in base 2.

As a general convention: lowercase letters $a, b, c, \ldots$ represent integers and capital letters $A, B, C, \ldots$ ciphertexts. Bold lowercase letters $\mathbf{r}, \mathbf{s}, \ldots$ are vectors and bold capitals $\mathbf{M}, \mathbf{A}, \ldots$ are matrices. We call κ the computational security parameter and u the statistical security parameter. In practice u can be set to be much smaller than κ, as it does not depend on the computing power of the adversary.

2 The Framework for Semi-homomorphic Encryption

In this section we introduce a framework for public-key cryptosystems, that satisfy a relaxed version of the *additive homomorphic property*. Let $\mathsf{PKE} = (\mathsf{G}, \mathsf{E}, \mathsf{D})$ be a tuple of algorithms where:

$\mathsf{G}(1^\kappa, p)$ is a randomized algorithm that takes as input a security parameter κ and a modulus p[2]; It outputs a public/secret key pair (pk, sk) and a set of parameters $\mathbb{P} = (p, M, R, \mathcal{D}_\sigma^d, \mathbb{G})$. Here, M, R are integers, $\mathcal{D}_\sigma^d$ is the description

[2] In the framework there are no restrictions for the choice of p; however in the next sections p will always be chosen to be a prime.

of a randomized algorithm producing as output d-vectors with integer entries (to be used as randomness for encryption). We require that except with negligible probability, $\mathcal{D}_\sigma^d$ will always output $\mathbf{r}$ with $||\mathbf{r}||_\infty \leq \sigma$, for some $\sigma < R$ that may depend on κ. Finally, $\mathbb{G}$ is the abelian group where the ciphertexts belong (written in additive notation). For practical purposes one can think of M and R to be of size super-polynomial in κ, and p and σ as being much smaller than M and R respectively. We will assume that every other algorithm takes as input the parameters $\mathbb{P}$, without specifying this explicitly.

$\mathsf{E}_{pk}(x, \mathbf{r})$ is a deterministic algorithm that takes as input an integer $x \in \mathbb{Z}$ and a vector $\mathbf{r} \in \mathbb{Z}^d$ and outputs a ciphertext $C \in \mathbb{G}$. We sometimes write $\mathsf{E}_{pk}(x)$ when it is not important to specify the randomness explicitly. Given $C_1 = \mathsf{E}_{pk}(x_1, \mathbf{r}_1)$, $C_2 = \mathsf{E}_{pk}(x_2, \mathbf{r}_2)$ in $\mathbb{G}$, we have $C_1 + C_2 = \mathsf{E}_{pk}(x_1 + x_2, \mathbf{r}_1 + \mathbf{r}_2)$. In other words, $\mathsf{E}_{pk}(\cdot, \cdot)$ is a homomorphism from $(\mathbb{Z}^{d+1}, +)$ to $(\mathbb{G}, +)$). Given some τ and ρ we call C a (τ, ρ)-*ciphertext* if there exists $x, \mathbf{r}$ with $|x| \leq \tau$ and $||\mathbf{r}||_\infty \leq \rho$ such that $C = \mathsf{E}_{pk}(x, \mathbf{r})$. Note that given a ciphertext τ and ρ are not unique. When we refer to a (τ, ρ)-ciphertext, τ and ρ should be interpreted as an upper limit to the size of the message and randomness contained in the ciphertext.

$\mathsf{D}_{sk}(C)$ is a deterministic algorithm that takes as input a ciphertext $C \in \mathbb{G}$ and outputs $x' \in \mathbb{Z}_p \cup \{\perp\}$.

We say that a semi-homomorphic encryption scheme PKE is *correct* if, $\forall p$:

$$\begin{aligned} \mathsf{Pr}[\ (pk, sk, \mathbb{P}) \leftarrow \mathsf{G}(1^\kappa, p),\ x \in \mathbb{Z}, |x| \leq M; \mathbf{r} \in \mathbb{Z}^d, ||\mathbf{r}||_\infty \leq R : \\ \mathsf{D}_{sk}(\mathsf{E}_{pk}(x, \mathbf{r})) \neq x \bmod p\ ] \qquad < \varepsilon(\kappa) \end{aligned}$$

where the probabilities are taken over the random coins of G and E.

We now define the IND-CPA security game for a semi-homomorphic cryptosystem. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a PPT TM, then we run the following experiment:

$$\begin{aligned} (pk, sk, \mathbb{P}) &\leftarrow \mathsf{G}(1^\kappa, p) \\ (m_0, m_1, \mathsf{state}) &\leftarrow \mathcal{A}_1(1^\kappa, pk) \text{ with } m_0, m_1 \in \mathbb{Z}_p \\ b \leftarrow \{0,1\},\ C &\leftarrow \mathsf{E}_{pk}(m_b),\ b' \leftarrow \mathcal{A}_2(1^\kappa, \mathsf{state}, C) \end{aligned}$$

We define the advantage of $\mathcal{A}$ as $\mathsf{Adv}^{\mathtt{CPA}}(\mathcal{A}, \kappa) = |\,\mathsf{Pr}[b = b'] - 1/2|$, where the probabilities are taken over the random choices of $\mathsf{G}, \mathsf{E}, \mathcal{A}$ in the above experiment. We say that PKE is IND-CPA *secure* if $\forall$ PPT $\mathcal{A}$, $\mathsf{Adv}^{\mathtt{CPA}}(\mathcal{A}, \kappa) < \varepsilon(\kappa)$.

Next, we discuss the motivation for the way this framework is put together: when in the following, honest players encrypt data, plaintext x will be chosen in $\mathbb{Z}_p$ and the randomness $\mathbf{r}$ according to $\mathcal{D}_\sigma^d$. This ensures IND-CPA security and also that such data can be decrypted correctly, since by assumption on $\mathcal{D}_\sigma^d$, $||\mathbf{r}||_\infty \leq \sigma \leq R$. However, we also want that a (possibly dishonest) player P_i is committed to x by publishing $C = \mathsf{E}_{pk}(x, \mathbf{r})$. We are not able to force a player to choose x in $\mathbb{Z}_p$, nor that $\mathbf{r}$ is sampled with the correct distribution. But our zero-knowledge protocols *can* ensure that C is a (τ, ρ)-ciphertext, for concrete values of τ, ρ. If $\tau < M, \rho < R$, then correctness implies that C commits P_i to $x \bmod p$, even if $x, \mathbf{r}$ may not be uniquely determined from C.

2.1 Examples of Semi-homomorphic Encryption

Regev's cryptosystem [Reg05] is parametrized by p, q, m and α, and is given by $(\mathsf{G}, \mathsf{E}, \mathsf{D})$. A variant of the system was also given in [BD10], where parameters are chosen slightly differently than in the original. In both [Reg05] and [BD10] only a single bit was encrypted, it is quite easy, though, to extend it to elements of a bigger ring. It is this generalized version of the variant in [BD10] that we describe here. All calculations are done in $\mathbb{Z}_q$. Key generation $\mathsf{G}(1^\kappa)$ is done by sampling $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ uniformly at random and $\mathbf{x} \in \mathbb{Z}_q^m$ from a discrete Gaussian distribution with mean 0 and standard deviation $\frac{q\alpha}{\sqrt{2\pi}}$. We then have the key pair $(pk, sk) = ((\mathbf{A}, \mathbf{As} + \mathbf{x}), \mathbf{s})$. Encryption of a message $\gamma \in \mathbb{Z}_p$ is done by sampling a uniformly random vector $\mathbf{r} \in \{-1, 0, 1\}^m$. A ciphertext C is then given by $C = \mathsf{E}_{pk}(\gamma, \mathbf{r}) = (\mathbf{a}, b) = (\mathbf{A}^T\mathbf{r}, (\mathbf{As} + \mathbf{x})^T\mathbf{r} + \gamma \lfloor q/p \rceil)$. Decryption is given by $\mathsf{D}_{sk}(C) = \lfloor (b - \mathbf{s}^T\mathbf{a}) \cdot p/q \rceil$. Regev's cryptosystem works with a decryption error, which can, however, be made negligibly small when choosing the parameters.

Fitting the cryptosystem to the framework is quite straight forward. The group $\mathbb{G} = \mathbb{Z}_q^n \times \mathbb{Z}_q$ and p is just the same. The distribution $\mathcal{D}_\sigma^d$ from which the randomness $\mathbf{r}$ is taken is the uniform distribution over $\{-1, 0, 1\}^m$, that is $d = m$ and $\sigma = 1$. Given two ciphertexts $(\mathbf{a}, b)$ and $(\mathbf{a}', b')$ we define addition to be $(\mathbf{a}+\mathbf{a}', b+b')$. With this definition it follows quite easily that the homomorphic property holds. Due to the choices of message space and randomness distribution in Regev's cryptosystem, we will always have that the relation $M = Rp/2$ should hold. How M can be chosen, and thereby also R, depends on all the original parameters of the cryptosystem. First assume that $q \cdot \alpha = \sqrt[d]{q}$ with $d > 1$. Furthermore we will need that $p \leq q/(4\sqrt[c]{q})$ for some constant $c < d$. Then to bound M we should have first that $M < q/(4p)$ and secondly that $M < p\sqrt[s]{q}/(2m)$ for some $s > cd/(d-c)$. Obtaining these bounds requires some tedious computation which we leave out here.

In *Paillier's cryptosystem* [Pai99] the secret key is two large primes p_1, p_2, the public key is $N = p_1p_2$, and the encryption function is $\mathsf{E}_{pk}(x, r) = (N + 1)^x r^N \bmod N^2$ where $x \in Z_N$ and r is random in $\mathbb{Z}_{N^2}^*$. The decryption function D'_{sk} reconstructs correctly any plaintext in $\mathbb{Z}_N$, and to get a semi-homomorphic scheme modulo p, we simply redefine the decryption as $\mathsf{D}(c) = \mathsf{D}'(c) \bmod p$. It is not hard to see that we get a semi-homomorphic scheme with $M = (N - 1)/2, R = \infty, d = 1, \mathcal{D}_\sigma^d = U_{\mathbb{Z}_{N^2}^*}, \sigma = \infty$ and $\mathbb{G} = \mathbb{Z}_{N^2}^*$. In particular, note that we do not need to bound the size of the randomness, hence we set $\sigma = R = \infty$.

The cryptosystem looks syntactically a bit different from our definition which writes $\mathbb{G}$ additively, while $Z_{N^2}^*$ is usually written with multiplicative notation; also for Paillier we have $\mathsf{E}_{pk}(x, r) + \mathsf{E}_{pk}(x', r) = \mathsf{E}_{pk}(x + x', r \cdot r')$ and not $\mathsf{E}_{pk}(x + x', r + r')$. However, this makes no difference in the following, except that it actually makes some of the zero-knowledge protocols simpler (more details in Section 2.2). It is easy to see that the generalization of Paillier in [DJ01] can be modified in a similar way to be semi-homomorphic.

In the full paper [BDOZ10] we show how several other cryptosystems are semi-homomorphic.

2.2 Zero-Knowledge Proofs

We present two zero-knowledge protocols, Π_{PoPK}, Π_{PoCM} where a prover P proves to a verifier V that some ciphertexts are correctly computed and that some ciphertexts satisfy a multiplicative relation respectively. Π_{PoPK} has (amortized) complexity $O(\kappa + u)$ bits per instance proved, where the soundness error is 2^{-u}. Π_{PoCM} has complexity $O(\kappa u)$. We also show a more efficient version of Π_{PoCM} that works only for Paillier encryption, with complexity $O(\kappa + u)$. Finally, in the full paper [BDOZ10], we define the *multiplication security* property that we conjecture is satisfied for all our example cryptosystems after applying a simple modification. We show that assuming this property, Π_{PoCM} can be replaced by a different check that has complexity $O(\kappa + u)$.

Π_{PoPK} and Π_{PoCM} will both be of the standard 3-move form with a random u-bit challenge, and so they are honest verifier zero-knowledge. To achieve zero-knowledge against an arbitrary verifier standard techniques can be used. In particular, in our MPC protocol we will assume – only for the sake of simplicity – a functionality $\mathcal{F}_{\text{RAND}}$ that generates random challenges on demand. The $\mathcal{F}_{\text{RAND}}$ functionality is specified in detail in the full paper [BDOZ10] and can be implemented in our key registration model using only semi-homomorphic encryption. In the protocols both prover and verifier will have public keys pk_P and pk_V. By $\mathsf{E}_P(a, \mathbf{r})$ we denote an encryption under pk_P, similarly for $\mathsf{E}_V(a, \mathbf{r})$.

We emphasize that the zero-knowledge property of our protocols does not depend on IND-CPA security of the cryptosystem, instead it follows from the homomorphic property and the fact that the honest prover creates, for the purpose of the protocol, some auxiliary ciphertexts containing enough randomness to hide the prover's secrets.

Proof of Plaintext Knowledge. Π_{PoPK} takes as common input u ciphertexts C_k, $k = 1, \ldots, u$. If these are (τ, ρ)-ciphertexts, the protocol is complete and statistical zero-knowledge. The protocol is sound in the following sense: assuming that pk_P is well-formed, if P is corrupt and can make V accept with probability larger than 2^{-u}, then all the C_k are $(2^{2u+\log u}\tau, 2^{2u+\log u}\rho)$-ciphertexts. The protocol is also a proof of knowledge with knowledge error 2^{-u} that P knows correctly formed plaintexts and randomness for all the C_k's.

In other words, Π_{PoPK} is a ZKPoK for the following relation, *except* that zero-knowledge and completeness only hold if the C_k's satisfy the stronger condition of being (τ, ρ)-ciphertexts. However, this is no problem in the following: the prover will always create the C_k's himself and can therefore ensure that they are correctly formed if he is honest.

$$R_{\text{PoPK}}^{(u,\tau,\rho)} = \{(x, w) | \quad x = (pk_P, C_1, \ldots, C_u); \\ w = ((x_1, \mathbf{r}_1), \ldots, (x_u, \mathbf{r}_u)) : C_k = \mathsf{E}_P(x_k, \mathbf{r}_k), \\ |x_k| \leq 2^{2u+\log u}\tau, ||\mathbf{r}_k||_\infty \leq 2^{2u+\log u}\rho\}$$

We use the approach of [CD09] to get small amortized complexity of the zero-knowledge proofs, and thereby gaining efficiency by performing the proofs on u

simultaneous instances. In the following we define $m = 2u-1$, furthermore $\mathbf{M_e}$ is an $m \times u$ matrix constructed given a uniformly random vector $\mathbf{e} = (e_1, \ldots, e_u) \in \{0,1\}^u$. Specifically the (i,k)-th entry $\mathbf{M}_{\mathbf{e},i,k}$ is given by $\mathbf{M}_{\mathbf{e},i,k} = \mathbf{e}_{i-k+1}$ for $1 \leq i-k+1 \leq u$ and 0 otherwise. By $\mathbf{M}_{\mathbf{e},i}$ we denote the i-th row of $\mathbf{M_e}$. The protocol can be seen in Figure 1. Completeness and zero-knowledge follow by standard arguments that can be found in the full paper [BDOZ10]. Here we argue soundness which is the more interesting case: Assume we are given any prover P^*, and consider the case where P^* can make V accept for both $\mathbf{e}$ and $\mathbf{e}'$, $\mathbf{e} \neq \mathbf{e}'$, by sending $\mathbf{z}$, $\mathbf{z}'$, $\mathbf{T}$ and $\mathbf{T}'$ respectively. We now have the following equation:

$$(\mathbf{M_e} - \mathbf{M_{e'}})\mathbf{c} = (\mathbf{d} - \mathbf{d}') \tag{1}$$

What we would like is to find $\mathbf{x} = (x_1, \ldots, x_u)$ and $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_u)$ such that $C_k = \mathsf{E}_P(x_k, \mathbf{r}_k)$. We can do this by viewing (1) as a system of linear equations. First let j be the biggest index such that $\mathbf{e}_j \neq \mathbf{e}'_j$. Now look at the $u \times u$ submatrix of $\mathbf{M_e} - \mathbf{M_{e'}}$ given by the rows j through $j+u$ both included. This is an upper triangular matrix with entries in $\{-1, 0, 1\}$ and $\mathbf{e}_j - \mathbf{e}'_j \neq 0$ on a diagonal. Now remember the form of the entries in the vectors $\mathbf{c}$, $\mathbf{d}$ and $\mathbf{d}'$, we have $C_k = \mathsf{E}_P(x_k, \mathbf{r}_k)$, $D_k = \mathsf{E}_P(z_k, \mathbf{t}_k)$, $D'_k = \mathsf{E}_P(z'_k, \mathbf{t}'_k)$. We can now directly solve the equations for the x_k's and the $\mathbf{r}_k$'s by starting with C_u and going up. We give examples of the first few equations (remember we are going bottom up). For simplicity we will assume that all entries in $\mathbf{M_e} - \mathbf{M_{e'}}$ will be 1.

$$\begin{aligned} \mathsf{E}_P(x_u, \mathbf{r}_u) &= \mathsf{E}_P(z_{u+j} - z'_{u+j}, \mathbf{t}_{u+j} - \mathbf{t}'_{u+j}) \\ \mathsf{E}_P(x_{u-1}, \mathbf{r}_{u-1}) + \mathsf{E}_P(x_u, \mathbf{r}_u) &= \mathsf{E}_P(z_{u+j-1} - z'_{u+j-1}, \mathbf{t}_{u+j-1} - \mathbf{t}'_{u+j-1}) \\ &\vdots \end{aligned}$$

Since we know all values used on the right hand sides and since the cryptosystem used is additively homomorphic, it should now be clear that we can find x_k and $\mathbf{r}_k$ such that $C_k = \mathsf{E}_P(x_k, \mathbf{r}_k)$. A final note should be said about what we can guarantee about the sizes of x_k and $\mathbf{r}_k$. Knowing that $|z_i| \leq 2^{u-1+\log u}\tau$, $|z'_i| \leq 2^{u-1+\log u}\tau$, $||\mathbf{t}_i||_\infty \leq 2^{u-1+\log u}\rho$ and $||\mathbf{t}'_i||_\infty \leq 2^{u-1+\log u}\rho$ we could potentially have that C_1 would become a $(2^{2u+\log u}\tau, 2^{2u+\log u}\rho)$ ciphertext. Thus this is what we can guarantee.

Proof of Correct Multiplication. $\Pi_{\text{PoCM}}(u, \tau, \rho)$ takes as common input u triples of ciphertexts (A_k, B_k, C_k) for $k = 1, \ldots, u$, where A_k is under pk_P and B_k and C_k are under pk_V (and so are in the group $\mathbb{G}_V$). If P is honest, he will know a_k and $a_k \leq \tau$. Furthermore P has created C_k as $C_k = a_k B_k + \mathsf{E}_V(r_k, \mathbf{t}_k)$, where $\mathsf{E}_V(r_k, \mathbf{t}_k)$ is a random $(2^{3u+\log u}\tau^2, 2^{3u+\log u}\tau\rho)$-ciphertext. Under these assumptions the protocol is zero-knowledge.

Jumping ahead, we note that in the context where the protocol will be used, it will always be known that B_k in every triple is a $(2^{2u+\log u}\tau, 2^{2u+\log u}\rho)$-ciphertext, as a result of executing Π_{PoPK}. The choice of sizes for $\mathsf{E}_V(r_k, \mathbf{t}_k)$ then ensures that C_k is statistically close to a random $(2^{3u+\log u}\tau^2, 2^{3u+\log u}\tau\rho)$-ciphertext, and so reveals no information on a_k to V.

Subprotocol Π_{PoPK}: Proof of Plaintext Knowledge

PoPK(u, τ, ρ)**:**

1. The input is u ciphertexts $\{C_k = \mathsf{E}_P(x_k, \mathbf{r}_k)\}_{k=1}^u$. We define the vectors $\mathbf{c} = (C_1, \ldots, C_u)$ and $\mathbf{x} = (x_1, \ldots, x_u)$ and the matrix $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_u)$, where the $\mathbf{r}_k$'s are rows.
2. P constructs m $(2^{u-1+\log u}\tau, 2^{u-1+\log u}\rho)$-ciphertexts $\{A_i = \mathsf{E}_P(y_i, \mathbf{s}_i)\}_{i=1}^m$, and sends them to V. We define vectors $\mathbf{a}$ and $\mathbf{y}$ and matrix $\mathbf{S}$ as above.
3. V chooses a uniformly random vector $\mathbf{e} = (e_1, \ldots, e_u) \in \{0,1\}^u$, and sends it to P.
4. Finally P computes and sends $\mathbf{z} = \mathbf{y} + \mathbf{M_e} \cdot \mathbf{x}$ and $\mathbf{T} = \mathbf{S} + \mathbf{M_e} \cdot \mathbf{R}$ to V.
5. V checks that $\mathbf{d} = \mathbf{a} + \mathbf{M_e} \cdot \mathbf{c}$ where $\mathbf{d} = (\mathsf{E}_P(z_1, \mathbf{t}_1), \ldots, \mathsf{E}_P(z_m, \mathbf{t}_m))$. Furthermore, V checks that $|z_i| \leq 2^{u-1+\log u}\tau$ and $||\mathbf{t}_i||_\infty \leq 2^{u-1+\log u}\rho$.

Fig. 1. Proof of Plaintext Knowledge

Subprotocol Π_{PoCM}: Proof of Correct Multiplication

PoCM(u, τ, ρ)**:**

1. The input is u triples of ciphertexts $\{(A_k, B_k, C_k)\}_{k=1}^u$, where $A_k = \mathsf{E}_P(a_k, \mathbf{h}_k)$ and $C_k = a_k B_k + \mathsf{E}_V(r_k, \mathbf{t}_k)$.
2. P constructs u uniformly random $(2^{3u-1+\log u}\tau, 2^{3u-1+\log u}\rho)$-ciphertexts $D_k = \mathsf{E}_P(d_k, \mathbf{s}_k)$ and u ciphertexts $F_k = d_k B_k + \mathsf{E}_V(f_k, \mathbf{y}_k)$, where $\mathsf{E}_V(f_k, \mathbf{y}_k)$ are uniformly random $(2^{4u-1+\log u}\tau^2, 2^{4u-1+\log u}\tau\rho)$-ciphertexts.
3. V chooses u uniformly random bits e_k and sends them to P.
4. P returns $\{(z_k, \mathbf{v}_k)\}_{k=1}^u$ and $\{(x_k, \mathbf{w}_k)\}_{k=1}^u$ to V. Here $z_k = d_k + e_k a_k$, $\mathbf{v}_k = \mathbf{s}_k + e_k \mathbf{h}_k$, $x_k = f_k + e_k r_k$ and $\mathbf{w}_k = \mathbf{y}_k + e_k \mathbf{t}_k$.
5. V checks that $D_k + e_k A_k = \mathsf{E}_P(z_k, \mathbf{v}_k)$ and that $F_k + e_k C_k = z_k B_k + \mathsf{E}_V(x_k, \mathbf{w}_k)$. Furthermore, he checks that $|z_k| \leq 2^{3u-1+\log u}\tau$, $||\mathbf{v}_k||_\infty \leq 2^{3u-1+\log u}\rho$, $|x_k| \leq 2^{4u-1+\log u}\tau^2$ and $||\mathbf{w}_k||_\infty \leq 2^{4u-1+\log u}\tau\rho$.
6. Step 2-5 is repeated in parallel u times.

Fig. 2. Proof of Correct Multiplication

Summarizing, Π_{PoCM} is a ZKPoK for the relation (under the assumption that pk_P, pk_V are well-formed):

$$
\begin{aligned}
R_{\text{PoCM}}^{(u,\tau,\rho)} = \{(x, w) \mid \quad & x = (pk_P, pk_V, (A_1, B_1, C_1), \ldots, (A_u, B_u, C_u)); \\
& w = ((a_1, \mathbf{h}_1, r_1, \mathbf{t}_1), \ldots, (a_u, \mathbf{h}_u, r_u, \mathbf{t}_u)) : \\
& A_k = \mathsf{E}_P(a_k, \mathbf{h}_k), B_k \in \mathbb{G}_V, C_k = a_k B_k + \mathsf{E}_V(r_k, \mathbf{t}_k), \\
& |a_k| \leq 2^{3u+\log u}\tau, ||\mathbf{h}_k||_\infty \leq 2^{3u+\log u}\rho, \\
& |r_k| \leq 2^{4u+\log u}\tau^2, ||\mathbf{t}_k||_\infty \leq 2^{4u+\log u}\tau\rho)\}
\end{aligned}
$$

The protocol can be seen in Figure 2. Note that Step 6 could also be interpreted as choosing e_k as a u-bit vector instead, thereby only calling $\mathcal{F}_{\text{RAND}}$ once. Completeness, soundness and zero-knowledge follow by standard arguments that can be found in the full paper [BDOZ10].

Zero-Knowledge Protocols for Paillier. For the particular case of Paillier encryption, Π_{PoPK} can be used as it is, except that there is no bound required on the randomness, instead all random values used in encryptions are expected to be in $\mathbb{Z}^*_{N^2}$. Thus, the relations to prove will only require that the random values are in $\mathbb{Z}^*_{N^2}$ and this is also what the verifier should check in the protocol.

For Π_{PoCM} we sketch a version that is more efficient than the above, using special properties of Paillier encryption. In order to improve readability, we depart here from the additive notation for operations on ciphertexts, since multiplicative notation is usually used for Paillier. In the following, let $pk_V = N$. Note first that based on such a public key, one can define an unconditionally hiding commitment scheme with public key $g = \mathsf{E}_V(0)$. To commit to $a \in \mathbb{Z}_N$, one sends $\mathsf{com}(a, r) = g^a r^N \bmod N$, for random $r \in \mathbb{Z}^*_{N^2}$. One can show that the scheme is binding assuming it is hard to extract N-th roots modulo N^2 (which must be the case if Paillier encryption is secure).

We restate the relation $R^{(u,\tau,\rho)}_{\text{PoCM}}$ from above as it will look for the Paillier case, in multiplicative notation and without bounds on the randomness:

$$
\begin{aligned}
R^{(\tau,\rho)}_{\text{PoCM},Paillier} = \{(x,w) \mid \quad & x = (pk_P, pk_V, (A_1, B_1, C_1), \ldots, (A_u, B_u, C_u)); \\
& w = ((a_1, h_1, r_1, t_1), \ldots, (a_u, h_u, r_u, t_u)) : \\
& A_k = \mathsf{E}_P(a_k, h_k), B_k \in \mathbb{Z}_{N^2}, C_k = B_k^{a_k} \cdot \mathsf{E}_V(r_k, t_k), \\
& |a_k| \leq 2^{2u + \log u}\tau, |r_k| \leq 2^{5u+2\log u}\tau^2\}
\end{aligned}
$$

The idea for the proof of knowledge for this relation is now to ask the prover to also send commitments $\Psi_k = \mathsf{com}(a_k, \alpha_k), \Phi_k = \mathsf{com}(r_k, \beta_k), k = 1 \ldots u$ to the r_k's and a_k's. Now, the prover must first provide a proof of knowledge that for each k: 1) the same bounded size value is contained in both A_k and Ψ_k, and that 2) a bounded size value is contained in Φ_k. The proof for $\{\Phi_k\}$ is simply Π_{PoPK} since a commitment has the same form as an encryption (with $(N + 1)$ replaced by g). The proof for $\{\Psi_k, A_k\}$ is made of two instances of Π_{PoPK} run in parallel, using the same challenge e and responses z_i in both instances. Finally, the prover must show that C_k can be written as $C_k = B_k^{a_k} \cdot \mathsf{E}_V(r_k, t_k)$, where a_k is the value contained in Ψ_k and r_k is the value in Φ_k. Since all commitments and ciphertexts live in the same group $\mathbb{Z}^*_{N^2}$, where $pk_V = N$, we can do this efficiently using a variant of a protocol from [CDN01]. The resulting protocol is shown in Figure 3.

Completeness of the protocol in steps 1-4 of Figure 3 is straightforward by inspection. Honest verifier zero-knowledge follows by the standard argument: choose e and the prover's responses uniformly in their respective domains and use the equations checked by the verifier to compute a matching first message D, X, Y. This implies completeness and honest verifier zero-knowledge for the overall protocol, since the subprotocols in steps 2 and 3 have these properties as well.

Subprotocol Π_{PoCM}: Proof of Correct Multiplication (only for Paillier)

1. P sends $\Psi_k = \mathsf{com}(a_k, \alpha_k), \Phi_k = \mathsf{com}(r_k, \beta_k), k = 1, \ldots, u$ to the verifier.
2. P uses Π_{PoPK} on Φ_k to prove that, even if P is corrupted, each Φ_k contains a value r_k with $|r_k| \leq 2^{5u+2\log u}\tau^2$.
3. P uses Π_{PoPK} in parallel on (A_k, Ψ_k) (where V uses the same $\mathbf{e}$ in both runs) to prove that, even if P is corrupted, Ψ_k and A_k contains the same value a_k and $|a_k| \leq 2^{2u+\log u}\tau$.
4. To show that the C_k's are well-formed, we do the following for each k:
   (a) P picks random $x, y, v, \gamma, \delta \leftarrow \mathbb{Z}^*_{N^2}$ and sends $D = B_k^x \mathsf{E}_V(y, v)$, $X = \mathsf{com}(x, \gamma_x), Y = \mathsf{com}(y, \gamma_y)$ to V.
   (b) V sends a random u-bit challenge e.
   (c) P computes $z_a = x + ea_k \bmod N, z_r = y + er_k \bmod N$.
   He also computes q_a, q_r, where $x + ea = q_a N + z_a$, $y + er_k = q_r N + z_r$[a].
   P sends z_a, z_r, $w = vs_k^e B_k^{q_a} \bmod N^2$, $\delta_a = \gamma_x \alpha_k^e g^{q_a} \bmod N^2$, and $\delta_r = \gamma_y \beta_k^e g^{q_r} \bmod N^2$ to V.
   (d) V accepts if $DC_k^e = B_k^{z_a} \mathsf{E}_V(z_r, w) \bmod N^2 \wedge X\Psi_k^e = \mathsf{com}(z_a, \delta_a) \bmod N^2 \wedge Y\Phi_k^e = \mathsf{com}(z_r, \delta_r) \bmod N^2$.

[a] Since g and B_k do not have order N, we need to explicitly handle the quotients q_a and q_r, in order to move the "excess multiples" of N into the randomness parts of the commitments and ciphertext.

Fig. 3. Proof of Correct Multiplication for Paillier encryption

Finally, soundness follows by assuming we are given correct responses in step 7 to two different challenges. From the equations checked by the verifier, we can then easily compute $a_k, \alpha_k, r_k, \beta_k, s_k$ such that $\Psi_k = \mathsf{com}(a_k, \alpha_k), \Phi_k(r_k, \beta_k), C_k = B_k^{a_k} \mathsf{E}_V(r_k, s_k)$. Now, by soundness of the protocols in steps 2 and 3, we can also compute bounded size values a'_k, r'_k that are contained in Ψ_k, Φ_k. By the binding property of the commitment scheme, we have $r'_k = r_k, a'_k = a_k$ except with negligible probability, so we have a witness as required in the specification of the relation.

3 The Online Phase

Our goal is to implement reactive arithmetic multiparty computation over $\mathbb{Z}_p$ for a prime p of size super-polynomial in the statistical security parameter u. The (standard) ideal functionality $\mathcal{F}_{\text{AMPC}}$ that we implement can be seen in Figure 6. We assume here that the parties already have a functionality for synchronous[3], secure communication and broadcast.

[3] A malicious adversary can always stop sending messages and, in any protocol for dishonest majority, all parties are required for the computation to terminate. Without synchronous channels the honest parties might wait forever for the adversary to send his messages. Synchronous channels guarantee that the honest parties can detect that the adversary is not participating anymore and therefore they can abort the protocol. If termination is not required, the protocol can be implemented over an asynchronous network instead.

We first present a protocol for an *online phase* that assumes access to a functionality $\mathcal{F}_{\text{TRIP}}$ which we later show how to implement using an *offline protocol*. The online phase is based on a representation of values in $\mathbb{Z}_p$ that are shared additively where shares are authenticated using information theoretic message authentication codes (MACs). Before presenting the protocol we introduce how the MACs work and how they are included in the representation of a value in $\mathbb{Z}_p$. Furthermore, we argue how one can compute with these representations as we do with simple values, and in particular how the relation to the MACs are maintained.

In the rest of this section, all additions and multiplications are to be read modulo p, even if not specified. The number of parties is denoted by n, and we call the parties $P_1, \ldots, P_n$.

3.1 The MACs

A key K in this system is a random pair $K = (\alpha, \beta) \in \mathbb{Z}_p^2$, and the authentication code for a value $a \in \mathbb{Z}_p$ is $\text{MAC}_K(a) = \alpha a + \beta \bmod p$.

We will apply the MACs by having one party P_i hold $a, \text{MAC}_K(a)$ and another party P_j holding K. The idea is to use the MAC to prevent P_i from lying about a when he is supposed to reveal it to P_j. It will be very important in the following that if we keep α constant over several different MAC keys, then one can add two MACs and get a valid authentication code for the sum of the two corresponding messages. More concretely, two keys $K = (\alpha, \beta), K' = (\alpha', \beta')$ are said to be *consistent* if $\alpha = \alpha'$. For consistent keys, we define $K + K' = (\alpha, \beta + \beta')$ so that it holds that $\text{MAC}_K(a) + \text{MAC}_{K'}(a') = \text{MAC}_{K+K'}(a + a')$.

The MACs will be used as follows: we give to P_i several different values $m_1, m_2, \ldots$ with corresponding MACs $\gamma_1, \gamma_2, \ldots$ computed using keys $K_i = (\alpha, \beta_i)$ that are random but consistent. It is then easy to see that if P_i claims a false value for any of the m_i's (or a linear combination of them) he can guess an acceptable MAC for such a value with probability at most $1/p$.

3.2 The Representation and Linear Computation

To represent a value $a \in \mathbb{Z}_p$, we will give a share a_i to each party P_i. In addition, P_i will hold MAC keys $K^i_{a_1}, \ldots, K^i_{a_n}$. He will use key $K^i_{a_j}$ to check the share of P_j, if we decide to make a public. Finally, P_i also holds a set of authentication codes $\text{MAC}_{K^j_{a_i}}(a_i)$. We will denote $\text{MAC}_{K^j_{a_i}}(a_i)$ by $m_j(a_i)$ from now on. Party P_i will use $m_j(a_i)$ to convince P_j that a_i is correct, if we decide to make a public. Summing up, we have the following way of representing a:

$$[a] = [\{a_i, \{K^i_{a_j}, m_j(a_i)\}_{j=1}^n\}_{i=1}^n]$$

where $\{a_i, \{K^i_{a_j}, m_j(a_i)\}_{j=1}^n\}$ is the information held privately by P_i, and where we use $[a]$ as shorthand when it is not needed to explicitly talk about the shares and MACs. We say that $[a] = [\{a_i, \{K^i_{a_j}, m_j(a_i)\}_{j=1}^n\}_{i=1}^n]$ is *consistent*, with $a = \sum_i a_i$, if $m_j(a_i) = \text{MAC}_{K^j_{a_i}}(a_i)$ for all i, j. Two representations

$$[a] = [\{a_i, \{K^i_{a_j}, m_j(a_i)\}_{j=1}^n\}_{i=1}^n], \quad [a'] = [\{a'_i, \{K^i_{a'_j}, m_j(a'_i)\}_{j=1}^n\}_{i=1}^n]$$

Opening: We can reliably open a consistent representation to P_j: each P_i sends $a_i, m_j(a_i)$ to P_j. P_j checks that $m_j(a_i) = \mathrm{MAC}_{K^j_{a_i}}(a_i)$ and broadcasts *OK* or *fail* accordingly. If all is OK, P_j computes $a = \sum_i a_i$, else we abort. We can modify this to opening a value $[a]$ to all parties, by opening as above to every P_j.

Addition: Given two key-consistent representations as above we get that

$$[a + a'] = [\{a_i + a'_i, \{K^i_{a_j} + K^i_{a'_j}, m_j(a_i) + m_j(a'_i)\}_{j=1}^n\}_{i=1}^n]$$

is a consistent representation of $a+a'$. This new representation can be computed only by local operations.

Multiplication by constants: In a similar way, we can multiply a public constant δ "into" a representation. This is written $\delta[a]$ and is taken to mean that all parties multiply their shares, keys and MACs by δ. This gives a consistent representation $[\delta a]$.

Addition of constants: We can add a public constant δ into a representation. This is written $\delta + [a]$ and is taken to mean that P_1 will add δ to his share a_1. Also, each P_j will replace his key $K^j_{a_1} = (\alpha^j_1, \beta^j_{a_1})$ by $K^j_{a_1+\delta} = (\alpha^j_1, \beta^j_{a_1} - \delta\alpha^j_1)$. This will ensure that the MACs held by P_1 will now be valid for the new share $a_1 + \delta$, so we now have a consistent representation $[a + \delta]$.

Fig. 4. Operations on $[\cdot]$-representations

are said to be *key-consistent* if they are both consistent, and if for all i, j the keys $K^i_{a_j}, K^i_{a'_j}$ are consistent. We will want *all* representations in the following to be key-consistent: this is ensured by letting P_i use the same α_j-value in keys towards P_j throughout. Therefore the notation $K^i_{a_j} = (\alpha^i_j, \beta^i_{a_j})$ makes sense and we can compute with the representations, as detailed in Figure 4.

3.3 Triples and Multiplication

For multiplication and input sharing we will need both random single values $[a]$ and triples $[a], [b], [c]$ where a, b are random and $c = ab \bmod p$. Also, we assume that all singles and triples we ever produce are key consistent, so that we can freely add them together. More precisely, we assume we have access to an ideal functionality $\mathcal{F}_{\mathrm{TRIP}}$ providing us with the above. This is presented in Figure 5.

The principle in the specification of the functionality is that the environment is allowed to specify all the data that the corrupted parties should hold, including all shares of secrets, keys and MACs. Then, the functionality chooses the secrets to be shared and constructs the data for honest parties so it is consistent with the secrets and the data specified by the environment.

Thanks to this functionality we are also able to compute multiplications in the following way: If the parties hold two key-consistent representations $[x], [y]$, we can use one precomputed key-consistent triple $[a], [b], [c]$ (with $c = ab$) to compute a new representation of $[xy]$.

Functionality $\mathcal{F}_{\text{TRIP}}$

Initialize: On input $(init, p)$ from all parties the functionality stores the modulus p. For each corrupted party P_i the environment specifies values $\alpha_j^i, j = 1, \ldots, n$, except those α_j^i where both P_i and P_j are corrupt. For each honest P_i, it chooses $\alpha_j^i, j = 1, \ldots, n$ at random.

Singles: On input $(singles, u)$ from all parties P_i, the functionality does the following, for $v = 1, \ldots, u$:

1. It waits to get from the environment either "stop", or some data as specified below. In the first case it sends "fail" to all honest parties and stops. In the second case, the environment specifies for each corrupt party P_i, a share a_i and n pairs of values $(m_j(a_i), \beta_{a_j}^i), j = 1, \ldots, n$, except those $(m_j(a_i), \beta_{a_j}^i)$ where both P_i and P_j are corrupt.
2. The functionality chooses $a \in \mathbb{Z}_p$ at random and creates the representation $[a]$ as follows:
   (a) First it chooses random shares for the honest parties such that the sum of these and those specified by the environment is correct: Let $\mathtt{C}$ be the set of corrupt parties, then a_i is chosen at random for $P_i \notin \mathtt{C}$, subject to $a = \sum_i a_i$.
   (b) For each honest P_i, and $j = 1, \ldots, n$, $\beta_{a_j}^i$ is chosen as follows: if P_j is honest, $\beta_{a_j}^i$ is chosen at random, otherwise it sets $\beta_{a_j}^i = m_i(a_j) - \alpha_j^i a_j$. Note that the environment already specified $m_i(a_j), a_j$, so what is done here is to construct the key to be held by P_i to be consistent with the share and MAC chosen by the environment.
   (c) For all $i = 1, \ldots, n, j = 1, \ldots, n$ it sets $K_{a_j}^i = (\alpha_j^i, \beta_{a_j}^i)$, and computes $m_j(a_i) = \text{MAC}_{K_{a_i}^j}(a_i)$.
   (d) Now all data for $[a]$ is created. The functionality sends $\{a_i, \{K_{a_j}^i, m_j(a_i)\}_{j=1,\ldots,n}\}$ to each honest P_i (no need to send anything to corrupt parties, the environment already has the data).

Triples: On input $(triples, u)$ from all parties P_i, the functionality does the following, for $v = 1, \ldots, u$:

1. Step 1 is done as in "Singles".
2. For each triple to create it chooses a, b at random and sets $c = ab$. Now it creates representations $[a], [b], [c]$, each as in Step 2 in "Singles".

Fig. 5. The ideal functionality for making singles $[a]$ and triples $[a], [b], [c]$

To compute [xy] we first open $[x] - [a]$ to get a value ε, and $[y] - [b]$ to get δ. Then, we have $xy = (a + \varepsilon)(b + \delta) = c + \varepsilon b + \delta a + \varepsilon\delta$. Therefore, we get a new representation of xy as follows:

$$[xy] = [c] + \varepsilon[b] + \delta[a] + \varepsilon\delta.$$

Using the tools from the previous sections we can now construct a protocol Π_{AMPC} that securely implements the MPC functionality $\mathcal{F}_{\text{AMPC}}$ in the UC security framework. $\mathcal{F}_{\text{AMPC}}$ and Π_{AMPC} are presented in Figure 6 and Figure 7 respectively. The proof of Theorem 1 can be found in the full paper [BDOZ10].

Functionality $\mathcal{F}_{\text{AMPC}}$

Initialize: On input $(init, p)$ from all parties, the functionality activates and stores the modulus p.

Rand: On input $(rand, P_i, varid)$ from all parties P_i, with $varid$ a fresh identifier, the functionality picks $r \leftarrow \mathbb{Z}_p$ and stores $(varid, r)$.

Input: On input $(input, P_i, varid, x)$ from P_i and $(input, P_i, varid, ?)$ from all other parties, with $varid$ a fresh identifier, the functionality stores $(varid, x)$.

Add: On command $(add, varid_1, varid_2, varid_3)$ from all parties (if $varid_1, varid_2$ are present in memory and $varid_3$ is not), the functionality retrieves $(varid_1, x)$, $(varid_2, y)$ and stores $(varid_3, x + y \bmod p)$.

Multiply: On input $(multiply, varid_1, varid_2, varid_3)$ from all parties (if $varid_1, varid_2$ are present in memory and $varid_3$ is not), the functionality retrieves $(varid_1, x)$, $(varid_2, y)$ and stores $(varid_3, x \cdot y \bmod p)$.

Output: On input $(output, P_i, varid)$ from all parties (if $varid$ is present in memory), the functionality retrieves $(varid, x)$ and outputs it to P_i.

Fig. 6. The ideal functionality for arithmetic MPC

Protocol Π_{AMPC}

Initialize: The parties first invoke $\mathcal{F}_{\text{TRIP}}(init, p)$. Then, they invoke $\mathcal{F}_{\text{TRIP}}(triples, u)$ and $\mathcal{F}_{\text{TRIP}}(singles, u)$ a sufficient number of times to create enough singles and triples.

Input: To share P_i's input $[x_i]$ with identifier $varid$, P_i takes a single $[a]$ from the set of available ones. Then, the following is performed:

1. $[a]$ is opened to P_i.
2. P_i broadcasts $\delta = x_i - a$.
3. The parties compute $[x_i] = [a] + \delta$.

Rand: The parties take an available single $[a]$ and store with identifier $varid$.

Add: To add $[x], [y]$ with identifiers $varid_1, varid_2$ the parties compute $[z] = [x]+[y]$ and assign $[z]$ the identifier $varid_3$.

Multiply: To multiply $[x], [y]$ with identifiers $varid_1, varid_2$ the parties do the following:

1. They take a triple $([a], [b], [c])$ from the set of the available ones.
2. $[x] - [a] = \varepsilon$ and $[y] - [b] = \delta$ are opened.
3. They compute $[z] = [c] + \varepsilon[b] + \delta[a] + \varepsilon\delta$
4. They assign $[z]$ the identifier $varid_3$ and remove $([a], [b], [c])$ from the set of the available triples.

Output: To output $[x]$ with identifier $varid$ to P_i the parties do an opening of $[x]$ to P_i.

Fig. 7. The protocol for arithmetic MPC

Theorem 1. *In the $\mathcal{F}_{\text{TRIP}}$-hybrid model, the protocol Π_{AMPC} implements $\mathcal{F}_{\text{AMPC}}$ with statistical security against any static*[4]*, active adversary corrupting up to $n-1$ parties.*

4 The Offline Phase

In this section we describe the protocol Π_{TRIP} which securely implements the functionality $\mathcal{F}_{\text{TRIP}}$ described in Section 3 in the presence of two standard functionalities: a key registration functionality $\mathcal{F}_{\text{KeyReg}}$ and a functionality that generates random challenges $\mathcal{F}_{\text{Rand}}$[5]. Detailed specifications of these functionalities can be found in the full paper [BDOZ10].

4.1 $\langle\cdot\rangle$-Representation

Throughout the description of the offline phase, E_i will denote E_{pk_i} where pk_i is the public key of party P_i, as established by $\mathcal{F}_{\text{KeyReg}}$. We assume the cryptosystem used is semi-homomorphic modulo p, as defined in Section 2. In the following, we will always set $\tau = p/2$ and $\rho = \sigma$. Thus, if P_i generates a ciphertext $C = \mathsf{E}_i(x, \mathbf{r})$ where $x \in \mathbb{Z}_p$ and $\mathbf{r}$ is generated by $\mathcal{D}_\sigma^d$, C will be a (τ, ρ)-ciphertext. We will use the zero-knowledge protocols from Section 2.2. They depend on an "information theoretic" security parameter u controlling, e.g., the soundness error. We will say that a semi-homomorphic cryptosystem is *admissible* if it allows correct decryption of ciphertext produced in those protocols, that is, if $M \geq 2^{5u+2\log u}\tau^2$ and $R \geq 2^{4u+\log u}\tau\rho$.

In the following $\langle x_k \rangle$ will stand for the following representation of $x_k \in \mathbb{Z}_p$: each P_i has published $\mathsf{E}_i(x_{k,i})$ and holds $x_{k,i}$ privately, such that $x_k = \sum_i x_{k,i} \bmod p$. For the protocol to be secure, it will be necessary to ensure that the parties encrypt small enough plaintexts. For this purpose we use the Π_{PoPK} described in Section 2.2, and we get the protocol in Figure 8 to establish a set $\langle x_k \rangle, k = 1, \ldots, u$ of such random representations.

4.2 $\langle\cdot\rangle$-Multiplication

The final goal of the Π_{TRIP} protocol is to produce triples $[a_k], [b_k], [c_k]$ with $a_k b_k = c_k \bmod p$ in the $[\cdot]$-representation, but for now we will disregard the MACs and construct a protocol $\Pi_{n\text{-MULT}}$ which produces triples $\langle a_k \rangle, \langle b_k \rangle, \langle c_k \rangle$ in the $\langle\cdot\rangle$-representation[6].

We will start by describing a two-party protocol. Assume P_i is holding a set of u (τ, ρ)-encryptions $\mathsf{E}_i(x_k)$ under his public key and likewise P_j is holding u

[4] Π_{AMPC} can actually be shown to adaptively secure, but our implementation of $\mathcal{F}_{\text{TRIP}}$ will only be statically secure.

[5] $\mathcal{F}_{\text{Rand}}$ is only introduced for the sake of a cleaner presentation, and it could easily be implemented in the $\mathcal{F}_{\text{KeyReg}}$ model using semi-homomorphic encryption only.

[6] In fact, due to the nature of the MACs, the same protocol that is used to compute two-party multiplications will be used later in order to construct the MACs as well.

Subprotocol Π_{SHARE}

Share(u)**:**

1. Each P_i chooses $x_{k,i} \in \mathbb{Z}_p$ at random for $k = 1, \ldots, u$ and broadcasts (τ, ρ)-ciphertexts $\{\mathsf{E}_i(x_{k,i})\}_{k=1}^{u}$.
2. Each pair $P_i, P_j, i \neq j$, runs $\Pi_{\text{PoPK}}(u, \tau, \rho)$ with the $\mathsf{E}_i(x_{k,i})$'s as input. This proves that the ciphertexts are $(2^{2u+\log u}\tau, 2^{2u+\log u}\rho)$-ciphertexts.
3. All parties output $\langle x_k \rangle = (\mathsf{E}_1(x_{k,1}), \ldots, \mathsf{E}_n(x_{k,n}))$, for $k = 1, \ldots, u$, where x_k is defined by $x_k = \sum_i x_{k,i} \bmod p$. P_i keeps the $x_{k,i}$ and the randomness for his encryptions as private output.

Fig. 8. Subprotocol allowing parties to create random additively shared values

Subprotocol $\Pi_{2\text{-MULT}}$

2-Mult(u, τ, ρ)**:**

1. Honest P_i and P_j input (τ, ρ)-ciphertexts $\{\mathsf{E}_i(x_k)\}_{k=1}^{u}$, $\{\mathsf{E}_j(y_k)\}_{k=1}^{u}$. (At this point of the protocol it has already been verified that the ciphertexts are $(2^{2u+\log u}\tau, 2^{2u+\log u}\rho)$-ciphertexts.)
2. For each k, P_i sends $C_k = x_k\, \mathsf{E}_j(y_k) + \mathsf{E}_j(r_k)$ to P_j. Here $\mathsf{E}_j(r_k)$ is a random $(2^{3u+\log u}\tau^2, 2^{3u+\log u}\tau\rho)$-encryption under P_j's public key. P_i furthermore invokes $\Pi_{\text{PoCM}}(u, \tau, \rho)$ with input C_k, $\mathsf{E}_i(x_k)$, $\mathsf{E}_j(y_k)$, to prove that the C_k's are constructed correctly.
3. For each k, P_j decrypts C_k to obtain v_k, and outputs $z_{k,j} = v_k \bmod p$. P_i outputs $z_{k,i} = -r_k \bmod p$.

Fig. 9. Subprotocol allowing two parties to obtain encrypted sharings of the product of their inputs

Subprotocol $\Pi_{n\text{-MULT}}$

n**-Mult**(u)**:**

1. The input is $\langle a_k \rangle, \langle b_k \rangle, k = 1, \ldots, u$, created using the Π_{SHARE} protocol. Each P_i initializes variables $c_{k,i} = a_{k,i}b_{k,i} \bmod p, k = 1, \ldots, u$.
2. Each pair $P_i, P_j, i \neq j$, runs $\Pi_{2\text{-MULT}}$ using as input the ciphertexts $\mathsf{E}_i(a_{k,i}), \mathsf{E}_j(b_{k,j}), k = 1, \ldots, u$, and adds the outputs to the private variables $c_{k,i}, c_{k,j}$, i.e., for $k = 1, \ldots, u$, P_i sets $c_{k,i} = c_{k,i} + z_{k,i} \bmod p$, and P_j sets $c_{k,j} = c_{k,j} + z_{k,i} \bmod p$.
3. Each P_i invokes Π_{SHARE}, where $c_{k,i}, k = 1, \ldots, u$ is used as the numbers to broadcast encryptions of. Parties output what Π_{SHARE} outputs, namely $\langle c_k \rangle, k = 1, \ldots, u$.

Fig. 10. Protocol allowing the parties to construct $\langle c_k = a_k b_k \bmod p \rangle$ from $\langle a_k \rangle, \langle b_k \rangle$

Subprotocol Π_{ADDMACS}

Initialize: For each pair $P_i, P_j, i \neq j$, P_i chooses α_j^i at random in $\mathbb{Z}_p$, sends a (τ, ρ)-ciphertext $\mathsf{E}_i(\alpha_j^i)$ to P_j and then runs $\Pi_{\text{POPK}}(u, \tau, \rho)$ with $(\mathsf{E}_i(\alpha_j^i), \ldots, \mathsf{E}_i(\alpha_j^i))$ as input and with P_j as verifier.

AddMacs(u)**:**

1. The input is a set $\langle a_k \rangle, k = 1, \ldots, u$. Each P_i already holds shares $a_{k,i}$ of a_k, and will store these as part of $[a_k]$.
2. Each pair P_i, P_j $i \neq j$ invokes $\Pi_{2\text{-MULT}}(u, \tau, \rho)$ with input $\mathsf{E}_i(\alpha_j^i), \ldots, \mathsf{E}_i(\alpha_j^i)$ from P_i and input $\mathsf{E}_j(a_{k,j})$ from P_j. From this, P_i obtains output $z_{k,i}$, and P_j gets $z_{k,j}$. Recall that $\Pi_{2\text{-MULT}}$ ensures that $\alpha_j^i a_{k,j} = z_{k,i} + z_{k,j} \bmod p$. This is essentially the equation defining the MACs we need, so therefore, as a part of each $[a_k]$, P_i stores $\alpha_j^i, \beta_{a_{k,j}}^i = -z_{k,i} \bmod p$ as the MAC key to use against P_j while P_j stores $m_i(a_{k,j}) = z_{k,j}$ as the MAC to use to convince P_i about $a_{k,j}$.

Fig. 11. Subprotocol constructing $[a_k]$ from $\langle a_k \rangle$

Protocol Π_{TRIP}

Initialize: The parties first invoke $\mathcal{F}_{\text{KEYREG}}(p)$ and then Initialize in Π_{ADDMACS}.

Triples(u)**:**

1. To get sets of representations $\{\langle a_k \rangle, \langle b_k \rangle, \langle f_k \rangle, \langle g_k \rangle\}_{k=1}^u$, the parties invoke Π_{SHARE} 4 times.
2. The parties invoke $\Pi_{n\text{-MULT}}$ twice, on inputs $\{\langle a_k \rangle, \langle b_k \rangle\}_{k=1}^u$, respectively $\{\langle f_k \rangle, \langle g_k \rangle\}_{k=1}^u$. They obtain as output $\{\langle c_k \rangle\}_{k=1}^u$, respectively $\{\langle h_k \rangle\}_{k=1}^u$.
3. The parties invoke Π_{ADDMACS} on each of the created sets of the representations. That means they now have $\{[a_k], [b_k], [c_k], [f_k], [g_k], [h_k]\}_{k=1}^u$.
4. The parties check that indeed $a_k b_k = c_k \bmod p$ by "sacrificing" the triples (f_k, g_k, h_k): First, the parties invoke $\mathcal{F}_{\text{RAND}}$ to get a random u-bit challenge e. Then, they open $e[a_k] - [f_k]$ to get ε_k, and open $[b_k] - [g_k]$ to get δ_k. Next, they open $e[c_k] - [h_k] - \delta_k[f_k] - \varepsilon_k[g_k] - \varepsilon_k\delta_k$ and check that the result is 0. Finally, parties output the set $\{[a_k], [b_k], [c_k]\}_{k=1}^u$.

Singles(u)**:**

1. To get a set of representations $\{\langle a \rangle\}_{k=1}^u$, Π_{SHARE} is invoked.
2. The parties invoke Π_{ADDMACS} on the created set of representations and obtain $\{[a_k]\}_{k=1}^u$.

Fig. 12. The protocol for the offline phase

(τ, ρ)-encryptions $\mathsf{E}_j(y_k)$ under his public key. For each k, we want the protocol to output $z_{k,i}, z_{k,j}$ to P_i, P_j, respectively, such that $x_k y_k = z_{k,i} + z_{k,j} \bmod p$. Such a protocol can be seen in Figure 9. This protocol does not commit parties to their output, so there is no guarantee that corrupt parties will later use their output correctly – however, the protocol ensures that malicious parties *know* which shares they ought to continue with. To build the protocol $\Pi_{n\text{-MULT}}$, the

first thing to notice is that given $\langle a_k \rangle$ and $\langle b_k \rangle$ we have that $c_k = a_k b_k = \sum_i \sum_j a_{k,i} b_{k,j}$. Constructing each of the terms in this sum in shared form is exactly what $\Pi_{2\text{-MULT}}$ allows us to do. The $\Pi_{n\text{-MULT}}$ protocol can now be seen in Figure 10. Note that it does not guarantee that the multiplicative relation in the triples holds, we will check for this later.

4.3 From $\langle \cdot \rangle$-Triples to $[\cdot]$-Triples

We first describe a protocol that allows us to add MACs to the $\langle \cdot \rangle$-representation. This consists essentially of invoking the $\Pi_{2\text{-MULT}}$ a number of times. The protocol is shown in Figure 11. The full protocol Π_{TRIP}, which also includes the possibility of creating a set of single values, is now a straightforward application of the subprotocols we have defined now. This is shown in Figure 12. The proof of Theorem 2 can be found in the full paper [BDOZ10].

Theorem 2. *If the underlying cryptosystem is semi-homomorphic modulo p, admissible and IND-CPA secure, then* Π_{TRIP} *implements* $\mathcal{F}_{\text{TRIP}}$ *with computational security against any static, active adversary corrupting up to* $n-1$ *parties, in the* $(\mathcal{F}_{\text{KeyReg}}, \mathcal{F}_{\text{Rand}})$*-hybrid model.*

References

[BCNP04] Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: FOCS, pp. 186–195 (2004)

[BD10] Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Heidelberg (2010)

[BDOZ10] Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorhic enryption and multiparty computation (full version). In: The Eprint Archive, report 2010/514 (2010)

[Bea91] Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 420–432. Springer, Heidelberg (1992)

[BOGW88] Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)

[Can01] Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS, pp. 136–145 (2001)

[CCD88] Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)

[CD09] Cramer, R., Damgård, I.: On the amortized complexity of zero-knowledge protocols. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 177–191. Springer, Heidelberg (2009)

[CDN01] Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 280–299. Springer, Heidelberg (2001)

[CLOS02] Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: STOC, pp. 494–503 (2002)

[DGK09] Damgård, I., Geisler, M., Krøigaard, M.: A correction to efficient and secure comparison for on-line auctions. IJACT 1(4), 323–324 (2009)

[DGHV10] van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)

[DJ01] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

[DO10] Damgård, I., Orlandi, C.: Multiparty computation for dishonest majority: From passive to active security at low cost. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 558–576. Springer, Heidelberg (2010)

[Gen09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)

[GHV10] Gentry, C., Halevi, S., Vaikuntanathan, V.: A simple bgn-type cryptosystem from lwe. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 506–522. Springer, Heidelberg (2010)

[HIK07] Harnik, D., Ishai, Y., Kushilevitz, E.: How Many Oblivious Transfers Are Needed for Secure Multiparty Computation? In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 284–302. Springer, Heidelberg (2007)

[IPS09] Ishai, Y., Prabhakaran, M., Sahai, A.: Secure arithmetic computation with no honest majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 294–314. Springer, Heidelberg (2009)

[LPS10] Lyubashevsky, V., Palacio, A., Segev, G.: Public-key cryptographic primitives provably as secure as subset sum. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 382–400. Springer, Heidelberg (2010)

[OU98] Okamoto, T., Uchiyama, S.: A new public-key cryptosystem as secure as factoring. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 308–318. Springer, Heidelberg (1998)

[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)

[PSSW09] Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure two-party computation is practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)

[RAD78] Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation, pp. 169–178 (1978)

[Reg05] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC, pp. 84–93 (2005)

Tight Proofs for Signature Schemes without Random Oracles

Sven Schäge

Horst Görtz Institute for IT-Security
Ruhr-University of Bochum
sven.schaege@rub.de

Abstract. We present the first tight security proofs for two general classes of Strong RSA based signature schemes. Among the affected signature schemes are the Cramer-Shoup, Camenisch-Lysyanskaya, Zhu, and Fischlin signature scheme. We also present two bilinear variants of our signature classes that produce short signatures. Similar to before, we show that these variants have tight security proofs under the the Strong Diffie-Hellman (SDH) assumption. We so obtain very efficient SDH-based variants of the Cramer-Shoup, Fischlin, and Zhu signature scheme and the first tight security proof of the recent Camenisch-Lysyanskaya scheme that was proposed and proven secure under the SDH assumption. Central to our results is a new proof technique that allows the simulator to avoid guessing which of the attacker's signature queries are re-used in the forgery. In contrast to previous proofs, our security reduction does not lose a factor of q here.

Keywords: signature class, tight security, SRSA, SDH, standard model.

1 Introduction

Provable Security and Tight Reductions. The central idea of provable security is to design a cryptographic scheme in such a way that if an attacker $\mathcal{A}$ could efficiently break its security properties then one can also construct an efficient algorithm $\mathcal{B}$, to break a supposedly hard problem. In this way, we prove the security of the scheme *by reduction* from the hardness assumption. Now, if $\mathcal{B}$ has almost the same success probability as $\mathcal{A}$ while running in roughly the same time we say that the security reduction is tight. Otherwise, the security reduction is said to be loose. It is no secret why cryptographers are interested in tight security proofs: besides being theoretically interesting, they allow for shorter security parameters and better efficiency. This work was also motivated by the observation that for several of the existing Strong RSA (SRSA) based signature schemes without random oracles we do not know if tight security proofs exist. Those schemes which we know to have a tight security proof, also have some limitations concerning practicability (which in turn cannot be found among the signature schemes with a loose security reduction). In 2007, Chevallier-Mames

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 189–206, 2011.

and Joye addressed this problem in the following way [6]: they took a tightly secure signature scheme, the Gennaro-Halevi-Rabin scheme [10], and improved its efficiency by re-designing one of its most time-consuming functions. The problem with such an approach is that it only affects *new* implementations of the considered signature scheme. Therefore, we take the same approach as Bernstein at EUROCRYPT '08 who proved tight security for the *original* Rabin-Williams signature scheme in the random-oracle model [2]. However, in contrast to Bernstein we concentrate on schemes that are secure in the standard model.

Contribution. In this work, we ask the following question: are there tight security proofs for the existing practical signature schemes by Cramer-Shoup [8], Zhu [19], Camenisch-Lysyanskaya [4] and Fischlin [9] (which we only know to have loose security reductions)? We answer this question in the affirmative and present the first tight proofs for the above signature schemes. However, our result is not limited to the original schemes. In our analysis, we generalize the schemes by Camenisch-Lysyanskaya, Fischlin and Zhu by introducing a new family of randomization functions, called combining functions. The result of this generalization is an abstract signature scheme termed 'combining scheme'. In a similar way, we introduce a second general class of signature schemes called 'chameleon hash scheme' that can be regarded as a generalization of the Cramer-Shoup signature scheme. Then, we prove the combining signature scheme and the chameleon hash scheme to be *tightly* secure under the SRSA assumption when instantiated with any secure combining function, respectively chameleon hash function. Finally, we show that our results do not only hold under the SRSA assumption. We analyze whether there also exist tight security reductions for analogous schemes based on the SDH assumption in bilinear groups. Interestingly, most of the above schemes have not been considered yet under the SDH assumption (except for the Camenisch-Lysyanskaya scheme), although, at the same security level, the group description is much shorter in bilinear groups than in factoring based groups. We develop an SDH-based variant of the combining signature scheme and the chameleon hash scheme and prove it to be existentially unforgeable under adaptive chosen message attacks with a *tight* security reduction. In doing so, we present the first SDH-based variants of the Fischlin, the Zhu and the Cramer-Shoup signature scheme and the first tight security proof of the SDH-based Camenisch-Lysyanskaya scheme. When instantiated with existing combining functions (respectively chameleon hash functions), we obtain short and efficient signature schemes. All our results (on the combining class) can easily be extended to signature schemes for message blocks (as defined in [4,5]), where we can even use distinct combining functions for each message block. Our results can be interpreted in two *positive* ways: 1) Existing implementations of the affected signature schemes (with a fixed parameter size) provide higher security than expected. 2) New implementations can have shorter security parameters what transfers to higher efficiency.

Technical Contribution. In the existing proofs, the simulator partitions the set of forgeries by at first guessing $j \in \{1, \ldots, q\}$ where q is the number of

signature queries made by the attacker. Only if the attacker's forgery shares some common values with the answer to the j-th signature query the simulator can break the SRSA assumption. Otherwise the simulator just aborts. The number of signature queries rises polynomially in the security parameter and the security proof loses a factor of q here. Our main contribution is a new technique that renders the initial guess unnecessary. As a consequence, *any* forgery helps the simulator to break the SRSA assumption. This results in a tight security proof.

Related Work. Our work is related to the existing hash-and-sign signature schemes without random oracles that are proven secure under the SRSA or the SDH assumption. We subsequently give a brief overview on the available results. In 1988, Goldwasser, Micali and Rivest published the first provably secure, but inefficient signature scheme [11]. More than a decade later, in 1999, Gennaro, Halevi, and Rabin [10] presented a signature scheme that is secure in the standard model under the Flexible or Strong RSA assumption (SRSA). This scheme is more efficient, both the key and the signature size are less than two group elements (à 1024 bits), but as a drawback, it relies on an impractical function that injectively maps messages to primes [7]. Advantageously, the Gennaro-Halevi-Rabin signature scheme is known to have a tight security proof. At the same time and also based on the SRSA assumption, Cramer and Shoup [8] proposed an efficient standard model signature scheme, that unlike [10] does not require to map messages to primes. In contrast, primes can be drawn uniformly at random from the set of primes of a given bitlength. Based on this work, Zhu [18,19], Fischlin [9], Camenisch and Lysyanskaya [4], and Hofheinz and Kiltz [12] in the following years presented further SRSA-based schemes. These schemes are either more efficient than the Cramer-Shoup scheme or very suitable in protocols for issuing signatures on committed values. In 2004, Boneh and Boyen presented the first hash-and-sign signature scheme that makes use of bilinear groups [3]. The big advantage of bilinear groups is the very compact representation of group elements. The Boneh-Boyen signature scheme is proven tightly secure under a new flexible assumption, the q-Strong Diffie Hellman (SDH) assumption. In 2004, Camenisch and Lysyanskaya also presented a signature scheme that relies on bilinear groups [5]. Unlike the Boneh-Boyen scheme, their scheme is proven secure under the LRSW [14] assumption. However, in the same paper they also propose a variant that is based on the SDH assumption in bilinear groups. The corresponding security proof was provided four years later in [15,1]. Similar to the original Camenisch-Lysyanskaya scheme the security proof of the SDH scheme is loose.

2 Preliminaries

Before presenting our results we briefly review the necessary formal and mathematical definitions. For convenience, we also describe two general setup and key generation procedures (settings) in Section 2.7 and Section 2.8. When describing our signature schemes in Sections 3.1, 3.2, 3.5 we will refer to the corresponding setting and only describe the signature generation and verification algorithms.

2.1 Notation

For $a, b \in \mathbb{Z},\ a \leq b$ we write $[a; b]$ to denote the set $\{a, a+1, \ldots, b-1, b\}$. For a string x, we write $|x|_2$ to denote its bit length. If $z \in \mathbb{Z}$, we write $|z|$ to denote the absolute value of z. For a set X, we use $|X|$ to refer to its size and $x \xleftarrow{\$} X$ to indicate that x is drawn from X uniformly at random. For $n \in \mathbb{N}$, we use QR_n to denote the set of quadratic residues modulo n, i.e. $QR_n = \{x | \exists y \in \mathbb{Z}_n^* : y^2 = x \bmod n\}$. If $\mathcal{A}$ is an algorithm we use $\mathcal{A}(x_1, x_2, \ldots)$ to denote that $\mathcal{A}$ has input parameters $x_1, x_2, \ldots$. Accordingly, $y \leftarrow \mathcal{A}(x_1, x_2, \ldots)$ means that $\mathcal{A}$ outputs y when running with inputs $x_1, x_2, \ldots$. PPT refers to probabilistic polynomial-time. We write $\kappa \in \mathbb{N}$ to indicate the security parameter and 1^κ to describe the string that consist of κ ones. In the following, we implicitly assume that the size of the generated key material is always polynomially dependent on the security parameter.

2.2 Signature Scheme

A digital signature scheme $\mathcal{S}$ consists of three algorithms. The PPT algorithm KeyGen on input 1^κ generates a secret and public key pair (SK, PK). The PPT algorithm Sign takes as input a secret key SK and the message m and outputs a signature σ. Finally, the deterministic polynomial time algorithm Verify takes a public key PK, a message m and a signature σ to check whether σ is a legitimate signature on m signed by the holder of the secret key corresponding to PK. Accordingly, the algorithm outputs 1 to indicate a successful verification and 0 otherwise.

2.3 Strong Existential Unforgeability

The standard notion of security for signature schemes is due to Goldwasser, Micali and Rivest [11]. In this paper, we use a slightly stronger definition called strong existential unforgeability. The signature scheme $\mathcal{S} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is strongly existentially unforgeable under an adaptive chosen message attack if it is infeasible for a forger, who only knows the public key and the global parameters, to produce, after obtaining polynomially (in the security parameter) many signatures $\sigma_1, \ldots, \sigma_q$ on messages $m_1, \ldots, m_q$ of its choice from a signing oracle $\mathcal{O}(SK, \cdot)$, a new message/signature pair.

Definition 1. *We say that $\mathcal{S}$ is (q, t, ϵ)-secure, if for all t-time adversaries $\mathcal{A}$ that send at most q queries to the signing oracle $\mathcal{O}(SK, \cdot)$ it holds that*

$$\Pr\left[\begin{array}{c}(SK, PK) \leftarrow \mathsf{KeyGen}(1^\kappa),\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(SK,\cdot)}(PK), \\ \mathsf{Verify}(PK, m^*, \sigma^*) = 1\end{array}\right] \leq \epsilon,$$

where the probability is taken over the random coins of KeyGen and $\mathcal{A}$ and (m^*, σ^*) is not among the message/signature pairs obtained using $\mathcal{O}(SK, \cdot)$ (i.e. $(m^*, \sigma^*) \notin \{(m_1, \sigma_1), \ldots, (m_q, \sigma_q)\}$).

2.4 Collision-Resistant Hashing

Definition 2 (Collision-resistant hash function). *Let $\mathcal{H}_k$ for $k \in \mathbb{N}$ be a collection of functions of the form $h : \{0,1\}^* \to \{0,1\}^k$. Let $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$. $\mathcal{H}$ is called (t_h, ϵ_h)-collision-resistant if for all t_h-time adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[h \xleftarrow{\$} \mathcal{H}_k,\ (m,m') \leftarrow \mathcal{A}(h),\ m \neq m',\ m,m' \in \{0,1\}^*,\ h(m) = h(m')\right] \leq \epsilon_h,$$

where the probability is over the random bits of $\mathcal{A}$.

2.5 Chameleon Hash Function

A chameleon hash function $\mathcal{CH} = (\mathsf{CHGen}, \mathsf{CHEval}, \mathsf{CHColl})$ consists of three algorithms [13]. The PPT algorithm CHGen takes as input the security parameter κ and outputs a secret key $SK_{\mathcal{CH}}$ and a public key $PK_{\mathcal{CH}}$. Given $PK_{\mathcal{CH}}$, a random r from a randomization space $\mathcal{R}$ and a message m from a message space $\mathcal{M}$, the algorithm CHEval outputs a chameleon hash value c in the hash space $\mathcal{C}$. Analogously, CHColl deterministically outputs, on input $SK_{\mathcal{CH}}$ and $(r, m, m') \in \mathcal{R} \times \mathcal{M} \times \mathcal{M}$, $r' \in \mathcal{R}$ such that $\mathsf{CHEval}(PK_{\mathcal{CH}}, m, r) = \mathsf{CHEval}(PK_{\mathcal{CH}}, m', r')$.

Definition 3 (Collision-resistant chameleon hash function). *We say that $\mathcal{CH}$ is $(\epsilon_{\mathcal{CH}}, t_{\mathcal{CH}})$-collision-resistant if for all $t_{\mathcal{CH}}$-time adversaries $\mathcal{A}$ that are only given $PK_{\mathcal{CH}}$ it holds that*

$$\Pr\left[\begin{array}{c}(SK_{\mathcal{CH}}, PK_{\mathcal{CH}}) \leftarrow \mathsf{CHGen}(1^\kappa),\ (m,m',r,r') \leftarrow \mathcal{A}(PK_{\mathcal{CH}}),\\ r,r' \in \mathcal{R},\ m,m' \in \mathcal{M},\ m' \neq m,\\ \mathsf{CHEval}(PK_{\mathcal{CH}}, r, m) = \mathsf{CHEval}(PK_{\mathcal{CH}}, r', m')\end{array}\right] \leq \epsilon_{\mathcal{CH}},$$

where the probability is over the random choices of $PK_{\mathcal{CH}}$ and the coin tosses of $\mathcal{A}$.

We also require that for an arbitrary but fixed public key $PK_{\mathcal{CH}}$ output by CHGen, all messages $m \in \mathcal{M}$ generate equally distributed hash values when drawing $r \in \mathcal{R}$ uniformly at random and outputting $\mathsf{CHEval}(PK_{\mathcal{CH}}, r, m)$. We write $ch(r, m)$ to denote $\mathsf{CHEval}(PK_{\mathcal{CH}}, r, m)$ and $ch^{-1}(r, m, m')$ for $\mathsf{CHColl}(SK_{\mathcal{CH}}, r, m, m')$ if the keys are obvious from the context. The security of chameleon hash functions can be based on very standard assumptions like the discrete logarithm assumption [13] or the factoring assumption [13,17] which are weaker than the SDH, respectively the SRSA assumption.

2.6 Combining Function

In this section, we introduce a new family of functions called combining functions. We will subsequently use the concept of combining functions to generalize several existing signature schemes.

Definition 4 (Combining Functions). *Let $\mathcal{V}_k$ for $k \in \mathbb{N}$ be a collection of functions of the form $z : \mathcal{R} \times \mathcal{M} \to \mathcal{Z}$ with $|\mathcal{Z}| \leq 2^k$. Let $\mathcal{V} = \{\mathcal{V}_k\}_{k\in\mathbb{N}}$. We say that $\mathcal{V}$ is $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining if for all attackers $\mathcal{A}$ there exist negligible functions $\epsilon_{comb}(k)$ and $\delta_{comb}(k)$ and the following properties hold for $z \xleftarrow{\$} \mathcal{V}_k$.*

1. *for all $m \in \mathcal{M}$ it holds that $|\mathcal{R}| = |\mathcal{Z}_m|$ where $\mathcal{Z}_m$ is defined as $\mathcal{Z}_m = z(\mathcal{R}, m)$. For all $m \in \mathcal{M}$ and all $t \in \mathcal{Z}$ there exists an efficient algorithm $z^{-1}(t, m)$ that, if $t \in \mathcal{Z}_m$, outputs the unique value $r \in \mathcal{R}$ such that $z(r, m) = t$, and $\perp$ otherwise.*
2. *for $t \xleftarrow{\$} \mathcal{Z}$ and $r' \xleftarrow{\$} \mathcal{R}$ we have for the maximal (over all $m \in \mathcal{M}$) statistical distance between r' and $z^{-1}(t, m)$ that*

$$\max_{m\in\mathcal{M}} \left\{ 1/2 \cdot \sum_{r\in\mathcal{R}} \left| \Pr[r' = r] - \Pr[z^{-1}(t, m) = r] \right| \right\} \leq \delta_{comb}.$$

3. *for all $r \in \mathcal{R}$, it holds for all t_{comb}-time attackers $\mathcal{A}$ that*

$$\Pr\left[\begin{array}{c} (m, m') \leftarrow \mathcal{A}(z, r),\ m, m' \in \mathcal{M}, \\ m \neq m',\ z(r, m) = z(r, m') \end{array} \right] \leq \epsilon_{comb},$$

where the probability is taken over the random bits of $\mathcal{A}$.

In the following, we assume that when used in signature schemes, $z \xleftarrow{\$} \mathcal{V}_k$ is chosen uniformly at random during the key generation phase.

Table 1. Examples of statistically secure combining functions. Let $\mathcal{V} = \{\mathcal{V}_k\}_{k\in\mathbb{N}}$ with $\mathcal{V}_k = \{z(r, m)\}$, $l, l_r, l_m \in \mathbb{N}$, $l_r > l_m$ and p be prime.

	$z(r, m)$	$\mathcal{R}$	$\mathcal{M}$	$\mathcal{Z}$	combining
EX1	$r + m \bmod p$	$\mathbb{Z}_p$	$\mathbb{Z}_p$	$\mathbb{Z}_p$	$(\cdot, 0, 0)$
EX2	$r \oplus m$	$\{0,1\}^l$	$\{0,1\}^l$	$\{0,1\}^l$	$(\cdot, 0, 0)$
EX3	$r + m$	$[0; 2^{l_r} - 1]$	$[0; 2^{l_m} - 1]$	$[0; 2^{l_r} + 2^{l_m} - 2]$	$(\cdot, 0, 2^{l_m - l_r})$

In Table 1, we present three concrete examples (EX1, EX2, EX3) of statistically secure combining functions. The following lemma shows that these examples are valid combining functions with respect to Definition 4.

Lemma 1. EX1 *and* EX2 *constitute* $(\cdot, 0, 0)$*-combining functions and* EX3 *constitutes a* $(\cdot, 0, 2^{l_m - l_r})$*-combining function.*

Proof. Let us first analyze EX1 and EX2. We have that $\mathcal{M} = \mathcal{R} = \mathcal{Z} = \mathcal{Z}_m$ for all $m \in \mathcal{M}$ and we can efficiently compute r as $r = t - m \bmod p$ or $r = t \oplus m$ for *all* given $t \in \mathcal{Z}$ and $m \in \mathcal{M}$. Furthermore, since z is bijective in both input parameters $z^{-1}(t, m)$ is uniformly distributed in $\mathcal{R}$ for all $m \in \mathcal{M}$ and random $t \in \mathcal{Z}$. Thus, $\delta_{\text{comb}} = 0$. Finally, since z is a bijection in the second input

parameter, it is collision-free (property 3) in both examples and we have that $\epsilon_{\text{comb}} = 0$. Now, let us analyze EX3. For given $m \in \mathcal{M}$ and $t \in \mathcal{Z}$, $z^{-1}(t, m)$ outputs $r = t - m$ if $t - m \in \mathcal{R}$ and $\perp$ otherwise. To show that z is collision-free, observe that $m \neq m'$ implies $r + m \neq r + m'$ for all $r \in \mathcal{R}$. To analyze $D = \max_{m \in \mathcal{M}} \left\{1/2 \cdot \sum_{r \in \mathcal{R}} \left|\Pr[r' = r] - \Pr[z^{-1}(t, m) = r]\right|\right\}$ first note that for $t' \xleftarrow{\$} \mathcal{Z}_m$, $z^{-1}(t', m)$ is uniform in $\mathcal{R}$ since $|\mathcal{Z}_m| = |\mathcal{R}|$ implies that $z^{-1}(\cdot, m)$ defines a bijection from $\mathcal{Z}_m$ to $\mathcal{R}$. For $t' \xleftarrow{\$} \mathcal{Z}_m$ and $t \xleftarrow{\$} \mathcal{Z}$ we get

$$D \leq \max_{m \in \mathcal{M}} \left\{ 1/2 \cdot \sum_{t_0 \in \mathcal{Z}_m} |\Pr[t' = t_0] - \Pr[t = t_0]| \right\} \leq 2^{l_m - l_r}$$

Three further examples of combining functions can be obtained when first applying a (t_h, ϵ_h)-collision-resistant hash function that maps (long) messages to $\mathcal{M}$. Lemma 2 guarantees that the results are still combining according to Definition 4. The proof of Lemma 2 is straight-forward and can be found in the full version.

Lemma 2. *Let $\mathcal{V}$ be a $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining function and $\mathcal{H}$ be a (t_h, ϵ_h)-collision-resistant hash function. Then $\mathcal{V}' = \{\mathcal{V}'_k\}_{k \in \mathbb{N}}$ with $\mathcal{V}'_k = \{z(r, h(m)) | z \xleftarrow{\$} \mathcal{V}_k, h \xleftarrow{\$} \mathcal{H}_k\}$ is $(\min\{t_{comb}, t_h\}, \epsilon_{comb} + \epsilon_h, \delta_{comb})$-combining.*

2.7 The Strong RSA Setting

Definition 5 (Strong RSA assumption (SRSA)). *Given an RSA modulus $n = pq$, where p, q are sufficiently large primes, and an element $u \in \mathbb{Z}_n^*$, we say that the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds if for all t_{SRSA}-time adversaries $\mathcal{A}$*

$$\Pr\left[(x, y) \leftarrow \mathcal{A}(n, u),\ x \in \mathbb{Z}_n^*,\ y > 1,\ x^y = u \bmod n\right] \leq \epsilon_{SRSA},$$

where the probability is over the random choices of u, n and the random coins of $\mathcal{A}$.

Definition 6 (SRSA setting). *In this setting, $\mathsf{KeyGen}(1^\kappa)$ outputs the key pair $(SK = (p, q), PK = n)$ for a safe modulus $n = pq$ such that $p = 2p' + 1$, $q = 2q' + 1$, and p, q, p', q' are primes. All computations are performed in the cyclic group QR_n. Let $l_i = l_i(\kappa)$ for $i \in \{n, t, c, e, m\}$ be polynomials. We require that $|n|_2 = l_n$ and $|p'|_2 = |q'|_2 = l_n/2 - 1$. Furthermore, we assume that the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds. We let u, v, w be public random generators of QR_n with unknown $\log_u v$, $\log_u w$, and $\log_v w$. When using combining functions $z(r, m)$, we assume that $\mathcal{M} \subseteq [0; 2^{l_m} - 1]$, $\mathcal{Z} \subseteq [0; 2^{l_z} - 1]$ and $\mathcal{R} \subseteq [0; 2^{l_r} - 1]$. We let $E \subseteq [2^{l_e - 1}; 2^{l_e} - 1]$ denote the set of l_e-bit primes. Finally, we require that $l_m \leq l_c, l_z, l_r < l_e < l_n/2 - 1$.*

2.8 The Strong Diffie-Hellman Setting

Definition 7 (Bilinear groups). *Let* $\mathbb{G}_1 = <g_1>, \mathbb{G}_2 = <g_2>$ *and* $\mathbb{G}_T$ *be groups of prime order* p*. The function* $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ *is a bilinear pairing if it holds that 1) for all* $a \in \mathbb{G}_1$*,* $b \in \mathbb{G}_2$*, and* $x, y \in \mathbb{Z}_p$ *we have* $e(a^x, b^y) = e(a, b)^{xy}$ *(bilinearity), 2)* $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$ *is a generator of* $\mathbb{G}_T$ *(non-degeneracy), and 3)* e *is efficiently computable (efficiency). We call* $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ *a bilinear group.*

Definition 8 (SDH assumption (SDH)). *Let* $(\mathbb{G}_1, \hat{g}_1, \mathbb{G}_2, \hat{g}_2, \mathbb{G}_T, p, e)$ *be a bilinear group. We say that the* $(q_{SDH}, t_{SDH}, \epsilon_{SDH})$*-SDH assumption holds if for all* t_{SDH}*-time attackers* $\mathcal{A}$ *that are given a* $(q_{SDH} + 3)$*-tuple of elements* $W = \left(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^{q_{SDH}})}, g_2, g_2^x\right) \in \mathbb{G}_1^{q_{SDH}+1} \times \mathbb{G}_2^2$ *it holds that*

$$\Pr\left[(s, c) \leftarrow \mathcal{A}(W),\ c \in \mathbb{Z}_p,\ s \in \mathbb{G}_1,\ e(s, g_2^x g_2^c) = e(g_1, g_2)\right] \leq \epsilon_{SDH},$$

where the probability is over the random choices of the generators $g_1 \in \mathbb{G}_1$*,* $g_2 \in \mathbb{G}_2$*,* $x \in \mathbb{Z}_p$ *and the random bits of* $\mathcal{A}$*.*

Definition 9 (SDH setting). *In the SDH setting, all computations are performed in the cyclic groups of* $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$ *such that* $|p|_2 = l_p = l_p(\kappa)$*. The PPT* $\mathsf{KeyGen}(1^\kappa)$ *chooses* $x \xleftarrow{\$} \mathbb{Z}_p$ *and outputs* $(SK = x, PK = g_2^x)$*. We assume that the* $(q_{SDH}, t_{SDH}, \epsilon_{SDH})$*-SDH assumption holds. Finally, we suppose that the values* $a, b, c \in \mathbb{G}_1$ *are public random generators of* $\mathbb{G}_1$ *such that* $\log_a b$*,* $\log_a c$*, and* $\log_b c$ *are unknown. In case of combining functions* $z(r, m)$*, we assume that* $\mathcal{Z} \subseteq \mathbb{Z}_p$ *and* $\mathcal{R} \subseteq \mathbb{Z}_p$*.*

3 Signature Classes

For convenience, we now introduce two general signature classes. The combining signature scheme $\mathcal{S}_{\text{CMB}}$ constitutes a useful abstraction of the Camenisch-Lysyanskaya, the Fischlin, and the Zhu signature scheme using combining functions. The chameleon signature scheme $\mathcal{S}_{\text{CH}}$ can be regarded as a general variant of the original Cramer-Shoup signature scheme where we do not specify a concrete instantiation of the chameleon hash function.

3.1 SRSA-Based Combining Signature Scheme $\mathcal{S}_{\text{CMB,SRSA}}$

In the SRSA setting, $\mathsf{Sign}(SK, m)$ randomly draws $r \in \mathcal{R}$ and $e \in E$ and computes a signature $\sigma = (r, s, e)$ on message m with $s = \left(uv^r w^{z(r,m)}\right)^{\frac{1}{e}}$. Let us now show that our construction generalizes the claimed signature schemes. Observe that we can easily obtain the Fischlin scheme [9] if we instantiate the combining function with EX2 of Table 1. Furthermore, we can also get the Camenisch-Lysyanskaya scheme [4] using EX3. This becomes obvious if we substitute v by

$v' = vw$ as $uv^r w^{r+m} = u(vw)^r w^m = u(v')^r w^m$[1]. We note that when we use the Camenisch-Lysyanskaya scheme with long messages we must first apply a collision-resistant hash function to the message. What we essentially get is Zhu's scheme [18,19]. By Lemma 2, the resulting function is still combining. The verification routine $\mathsf{Verify}(PK, m, \sigma)$ takes a purported signature $\sigma = (r, s, e)$ and checks if $s^e \stackrel{?}{=} uv^r w^{z(r,m)}$, if $|e|_2 = l_e$, and if e is odd.

3.2 SDH-Based Combining Signature Scheme $\mathcal{S}_{\mathrm{CMB,SDH}}$

We also present an SDH-based variant $\mathcal{S}_{\mathrm{CMB,SDH}}$ of the combining signature scheme. We remark that for the Camenisch-Lysyanskaya scheme there already exists a corresponding SDH-based variant, originally introduced in [5] and proven secure in [15,1]. Similar to $\mathcal{S}_{\mathrm{CMB,SRSA}}$, we obtain the SDH-based Camenisch-Lysyanskaya scheme when instantiating the combining function with EX1. In the same way, we can also get SDH-based variants of the Fischlin signature scheme (using EX2) and of Zhu's scheme (using Lemma 2). In the SDH-based combining scheme, $\mathsf{Sign}(SK, m)$ at first chooses a random $r \in \mathcal{R}$ and a random $t \in \mathbb{Z}_p \setminus \{-x\}$. It then computes the signature $\sigma = (r, s, t)$ with $s = \left(ab^r c^{z(r,m)}\right)^{\frac{1}{x+t}}$. Given a signature $\sigma = (r, s, t)$, $\mathsf{Verify}(PK, m, \sigma)$ checks if $e\left(s, PK g_2^t\right) \stackrel{?}{=} e\left(ab^r c^{z(r,m)}, g_2\right)$.

3.3 SRSA-Based Chameleon Hash Signature Scheme $\mathcal{S}_{\mathrm{CH,SRSA}}$

The scheme $\mathcal{S}_{\mathrm{CH,SRSA}}$ is defined in the SRSA setting. $\mathsf{KeyGen}(1^\kappa)$ additionally generates the key material $(SK_{\mathcal{CH}}, PK_{\mathcal{CH}})$ for a chameleon hash function. The value $PK_{\mathcal{CH}}$ is added to the scheme's public key. ($SK_{\mathcal{CH}}$ is not required. However, it may be useful when turning the signature scheme into an online-offline signature scheme [17].) The signature generation algorithm $\mathsf{Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}$ and a random prime $e \in E$. It then outputs the signature $\sigma = (r, s, e)$ on a message m where $s = \left(uv^{ch(r,m)}\right)^{\frac{1}{e}}$. To verify a purported signature $\sigma = (r, s, e)$ on m, $\mathsf{Verify}(PK, m, \sigma)$ checks if e is odd, if $|e|_2 = l_e$, and if $s^e \stackrel{?}{=} uv^{ch(r,m)}$.

3.4 SDH-Based Chameleon Hash Signature Scheme $\mathcal{S}_{\mathrm{CH,SDH}}$

Let us now define a new variant of the chameleon hash signature scheme that is based on the SDH assumption. Again, $\mathsf{KeyGen}(1^\kappa)$ also adds the public key

[1] To be precise, our generalization slightly differs from the Camenisch-Lysyanskaya scheme. In the original scheme, it is required that $l_r = l_n + l_m + 160$. As a result, the authors recommend for 160 bit long messages that $l_r = 1346$, $l_s = 1024$, and $l_e = 162$. In our scheme, we simply require that $l_m \leq l_r < l_e < l_n/2 - 1$. Then, we can set $l_r = 320$, $l_s = 1024$, and $l_e = 321$ for a probability $\epsilon_{\mathrm{comb}} = 2^{-160}$. Therefore, the signature size of our signature scheme is much shorter (only $(320 + 1024 + 321)/(1346 + 1024 + 162) \approx 66\%$ of the original signature size) and the scheme is more efficient (since shorter exponents imply faster exponentations) than the original scheme.

$PK_{\mathcal{CH}}$ of a chameleon hash function to PK. In the SDH setting, $\mathsf{Sign}(SK, m)$ first chooses a random $r \in \mathcal{R}$ and a random $t \in \mathbb{Z}_p \setminus \{-x\}$. Using $SK = x$, it then outputs the signature σ on m as $\sigma = (r, s, t)$ where $s = \left(ab^{ch(r,m)}\right)^{\frac{1}{x+t}}$. To verify a given signature $\sigma = (r, s, t)$ on m, $\mathsf{Verify}(PK, m, \sigma)$ checks if $e\left(s, PKg_2^t\right) \stackrel{?}{=} e\left(ab^{ch(r,m)}, g_2\right)$. A suitable chameleon hash function can for example be found in [13].

3.5 The Cramer-Shoup Signature Scheme $\mathcal{S}_{\mathbf{CS,SRSA}}$

Let us now review the Cramer-Shoup signature scheme that is defined in the SRSA setting. The Cramer-Shoup scheme $\mathcal{S}_{\mathrm{CS,SRSA}}$ additionally requires a collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^{l_c}$. The message space is so extended to $\mathcal{M} = \{0,1\}^*$. Suppose $l_c < l_e < l_n/2 - 1$.

- $\mathsf{KeyGen}(1^\kappa)$ additionally computes a random l_e-bit prime $\tilde{e}$. The secret key is $SK = (p, q)$ the public key is $PK = (n, \tilde{e})$.
- $\mathsf{Sign}(SK, m)$ first chooses a random $r \in QR_n$ and evaluates (the chameleon hash function) $c = r^{\tilde{e}}/v^{h(m)} \mod n$. Then it draws a random l_e-bit prime $e \neq \tilde{e}$ and computes the value $s = \left(uv^{h(c)}\right)^{1/e} \mod n$. The signature is $\sigma = (r, s, e)$.
- $\mathsf{Verify}(PK, m, \sigma)$ re-computes $c = r^{\tilde{e}}/v^{h(m)} \mod n$ and checks whether $s \stackrel{?}{=} \left(uv^{h(c)}\right)^{1/e} \mod n$, if e is odd, and if $|e|_2 = l_e$.

Unfortunately, the proof of the more general chameleon hash scheme class does not formally transfer to the Cramer-Shoup signature scheme because in the Cramer-Shoup scheme the key material of its chameleon hash function is not chosen independently. In particular, the chameleon hash function uses the same RSA modulus and the same value v. This requires slightly more care in the security proof. We provide a full proof of the Cramer-Shoup signature scheme in the full version.

4 Security

Theorem 1. *The Cramer-Shoup signature scheme, the combining signature class (in both the SRSA and the SDH setting), and the chameleon signature class (in both the SRSA and the SDH setting) are tightly secure against adaptive chosen message attacks. In particular, this implies that the Camenisch-Lysyanskaya, the Fischlin, the Zhu, and the SDH-based Camenisch-Lysyanskaya scheme are tightly secure against strong existential forgeries under adaptive chosen message attacks.*

We subsequently provide the intuition behind our security proofs. In Section 4.4, we present a *full* proof of security for $\mathcal{S}_{\mathrm{CMB,SRSA}}$, which seems to us to be the technically most involved reduction. The proof of $\mathcal{S}_{\mathrm{CMB,SDH}}$ proceeds analogously and appears in the full version. We then informally show how to transfer our technique to $\mathcal{S}_{\mathrm{CH}}$. In the full version we also provide a full proof of security of the Cramer-Shoup signature scheme.

4.1 The SRSA-Based Schemes

Let us first consider the SRSA-based schemes, where $\mathcal{B}$ is given an SRSA challenge $(\hat{u}, n)$ with $\hat{u} \in \mathbb{Z}_n^*$. Assume that attacker $\mathcal{A}$ issues q signature queries $m_1, \ldots, m_q \in \mathcal{M}$. As a response to each query m_i with $i \in [1;q]$, $\mathcal{A}$ receives a corresponding signature $\sigma_i = (r_i, s_i, e_i) \in \mathcal{R} \times QR_n \times E$.

Recall that the existing security proofs for schemes of the combining class (e.g. [9]) consider two forgers that loosely reduce from the SRSA assumption. This is the case when it holds for $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$ that $\gcd(e^*, \prod_{i=1}^{q} e_i) \neq 1$[2]. Given that $|e^*|_2 = l_e$ this means that $e^* = e_j$ for some $j \in [1;q]$. Let us concentrate on the case that $r^* \neq r_j$. The proof of the remaining case ($e^* = e_j$, $r^* = r_j$ and $m^* \neq m_j$) is very similar. It additionally exploits the properties of the combining function.

The proofs in [8,9,18,4,19] work as follows: the simulator $\mathcal{B}$ at first guesses $j \xleftarrow{\$} \{1, \ldots, q\}$. By construction, $\mathcal{B}$ can answer all signature queries but only if $\mathcal{A}$ outputs a forgery where $e^* = e_j$ it can extract a solution to the SRSA challenge. In all other cases (if $e^* = e_i$ for some $i \in \{1, \ldots, q\} \setminus \{j\}$), $\mathcal{B}$ just aborts. Since the number of signature queries q rises polynomially in the security parameter, the probability for $\mathcal{B}$ to correctly guess j in advance is q^{-1} and thus not negligible. However, the security reduction loses a factor of q here.

Our aim is to improve this reduction step. Ideally, we have that *any* forgery which contains $e^* \in \{e_1, \ldots, e_q\}$ helps the simulator to break the SRSA assumption. As a result, the simulator can completely avoid guessing. The main task is to re-design the way $\mathcal{B}$ computes $\mathcal{A}$'s input parameters: for *every* $i \in \{1, \ldots, q\}$, we must have exactly one choice of r_i such that $\mathcal{B}$ can simulate the signing oracle without having to break the SRSA assumption. On the other hand, if $\mathcal{A}$ outputs $(m^*, (r^*, s^*, e^*))$ with $e^* = e_i$ for some $i \in [1;q]$ and $r^* \neq r_i$, $\mathcal{B}$ must be able to compute a solution to the SRSA challenge. Let us now go into more detail.

For simplicity, assume that $\mathcal{B}$ can setup $\mathcal{A}$'s input parameters such that the verification of a signature $\sigma = (r, s, e)$ always reduces to

$$s^e = \hat{u}^{f(r)} \bmod n. \tag{1}$$

Suppose that neither $\hat{u}$ nor $f : \mathcal{R} \to \mathbb{N}$ are ever revealed to $\mathcal{A}$. We exploit that the r_i are chosen independently at random. So, they can be specified *prior* to the signature queries. Now, $\mathcal{B}$'s strategy to simulate the signing oracle is to define $r_1, \ldots, r_q$ such that for every $i \in [1;q]$ it can compute a prime $e_i \in E$ with $e_i | f(r_i)$. Without having to break the SRSA assumption, $\mathcal{B}$ can then compute $s_i = \hat{u}^{f(r_i)/e_i}$ and output the i-th signature as (r_i, s_i, e_i).

Let us now turn our attention to the extraction phase where $\mathcal{B}$ is given $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption we have $e^* = e_i$ for some $i \in [1;q]$ and $r^* \neq r_i$. $\mathcal{B}$ wants to have that $\gcd(e^*, f(r^*)) = D < e^*$ (or $f(r^*) \neq 0 \bmod e^*$) because then it can find a solution to the SRSA challenge by computing $a, b \in \mathbb{Z} \setminus \{0\}$ with $af(r^*)/D + be^*/D = 1$ using extended Euclidean algorithm and outputting

[2] The proof of the case $\gcd(e^*, \prod_{i=1}^{q} e_i) = 1$ is straight-forward.

$$(s^*)^a \hat{u}^b = \hat{u}^{D/e^*}, e^*/D.$$

$\mathcal{B}$'s strategy to guarantee $\gcd(e^*, f(r^*)) = D < e^*$ is to ensure that we have $e^* = e_i \not| f(r^*)$. Unfortunately, $\mathcal{B}$ cannot foresee r^*. Therefore, the best solution is to design f such that $e_i \not| f(r^*)$ for *all* $r^* \neq r_i$.

Obviously, $\mathcal{B}$ makes strong demands on f. We now present our construction of f and argue that it perfectly fulfills all requirements. We define f as

$$f(r) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j - r \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j \neq i}}^{q} e_j, \tag{2}$$

for $r_1, \ldots, r_q \in \mathcal{R}$. Furthermore, $e_1, \ldots, e_q \in E$ must be distinct primes. First, observe that for every $k \in [1; q]$ the function reduces to $f(r_k) = \sum_{i=1, i \neq k}^{q} (r_i - r_k) \prod_{j=1, j \neq i}^{q} e_j$ and thus $f(r_k) = 0 \bmod e_k$. On the other hand, it holds for $r \neq r_k$ that $f(r) = (r_k - r) \prod_{j=1, j \neq k}^{q} e_j \bmod e_k$. Since $l_r < l_e$, we have that $|r_k - r| < e_k$ and as the e_i are distinct primes, we finally get that $\gcd((r_k - r) \prod_{j=1, j \neq k}^{q} e_j, e_k) = 1$ and thus $f(r) \neq 0 \bmod e_k$ for $r \neq r_k$.

4.2 The SDH-Based Schemes

Under the SDH assumption, the situation is very similar. Here we also analyze three possible types of forgeries $(m^*, (r^*, s^*, t^*))$: 1.) $t^* \notin \{t_1, \ldots, t_q\}$, 2.) $t^* = t_i$ with $i \in [1; q]$ but $r^* \neq r_i$, and 3.) $t^* = t_i$, $r^* = r_i$ (but $m^* \neq m_i$) with $i \in [1; q]$. Again, we concentrate on the second case. At the beginning, $\mathcal{B}$ is given an SDH challenge $\left(\hat{g}_1, \hat{g}_1^x, \hat{g}_1^{(x^2)}, \ldots, \hat{g}_1^{(x^q)}, g_2, g_2^x\right)$. This time, $\mathcal{B}$ chooses $PK = g_2^x$. In the SDH setting, Equation (1) transfers to

$$e(s, PK g_2^t) = e(\hat{g}_1^{f(r,x)}, g_2) \Leftrightarrow s^{x+t} = \hat{g}_1^{f(r,x)}. \tag{3}$$

In contrast to the SRSA setting, f is now a polynomial with indeterminate x and maximal degree q. Again, $\mathcal{B}$ must keep $f(r, x)$ and the $\hat{g}_1^{(x^i)}$ secret from $\mathcal{A}$. We define

$$f(r, x) = \sum_{i=1}^{q} r_i \prod_{\substack{j=1 \\ j \neq i}}^{q} (x + t_j) - r \sum_{i=1}^{q} \prod_{\substack{j=1 \\ j \neq i}}^{q} (x + t_j),$$

for $r_1, \ldots, r_q \in \mathcal{R}$ and distinct $t_1, \ldots, t_q \in \mathbb{Z}_p$. Using the SDH challenge, $\mathcal{B}$ can easily compute $\hat{g}_1^{f(r,x)}$ since $f(r, x)$ has maximal degree q. Observe that it always holds that $(f(r, x) - (r_k - r) \prod_{j=1, j \neq k}^{q} (x + t_j))/(x + t_k) \in \mathbb{Z}$. If $r = r_k$, we surely have that $f(r, x)/(x + t_k) \in \mathbb{Z}$. If $r \neq r_k$, then long division gives us $D \in \mathbb{Z}$ with $D \neq 0$ and a new polynomial $\tilde{f}_{t_k}(r, x)$ with coefficients in $\mathbb{Z}$ such that $f(r, x) = \tilde{f}_{t_k}(r, x)(x + t_k) + D$. Similar to the SRSA class, we can find a solution to the SDH challenge from $\mathcal{A}$'s forgery as

$$\left((s^*) \hat{g}_1^{-\tilde{f}_{t^*}(r^*, x)}\right)^{1/D} = \hat{g}_1^{1/(x + t^*)}, t^*.$$

4.3 Security of the Chameleon Hash Signature Class

The chameleon hash class is also tightly secure in the SRSA and the SDH setting. For convenience let $c_i = ch(r_i, m_i)$ for $i \in [1;q]$ and $c^* = ch(r^*, m^*)$. Altogether there are again three types of forgeries to consider: 1) $e^* \notin \{e_1, \ldots, e_q\}$ ($t^* \notin \{t_1, \ldots, t_q\}$), 2) $e^* = e_i$ ($t^* = t_i$) but $c^* \neq c_i$, and 3) $e^* = e_i$ ($t^* = t_i$), $c^* = c_i$ but $m^* \neq m_i$. The proof of 1) is straight-forward and very similar to the proof of Type I forgers of the combining class. The proof of 3) clearly reduces to the security properties of the chameleon hash function. The proof of 2) requires our new technique to set up $f(c)$ ($f(c,x)$). Recall Section 4 where we analyzed the equations $s^e = \hat{u}^{f(c)}$ and $f(c) = \sum_{i=1}^{q} c_i \prod_{j=1, j\neq i}^{q} e_j - c \sum_{i=1}^{q} \prod_{j=1, j\neq i}^{q} e_j$ in the SRSA setting (and $s^{x+t} = \hat{g}_1^{f(c,x)}$ and $f(c,x) = \sum_{i=1}^{q} c_i \prod_{j=1, j\neq i}^{q} (x + t_j) - c \sum_{i=1}^{q} \prod_{j=1, j\neq i}^{q} (x + t_j)$ in the SDH setting).

In the proof of the combining class the c_i are random values ($c_i = r_i$) that can be specified prior to the simulation phase. In the proof of the chameleon hash class we take a similar approach. Now the c_i are the output values of a chameleon hash function. In the initialization phase of the proof we choose q random input pairs $(m'_i, r'_i) \in \mathcal{M} \times \mathcal{R}$, $i \in [1;q]$ to compute the $c_i = \mathsf{CHEval}(PK_{\mathcal{CH}}, m'_i, r'_i)$. Then we prepare the function $f(c)$ ($f(c,x)$) with $C = \{c_1, \ldots, c_q\}$ and a set of q random primes l_e-bit primes (random values $t_1, \ldots, t_q \in \mathbb{Z}_p$) as in the proofs of the combining class. Next, we embed $f(c)$ ($f(c,x)$) in the exponents of the two group elements u, v (a, b). In the simulation phase we give the simulator $SK_{\mathcal{CH}}$ to map the attacker's messages m_i to the prepared c_i by computing $r_i = \mathsf{CHColl}(SK_{\mathcal{CH}}, r'_i, m'_i, m_i)$. In this way we can successfully simulate the signing oracle. In the extraction phase, the properties of the chameleon hash function guarantee that $c^* \notin \{c_1, \ldots, c_q\}$ (otherwise we can break the security of the chameleon hash function). This ensures that we can find a solution to the SRSA challenge (SDH challenge).

4.4 Security Analysis of $\mathcal{S}_{\mathbf{CMB,SRSA}}$

Lemma 3. *In the SRSA setting, suppose the $(t_{SRSA}, \epsilon_{SRSA})$-SRSA assumption holds and $\mathcal{V}$ is a $(t_{comb}, \epsilon_{comb}, \delta_{comb})$-combining function. Then, the combining signature class as presented in Section 3.1 is (q, t, ϵ)-secure[3] against adaptive chosen message attacks provided that*

$$q = q_{SRSA}, \quad \epsilon \leq \frac{9}{2}\epsilon_{SRSA} + 3\epsilon_{comb} + 3q\delta_{comb} + \frac{3q^2}{|E|} + 9 \cdot 2^{2-l_n/2}, \quad t \approx t_{SRSA}.$$

The proof of Lemma 3 is the first step in the proof of Theorem 1. It implies that the original Camenisch-Lysyanskaya, the Fischlin and the Zhu's signature scheme are tightly secure against existential forgeries under adaptive chosen message attacks.

[3] Using explicit bounds on the prime counting function [16], we can lower bound the number of primes in E for $l_e \geq 7$ as $|E| > (2^{l_e} - 1)/(\ln(2^{l_e} - 1) + 2) - (2^{l_e - 1} - 1)/(\ln(2^{l_e - 1} - 1) - 4)$.

Proof. Assume that $\mathcal{A}$ is a forger that (q, t, ϵ)-breaks the strong existential unforgeability of $\mathcal{S}_{\text{CMB,SRSA}}$. Then, we can construct a simulator $\mathcal{B}$ that, by interacting with $\mathcal{A}$, solves the SRSA problem in time t_{SRSA} with advantage ϵ_{SRSA}. We consider three types of forgers that after q queries $m_1, \ldots, m_q$ and corresponding responses $(r_1, s_1, e_1), \ldots, (r_q, s_q, e_q)$ partition the set of all possible forgeries $(m^*, (r^*, s^*, e^*))$. In the proof, we treat all types of attackers differently. At the beginning, we let $\mathcal{B}$ guess with probability at least $\frac{1}{3}$ which forgery $\mathcal{A}$ outputs. Lemma 3 then follows by a standard hybrid argument. We assume that $\mathcal{B}$ is given an SRSA challenge instance $(\hat{u}, n)$. Let $\Pr[S_i]$ denote the success probability of an attacker to successfully forge signatures in Game i.

Type I Forger $(e^* \notin \{e_1, \ldots, e_q\})$

Game$_0$. This is the original attack game. By assumption, $\mathcal{A}$ (q, t, ϵ)-breaks $\mathcal{S}_{\text{CMB,SRSA}}$ when interacting with the signing oracle $\mathcal{O}(SK, \cdot)$. We have that,

$$\Pr[S_0] = \epsilon \ . \tag{4}$$

Game$_1$. Now, $\mathcal{B}$ constructs the values u, v, w using the SRSA challenge instead of choosing them randomly from QR_n. First, $\mathcal{B}$ chooses q random primes $e_1, \ldots, e_q \xleftarrow{\$} E$ and three random elements $t_0', t_0'' \xleftarrow{\$} \mathbb{Z}_{(n-1)/4}$ and $t_0 \xleftarrow{\$} \mathbb{Z}_{3(n-1)/4}$. In the following let $\bar{e} := \prod_{k=1}^{q} e_k$, $\bar{e}_i := \prod_{k=1, k\neq i}^{q} e_k$ and $\bar{e}_{i,j} := \prod_{k=1, k\neq i, k\neq j}^{q} e_k$. The simulator computes $u = \hat{u}^{2t_0\bar{e}}$, $v = \hat{u}^{2t_0'\bar{e}}$, $w = \hat{u}^{2t_0''\bar{e}}$ using the SRSA challenge. Since the t_0, t_0', t_0'' are not chosen uniformly at random from $\mathbb{Z}_{p'q'}$ we must analyze the success probability for $\mathcal{A}$ to detect our construction. Observe that $(n-1)/4 = p'q' + (p'+q')/2 > p'q'$. Without loss of generality let $p' > q'$. Now, the probability of a randomly chosen $x \in \mathbb{Z}_{(n-1)/4}$ not to be in $\mathbb{Z}_{p'q'}$ is

$$\Pr[x \xleftarrow{\$} \mathbb{Z}_{(n-1)/4},\ x \notin \mathbb{Z}_{p'q'}] = 1 - \frac{|\mathbb{Z}_{p'q'}|}{|\mathbb{Z}_{(n-1)/4}|} < \frac{1}{q'+1} < 2^{-(|q'|_2 - 1)} \ .$$

With the same arguments we can show that t_0 is also distributed almost uniformly at random in $\mathbb{Z}_{p'q'}$ and $\mathbb{Z}_{3p'q'}$. Since the e_i are primes smaller than p' and q' it holds that $e_i \not| p'q'$. Therefore, the distribution of the generators is almost equal to the previous game and we get by a union bound that

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)} \ . \tag{5}$$

Game$_2$. Now, $\mathcal{B}$ simulates $\mathcal{O}(SK, \cdot)$ by answering $\mathcal{A}$'s signature queries. Subsequently, set $z_j = z(e_j, m_j)$ and $z^* = z(e^*, m^*)$. The simulator $\mathcal{B}$ sets $PK = n$ and for all $j \in \{1, \ldots, q\}$ it chooses a random $r_j \in \mathcal{R}$ and outputs $\sigma_j = (r_j, s_j, e_j)$ with $s_j = (uv^{r_j}w^{z_j})^{\frac{1}{e_j}} = \hat{u}^{2(t_0 + t_0' r_j + t_0'' z_j)\bar{e}_j}$. The distribution of the so computed values is equal to the previous game and

$$\Pr[S_2] = \Pr[S_1] \ . \tag{6}$$

Game$_3$. Now, consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. Define $\hat{e} = (t_0 + t_0' r^* + t_0'' z^*)$. For $\mathcal{A}$'s forgery it holds that $(s^*)^{e^*} = \hat{u}^{2\bar{e}\hat{e}}$. We also have that

$\gcd(e^*, 2\bar{e}\hat{e}) = \gcd(e^*, \hat{e})$ since by assumption we know $\gcd(e^*, 2\bar{e}) = 1$. We will now analyze the probability for the event $\gcd(e^*, \hat{e}) < e^*$ to happen. If $\gcd(e^*, \hat{e}) = e^*$ (or $\hat{e} = 0 \bmod e^*$) $\mathcal{B}$, simply aborts and restarts. Since $|e^*|_2 = l_e$, it holds that $\gcd(e^*, p'q') < e^*$. Write $t_0 \in \mathbb{Z}_{3(n-1)/4}$ as $t_0 = t_{0,1} + p'q't_{0,2}$ where $t_{0,2} \in [0; 2]$ and $t_{0,1} \in [0, p'q' - 1]$ and observe that $\mathcal{A}$'s view is independent from $t_{0,2}$. Let $T = \hat{e} - p'q't_{0,2}$. We now argue that there exists at most one $\tilde{t}_{0,2} \in [0; 2]$ such that $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$. This is crucial because if $\mathcal{A}$ produces forgeries with $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$ for *all* $\tilde{t}_{0,2} \in [0; 2]$ it always holds that $\gcd(e^*, \hat{e}) = e^*$ and $\mathcal{B}$ cannot extract a solution to the SRSA challenge (using the techniques described below). Assume there exists at least one such $\tilde{t}_{0,2}$. Then, we have that $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$. Let us analyze the remaining possibilities $\tilde{t}_{0,2} \pm 1$ and $\tilde{t}_{0,2} \pm 2$ as $A = T + \tilde{t}_{0,2}p'q' \pm p'q' \mod e^*$ and $B = T + \tilde{t}_{0,2}p'q' \pm 2p'q' \mod e^*$. Since $\gcd(e^*, p'q') < e^*$ we know that $p'q' \neq 0 \bmod e^*$. As $T + \tilde{t}_{0,2}p'q' = 0 \mod e^*$ we must have that $A \neq 0 \bmod e^*$. Also, because e^* is odd we know that $2p'q' \neq 0 \bmod e^*$ and thus $B \neq 0 \bmod e^*$. So, because there can only exist at most one $\tilde{t}_{0,2} \in [0; 2]$ with $\gcd(e^*, \hat{e}) = e^*$ and since this $\tilde{t}_{0,2}$ is hidden from $\mathcal{A}$'s view, $\mathcal{A}$'s probability to output it is at most $1/3$. This means that with probability at least $2/3$, $\mathcal{B}$ has that $\gcd(e^*, \hat{e}) = d < e^*$. Using $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$, $\mathcal{B}$ can break the SRSA assumption by computing $a, b \in \mathbb{Z}$ with $\gcd(e^*/d, 2\bar{e}\hat{e}/d) = ae^*/d + b2\bar{e}\hat{e}/d = 1$ and

$$\hat{u}^{d/e^*} = \hat{u}^a (s^*)^b, e^*/d.$$

Finally, we have that

$$\Pr[S_3] \geq 2 \cdot \Pr[S_2]/3 \tag{7}$$

and

$$\Pr[S_3] = \epsilon_{\text{SRSA}} \ . \tag{8}$$

Plugging in Equations (4)–(8), we get that $\epsilon \leq \frac{3}{2}\epsilon_{\text{SRSA}} + 3 \cdot 2^{2-l_n/2}$.

Type II Forger ($e^* = e_i$ and $r^* \neq r_i$)
We only present the differences to the previous proof.

Game$_1$. First, $\mathcal{B}$ randomly chooses q *distinct* l_e-bit primes $e_1, \ldots, e_q$ and q random elements $r_1, \ldots, r_q \in \mathcal{R}$. Additionally, it chooses three random elements t_0, t_0', t_0'' from $\mathbb{Z}_{(n-1)/4}$. Next, $\mathcal{B}$ computes $u = \hat{u}^{2\left(t_0\bar{e} + \sum_{i=1}^{q} r_i\bar{e}_i\right)}$, $v = \hat{u}^{2\left(t_0'\bar{e} - \sum_{i=1}^{q}\bar{e}_i\right)}$, and $w = \hat{u}^{2t_0''\bar{e}}$ using the SRSA challenge. Again,

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)} \ . \tag{9}$$

Game$_2$. Now $\mathcal{B}$ simulates the signing oracle $\mathcal{O}(SK, \cdot)$. On each signature query m_j with $j \in \{1, \ldots, q\}$, $\mathcal{B}$ responds with $\sigma_j = (r_j, s_j, e_j)$ using the precomputed r_j and e_j and computing s_j as

$$s_j = \hat{u}^{2\left((t_0 + t_0'r_j + t_0''z_j)\bar{e}_j + \sum_{i=1, i\neq j}^{q}(r_i - r_j)\bar{e}_{i,j}\right)} \ .$$

Since we have chosen the e_i to be distinct primes we have by a union bound that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|} \ . \tag{10}$$

Game$_3$. Now consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption there is an $i \in \{1, \ldots, q\}$ with $e^* = e_i$ and $r_i \neq r^*$. Then we have that

$$\left((s^*) \cdot \hat{u}^{-2\left((t_0+t_0' r^*+t_0'' z^*)\bar{e}_i+\sum_{j=1, j\neq i}^{q}(r_j-r^*)\bar{e}_{i,j}\right)}\right)^{e_i} = \hat{u}^{2(r_i-r^*)\bar{e}_i} \ .$$

Since $|r_i - r^*| < e_i$ and e_i is an odd prime, we get $\gcd(2(r_i - r^*), e_i) = 1$ and as before we can compute $\hat{u}^{\frac{1}{e_i}}$ which is a solution to the SRSA challenge.

$$\Pr[S_3] = \epsilon_{\text{SRSA}} \ . \tag{11}$$

Summing up Equations (9)–(11), we get $\epsilon \leq \epsilon_{\text{SRSA}} + q^2/|E| + 3 \cdot 2^{2-l_n/2}$.

Type III Forger ($e^* = e_i$ and $r^* = r_i$)
There are only minor differences as compared to the previous proof.

Game$_1$. First, $\mathcal{B}$ randomly chooses q l_e-bit primes $e_1, \ldots, e_q$ and q random $z_1, \ldots, z_q \in \mathcal{Z}$, Then, $\mathcal{B}$ draws three random elements t_0, t_0', t_0'' from $\mathbb{Z}_{(n-1)/4}$. Next, $\mathcal{B}$ computes u, v, and w as $u = \hat{u}^{2\left(t_0\bar{e}+\sum_{i=1}^{q} z_i\bar{e}_i\right)}$, $v = \hat{u}^{2t_0'\bar{e}}$, and $w = \hat{u}^{2\left(t_0''\bar{e}-\sum_{i=1}^{q}\bar{e}_i\right)}$.

$$\Pr[S_1] \geq \Pr[S_0] - 3 \cdot 2^{-(l_n/2-2)} \ . \tag{12}$$

Game$_2$. This game is equal to the previous game except that we require the e_i to be all distinct. We have that

$$\Pr[S_2] \geq \Pr[S_1] - \frac{q^2}{|E|} \ . \tag{13}$$

Game$_3$. Now $\mathcal{B}$ simulates the signing oracle. For each queries m_j with $j \in \{1, \ldots, q\}$, $\mathcal{B}$ computes $r_j = z^{-1}(z_j, m_j)$. If $r_j \notin \mathcal{R}$, $\mathcal{B}$ aborts. Otherwise it outputs the signature $\sigma_j = (r_j, s_j, e_j)$ with s_j being computed as

$$s_j = (uv^{r_j}w^{z_j})^{\frac{1}{e_j}} = \hat{u}^{2\left((t_0+t_0' r_j+t_0'' z_j)\bar{e}_j+\sum_{i=1, i\neq j}^{q}(z_i-z_j)\bar{e}_{i,j}\right)} \ .$$

The properties of the combining function guarantee that the r_j are statistically close to uniform over $\mathcal{R}$ such that,

$$\Pr[S_3] \geq \Pr[S_2] - q\delta_{\text{comb}} \ . \tag{14}$$

Game$_4$. This game is like the previous one except that $\mathcal{B}$ aborts whenever there is a collision such that $z_i = z(r_i, m_i) = z(r_i, m^*) = z^*$ for some r_i. Observe that we must have $m^* \neq m_i$, otherwise $\mathcal{A}$ just replayed the i-the message/signature pair. For all t_{comb}-time attackers this happens with probability at most ϵ_{comb}. Therefore,

$$\Pr[S_4] \geq \Pr[S_3] - \epsilon_{\text{comb}} \ . \tag{15}$$

Consider $\mathcal{A}$'s forgery $(m^*, (r^*, s^*, e^*))$. By assumption, there is one index $i \in \{1, \ldots, q\}$ with $e^* = e_i$ and $r^* = r_i$. For this index it holds that

$$\left((s^*) \cdot \hat{u}^{-2\left((t_0+t_0' r^*+t_0'' z^*)\bar{e}_i+\sum_{j=1, j\neq i}^{q}(z_j-z^*)\bar{e}_{i,j}\right)}\right)^{e_i} = \hat{u}^{2(z_i-z^*)\bar{e}_i} \ .$$

Since we have excluded collisions, it follows that $z_i \neq z^*$. As $|z_i - z^*| \leq e_i$, $\mathcal{B}$ can compute $\hat{u}^{\frac{1}{e_i}}$ as a solution to the SRSA challenge. Finally,

$$\Pr[S_4] = \epsilon_{\text{SRSA}} \ . \tag{16}$$

Equations (12)–(16) show $\epsilon \leq \epsilon_{\text{SRSA}} + \epsilon_{\text{comb}} + q\delta_{\text{comb}} + q^2/|E| + 3 \cdot 2^{2-l_n/2}$.

Acknowledgement. I would like to thank Mathias Herrmann, Tibor Jager, Eike Kiltz, and Maike Ritzenhofen for useful comments on earlier drafts of this paper and the anonymous referees of EUROCRYPT'11 for helpful comments and suggestions.

References

1. Au, M.H., Susilo, W., Mu, Y.: Constant-size dynamic k-TAA. In: Prisco, R.D., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 111–125. Springer, Heidelberg (2006)
2. Bernstein, D.J.: Proving tight security for rabin-williams signatures. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 70–87. Springer, Heidelberg (2008)
3. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. J. Cryptology 21(2), 149–177 (2008)
4. Camenisch, J., 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)
5. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
6. Chevallier-Mames, B., Joye, M.: A practical and tightly secure signature scheme without hash function. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 339–356. Springer, Heidelberg (2006)
7. Coron, J.S., Naccache, D.: Security analysis of the gennaro-halevi-rabin signature scheme. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 91–101. Springer, Heidelberg (2000)
8. Cramer, R., Shoup, V.: Signature schemes based on the Strong RSA assumption. ACM Trans. Inf. Syst. Secur. 3(3), 161–185 (2000)
9. Fischlin, M.: The cramer-shoup strong-RSA Signature scheme revisited. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
10. 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)

11. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM J. Comput. 17(2), 281–308 (1988)
12. 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)
13. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS. The Internet Society, San Diego (2000)
14. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
15. Okamoto, T.: Efficient blind and partially blind signatures without random oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
16. Rosser, B.: Explicit bounds for some functions of prime numbers. American Journal of Mathematics 63(1), 211–232 (1941)
17. Shamir, A., Tauman, Y.: Improved online/offline signature schemes. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 355–367. Springer, Heidelberg (2001)
18. Zhu, H.: New digital signature scheme attaining immunity to adaptive-chosen message attack. Chinese Journal of Electronics 10(4), 484–486 (2001)
19. Zhu, H.: A formal proof of Zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155 (2003), http://eprint.iacr.org/

Adaptive Pseudo-free Groups and Applications*

Dario Catalano[1], Dario Fiore[2,**], and Bogdan Warinschi[3]

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy
catalano@dmi.unict.it
[2] École Normale Supérieure, CNRS - INRIA, Paris, France
dario.fiore@ens.fr
[3] Dept. Computer Science, University of Bristol, UK
bogdan@cs.bris.ac.uk

Abstract. In this paper we explore a powerful extension of the notion of pseudo-free groups, proposed by Rivest at TCC 2004. We identify, motivate, and study pseudo-freeness in face of *adaptive* adversaries who may learn solutions to other non-trivial equations before having to solve a new non-trivial equation.

We present a novel, carefully crafted definition of *adaptive* pseudo-freeness that walks a fine line between being too weak and being unsatisfiable. We show that groups that satisfy our definition yield, via a generic construction, digital and network coding signature schemes.

Finally, we obtain concrete constructions of such schemes in the RSA group by showing this group to be adaptive pseudo-free. In particular, we demonstrate the generality of our framework for signatures by showing that most existing schemes are instantiations of our generic construction.

1 Introduction

Background. The search for abstractions that capture the essential security properties of primitives and protocols is crucial in cryptography. Among other benefits, such abstractions allow for modular security analysis, reusable and scalable proofs. The random oracle model [3], the universal composability framework [7] and variants [1,2,17] of the Dolev-Yao models [9] are results of this research direction. Most such abstractions (the above examples included) tackle mostly primitives and protocols and are not concerned with the more basic mathematical structures that underlie current cryptographic constructions. One notable exception is the work on pseudo-free groups, a notion put forth by Hohenberger [14] and later refined by Rivest [18]. In this paper we continue the investigation of this abstraction.

Roughly speaking, a computational group $\mathbb{G}$ (a group where the group operations have efficient implementations) is pseudo-free if it behaves as a free group as far as a computationally bounded adversary is concerned. More specifically, a

* A full version of this paper is available at http://eprint.iacr.org/2011/053
** Work partially done while student at University of Catania.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 207–223, 2011.

group is pseudo-free if an adversary who is given a description of the group cannot find solutions for non-trivial equations. Here, non-triviality means that the equation does not have a solution in the free group. For instance, in a pseudo-free group given a random element a it should be hard to find a solution for an equation of the form $x^e = a$, when $e \neq 1$, or for the equation $x_1^2 x_2^4 = a^5$, but not for the equation $x_1 x_2^3 = a^5$. This last equation is trivial since it can be solved over the free group (it has $x_1 = a^2, x_2 = a$ as solution in the free group) and a solution in the free group immediately translates to a solution over $\mathbb{G}$. The notion of pseudo-freeness generalizes the strong RSA assumption (when $\mathbb{G}$ is an RSA group) but also numerous other assumptions currently used in cryptography; see [18] for further details. Rivest's conjecture that the RSA group is pseudo-free was largely settled by Micciancio [16] who proved that this is indeed the case when the RSA modulus is the product of two safe primes.

In its most basic form that had been studied so far, the notion of pseudo-free groups did not lend itself easily to applications. The problem is that in most of the interesting uses of the RSA group the adversary is not only given a description of the group, but often he is allowed to see solutions to non-trivial equations before having to come up with his own new equation and solution. This is the case for example in RSA-based signature schemes where one can think of a signature as the solution to some non-trivial equation. A chosen-message attack allows the adversary access to an oracle that solves (non-trivial) equations over the group, and a forgery is a solution to a new equation.

This problem was recognized early on by Rivest [18] who also left as open problems the design of a notion of pseudo-freeness for adaptive adversaries and, of course, whether such groups exist. In this paper we put forth such a notion, prove that the RSA group is adaptive pseudo-free, and exhibit several applications for adaptive pseudo-free groups. We detail our results next.

Adaptive pseudo-free groups. We first extend the notion of pseudo-freeness to adaptive adversaries. Informally, we consider an adversary that can see solutions for some equations and has as goal solving a new non-trivial equation. As explained above, this scenario captures typical uses of groups in cryptography.

Our definition involves two design decisions. The first is to fix the type of equations for which the adversary is allowed to see solutions and how are these equations chosen: too much freedom in selecting these equations immediately leads to potentially unsatisfiable notions, whereas too severe restrictions may not model the expected intuition of what an adaptive adversary is and may not allow for applications. In the definition that we propose, equations are selected from a distribution over the set of equations. Importantly, the distribution depends on a parameter supplied by the adversary. This models the idea that in applications, the adversary may have some control over how the equations are selected. Different choices for this distribution lead to a variety of adversaries from very weak ones where no equation is provided (precisely the setting of pseudo-freeness proposed earlier), to a setting where the adversary has no influence on the choice of equations, and ending with the very strong notion where the adversary basically selects the equations on his own.

The second issue is to define what is a non-trivial equation in the adaptive setting. Indeed, previous definitions of triviality do not apply since in our new setting the adversary knows additional relations between the group elements which in turn may help him in solving additional equations. We define non-triviality in a way motivated by existing uses of groups in cryptography and an analysis of equations over quotients of free groups. Our definition is for the case of univariate equations but can be easily extended to multivariate equations as well as systems of equations.

Generic constructions for signatures. Our definition of pseudo-freeness is parametrized by a distribution over equations. We show that for any distribution in a class of distributions that satisfy certain criteria, one can construct secure digital signatures and network coding signature schemes. The requirements on the distribution include the ability to efficiently check membership in the support of the distribution, and a property on the distribution of the exponents in the equation.

Our generic construction for network coding signatures is secure in the vanilla model based only on the adaptive pseudo-freeness of the underlying group. Any instantiation of such groups would thus yield network signature schemes secure in the standard model. Indeed, given the instantiation that we discuss below, our framework yields the first RSA-based network coding homomorphic signature scheme secure in the standard model.

The RSA group is adaptive pseudo-free. Next, we turn to proving that the RSA group is adaptive pseudo-free. We do so for a class of distributions closely related but slightly more general than the distributions that yield signatures schemes. We show that an adversary that contradicts pseudo-freeness of the RSA group with respect to the distribution can be used to contradict the strong RSA assumption. We also prove that the RSA group is pseudo-free for a weaker version of adaptive adversaries who output their inputs to the distribution non-adaptively, but in this case the proof is for a larger class of distributions.

We do not attempt to prove adaptive pseudo-freeness of the RSA group for multivariate equations. While this is potentially an interesting topic for further research, we are not aware of cryptographic applications where such equations are used.

Instantiations. An appealing interpretation of the proof of adaptive pseudo-freeness for the RSA group is that it distills the core argument that underlies the typical security proofs for signatures based on the strong RSA assumption. Each such proof explains how a signature forgery can be used to break strong RSA. In this sense our proof is a generalization to a broader (abstractly defined) set of equations rather than the particular equations that define an individual signature scheme.

Indeed, we show that almost *all* strong RSA signature schemes are instances of our generic construction. We explain how to obtain the schemes by Cramer and Shoup [8], Fischlin [10], Camenisch and Lysyanskaya [6], Zhu [19], Hofheinz

and Kiltz [13][1], and that by Gennaro, Halevi, and Rabin [11] by instantiating our generic distribution in appropriate ways. The security of all of these schemes follows as a corollary from the security of our generic construction.

Other related work. In [13] Hofheinz and Kiltz introduced the notion of programmable hash function (PHF), an information theoretic tool that, when used in groups where the discrete logarithm is hard, allows for black box proofs for various cryptographic constructions dealing with adaptive attacks. Among other things they show how to construct generic signatures from the strong RSA assumption. Still, PHF and adaptive pseudo free groups seem to abstract away different aspects of strong-RSA based signatures (for instance PHF can deal with bilinear groups while our framework allows to encompass network signatures).

1.1 Preliminaries and Notation

In our work we use the notion of *division intractable functions*. Informally, a function H is division intractable if an adversary $\mathcal{A}$ cannot find $x_1, x_2, \ldots, x_t, y$ such that: $y \neq x_i$ and $H(y)$ divides the product of the $H(x_i)$'s. It is easy to see that this notion is satisfied by any function that maps inputs to (distinct) prime numbers. Such mappings can be instantiated without making any cryptographic assumptions (see [5] for a construction), but they are not very efficient in practice. Gennaro *et al.* introduced in [11] the notion of division intractable hash functions and also showed how to get practical implementations of them.

For lack of space, we defer the interested reader to the full version for other standard definitions and notations used throughout the paper.

2 Static Pseudo-free Groups

As warm up, we recall the notion of pseudo-free groups as introduced by Rivest [18]. To distinguish it from the notions that we develop in this paper we refer to the older notion as *static* pseudo-free groups.

Free abelian groups. For any set of symbols $A = \{a_1, a_2, \ldots, a_m\}$ we write A^{-1} for the set of symbols $A^{-1} = \{a_1^{-1}, a_2^{-1}, \ldots, a_m^{-1}\}$. Let $X = \{x_1, \ldots, x_n\}$ and $A = \{a_1, \ldots, a_m\}$ be two disjoint sets of variables and constant symbols. An equation over X with constants in A is a pair $\lambda = (w_1, w_2) \in (X^* \times A^*)$. We usually write an equation $\lambda = (w_1, w_2)$ as $w_1 = w_2$ and looking ahead (we will only consider these equations over abelian groups), we may also write it as $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n} = a_1^{s_1} a_2^{s_2} \cdots a_m^{s_m}$ where $\{e_1, \ldots, e_n\}$ and $\{s_1, \ldots, s_m\}$ are integers.

Let $(G, \cdot)$ be an arbitrary abelian group and $\alpha : A \to G$ be an interpretation of the constants in A as group elements. We write λ_α for the equation λ interpreted over G via α. An evaluation $\psi : X \to G$ is a solution for λ_α if

$$\psi(x_1)^{e_1} \cdots \psi(x_n)^{e_n} = \alpha(a_1)^{s_1} \cdots \alpha(a_m)^{s_m}.$$

[1] We remark that for the case of Hofheinz-Kiltz signatures our framework captures a variant of the main instantiation with non-optimized params. Extending the framework to deal with smaller eponents is an interesting open problem.

Any equation λ over X and A can be viewed as an equation over the free group $\mathcal{F}(A)$ via the interpretation $1_A : A \to \mathcal{F}(A)$ that maps a to a. It can be easily shown [18,16] that the equation λ_{1_A} has a solution in $\mathcal{F}(A)$ if and only if $\forall i = 1, \ldots, m$, it holds $gcd(e_1, \ldots, e_n) \mid s_i$. We call such equations *trivial*, in the sense that these equations have solutions over the free group. All of the other equations are deemed *non-trivial*.

STATIC PSEUDO-FREE GROUPS. A computational group consists of a (finite) set of representations for the group elements together with efficient implementations for the two group operations. Informally, a computational group is pseudo-free if it is hard to find an equation which is unsatisfiable over the free group, together with a solution in the computational group. It is worth noting that if the order of the group is known then finding solutions for non-trivial equations may be easy. Therefore, the notion of pseudo-free groups holds for families $\mathcal{G} = \{\mathbb{G}_N\}_{N \in \mathcal{N}_k}$ of computational groups where N is chosen at random from the set of indexes $\mathcal{N}_k$ (typically these are the strings of length k) and the corresponding order $ord(\mathbb{G}_N)$ is hidden to the adversary.

In the following we recall the formal definition given by Micciancio in [16] (which is similar to that of Rivest [18]). The adversary that is considered in the following definition is static (in that it is only allowed to see a description of the group, but obtains no further information). To distinguish this class of groups from others that we define in this paper we call them *static pseudo-free groups*.

Definition 1 (Static Pseudo-Free Groups [16]). *A family of computational groups $\mathcal{G} = \{\mathbb{G}_N\}_N$ is* static pseudo-free *if for any set A of polynomial size $|A| = p(k)$ (where k is a security parameter), and PPT algorithm $\mathcal{A}$, the following holds. Let $N \in \mathcal{N}_k$ be a randomly chosen group index, and define $\alpha : A \to \mathbb{G}_N$ by choosing $\alpha(a)$ uniformly at random in $\mathbb{G}_N$, for each $a \in A$. Then, the probability (over the selection of α) that on input (N, α) adversary $\mathcal{A}$ outputs an equation λ and a solution ψ for λ_α is negligible in k.*

3 Adaptive Pseudo-free Groups

A ROUGH DEFINITION. The notion described above requires an adversary to produce a solution for some non-trivial equation only given some randomly chosen generators to be used in the equation, but no additional information. In contrast, the notion that we develop attempts to capture the idea that an adversary against the computational group gets to see several equations with solutions, and then attempts to solve a new *non-trivial* equation. A typical cryptographic game that captures this situation involves an adversary $\mathcal{A}$ who works against a Challenger as follows.

Setup. The Challenger chooses a random instance of the computational group $\mathbb{G}_N$ (by picking a random index $N \xleftarrow{\$} \mathcal{N}_k$) from a family $\mathcal{G} = \{\mathbb{G}_N\}_{N \in \mathcal{N}_k}$. Then he fixes an assignment $\alpha : A \to \mathbb{G}_N$ for the set of constants and gives $(\alpha, \mathbb{G}_N)$ to the adversary.

Equations queries. In this phase the adversary is allowed to see non-trivial equations together with their solutions.

Challenge. At some point the adversary is supposed to output a new "non-trivial" equation λ^* (defined by $(e^*, \boldsymbol{s^*})$) together with a solution ψ^*.

Notice that the above description incorporates an assumption that we make for simplicity, namely that all equations are univariate. In general, any univariate equation over A is of the form: $x^e = a_1^{s_1} a_2^{s_2} \cdots a_m^{s_m}$. For the case of static pseudo-free groups, this restriction is justified by a lemma that was proved by Micciancio in [16]. Informally the lemma says that any (multivariate) equation and solution (λ, ψ) can be efficiently transformed into a univariate equation and solution (λ', ψ'). Whilst we extend the definition of trivial equations to the multivariate case (for lack of space it is given in the full version of the paper), it would be interesting to see if a similar lemma is possible in the context of adaptive pseudo-freeness.

The general definition of pseudo-freeness that we sketched above leaves open two important points: 1) How are the equations for which the adversary sees solutions produced? and 2) What does "non-trivial equation" mean when other equations and solutions are given? We discuss and give answers to these two problems in Sections 3.1 and 3.2 respectively.

3.1 A Spectrum of Adaptive Adversaries

The second phase of the above generic game requires that adversaries be given non-trivial equations together with their solutions, so we need to clarify how are these equations produced. Here we identify a whole spectrum of possible choices. The weakest definition one might consider is one where the adversary does not have any control over these equations. For instance, this means that, whenever the Challenger is queried in the second phase, the Challenger chooses an equation λ_i (more precisely it chooses its exponents $(e_i, \boldsymbol{s_i})$) and gives λ_i and its solution in $\mathbb{G}$, ψ_i, to the adversary. Unfortunately, in such a game the adversary is not really adaptive: it may receive all the equations and solutions at once.

The strongest possible notion, and perhaps the most natural one, would be to consider an adversary that is allowed to choose equations λ_i (namely their respective exponents $(e_i, \boldsymbol{s^i})$) in any way it wants. In particular the choice of the equations can be done in an adaptive way, namely $\mathcal{A}$ asks for an equation, sees its solutions, then chooses another equation and so on. We call this definition "Strong Adaptive Pseudo-freeness". Unfortunately this choice seems to lead to an unrealizable notion[2]. We therefore settle on an intermediary variant where the adversary is allowed to be adaptive, but still cannot choose the equations in a completely arbitrary way. Instead, we consider a setting where the equations are selected from the set of all equations according to some distribution over which the adversary has some *limited* control. We formulate this limitation via

[2] For example, it is not clear at all if a group like $\mathbb{Z}_N^*$ can be proved strongly-adaptive pseudo-free under any reasonable assumption (e.g. Strong RSA).

a *parametric distribution* φ over the set of all possible equations. Sampling from such a distribution requires some parameter M of some appropriate length which is provided by the adversary. The distribution then produces a tuple of $m+1$ integers which for expressivity we write $(e, \boldsymbol{s})$. Here e is an integer (the exponent for the variable) and $\boldsymbol{s}$ is a vector of m integers (the exponents for the generators). The idea is that once the parameter M is fixed, $\varphi(M)$ is some fixed distribution from which $(e, \boldsymbol{s})$ are drawn. Notice that the two ends of the spectrum can be modeled via appropriate choices of φ.

3.2 Non-trivial Equation w.r.t. Other Equations

Our definition of adaptive pseudo-freeness requires an adversary to find a solution to a non-trivial equation. In the original setting of Rivest, non-triviality of an equation simply meant that the equation has no solution in the free group. In our setting, non-triviality is less clear: the adversary is already given solutions for some equations which may lead to solutions for other equations that are difficult to solve otherwise. In this section we develop a notion of triviality for equations given solutions to other equations. Our ultimate goal is to characterize, using the world and vocabulary afferent to free groups those equations that cannot be solved in the computational group.

General deducibility modulo equations. We frame the discussion in slightly more general terms to obtain a framework suitable for talking about non-triviality of both univariate and multi-variate equations.

Let $\mathcal{F}$ be the free abelian group generated by the set $\{a_1, a_2, \ldots, a_m\}$ and let $\Lambda \subseteq \mathcal{F} \times \mathcal{F}$ be an arbitrary binary relation on $\mathcal{F}$ that models equalities between words in $\mathcal{F}$ (equations with solutions can be thought of as such relations). We therefore aim to characterize the set of all equalities that can be derived from Λ. Recall that eventually these equalities are interpreted over computational groups, hence there are two ways for an adversary to derive new equalities. The first is to use the group operations and their properties. For example, if $\Lambda = \{a_1 a_2 = a_1^2 a_4\}$, then it can also be derived that $a_1 a_2^2 = a_1^2 a_4 a_2 = a_1^3 a_4^2$, where the first equality is obtained by simply multiplying a_2 to the known equation, and the second equality follows using the commutativity of $\mathcal{F}$ and the known equality. The second possibility reflects an ability that computational adversaries have (when working against computational groups). Specifically, if an equality of the form $w_1^q = w_2^q$ can be derived in a computational group, then the equality $w_1 = w_2$ can also be derived (provided that q is relatively prime with the order of the group). Furthermore, since we search for an abstraction independent of the order of the group, we have to consider the above possibility for any q. The following definition is motivated by the above discussion.

Definition 2. *Let $\mathcal{F}$ be a freely generated abelian group and let $\Lambda \subseteq \mathcal{F} \times \mathcal{F}$ be an arbitrary binary relation on $\mathcal{F}$. Let $\equiv_\Lambda$ be the smallest congruence on $\mathcal{F}$ that:*

- $\Lambda \subseteq \equiv_\Lambda$
- $\forall q \in \mathbb{N}, \forall w_1, w_2 \in \mathcal{F},\ w_1^q \equiv_\Lambda w_2^q \implies w_1 \equiv_\Lambda w_2$.

Then, w_1 and w_2 are trivially equal with respect to Λ if $w_1 \equiv_\Lambda w_2$.

Next, we derive an explicit description for $\equiv_\Lambda$. Let

$$\Lambda = \{(w_{1,1}, w_{2,1}), (w_{1,2}, w_{2,2}), \ldots, (w_{1,t}, w_{2,t})\}.$$

Consider the binary relation R_Λ on $\mathcal{F}$ defined by: $(w_1, w_2) \in R_\Lambda$ if and only if there exist $l_1, l_2, \ldots, l_t \in \mathbb{Q}$ such that $w_1 = w_2 \cdot \Pi_{i=1}^{t}(w_{1,i}^{-1} \cdot w_{2,i})^{l_i}$. Here, exponentiation of a word $w = a_1^{s_1} a_2^{s_2} \ldots a_n^{s_n}$ with a rational number $l = p/q$ is defined (in the obvious way) if and only if q divides $\gcd_{1 \le i \le n} p \cdot s_i$. The following proposition states that $\equiv_\Lambda$ and R_Λ are one and the same relation. Its proof is in the full version of the paper.

Proposition 1. *Let R_Λ and $\equiv_\Lambda$ defined as above. Then $(w_1, w_2) \in R_\Lambda$ if and only if $(w_1, w_2) \in \equiv_\Lambda$.*

Trivial equations. Using the notion of deducibility modulo equations developed above we can now specify the class of equations that we consider trivial (given solutions for the equations in some set Λ). For simplicity, we focus on the case of univariate equations which is more relevant for the cryptographic applications of this paper. The definition easily extends to the case of multivariate equations (for completeness this variation is given in the full version). Assume that we are given a set of equations

$$\Lambda = \left\{ x^{e_k} = a_1^{s_1^k} \cdots a_m^{s_m^k} \right\}_{k=1}^{t}$$

together with $\{\phi_k\}_{k=1}^{t}$, their corresponding solutions. (Notice that these are equations in a computational group; solutions for these equations may simply not exist in a free group). Let $\mathcal{F}$ be the the free abelian group generated by $\{\phi_1, \phi_2, \ldots, \phi_t, a_1, a_2, \ldots, a_m\}$ (interpreted as symbols). The equations in Λ induce a binary relation on $\mathcal{F}$ which (by a slight abuse of notation) we also call Λ. So $\Lambda = \{(\phi_k^{e_k}, a_1^{s_1^k} \cdots a_m^{s_m^k}) \mid 1 \le k \le t\}$. The following definition simply is a particular instance of Definition 2 to the case of univariate equations.

Definition 3. *Equation $x^{e^*} = a_1^{s_1^*} \cdots a_m^{s_m^*}$ is trivial with respect to Λ if the equation has a solution over $\mathcal{F}/\equiv_\Lambda$.*

We use the characterization of $\equiv_\Lambda$ that we gave earlier to explicitly determine the class of trivial equations. Let

$$x^{e^*} = a_1^{s_1^*} \cdots a_m^{s_m^*} \tag{1}$$

be an equation that has a solution over $\mathcal{F}/\Lambda$. Let $\phi = \phi_1^{k_1} \cdots \phi_t^{k_t} a_1^{v_1} \cdots a_m^{v_m}$ be such a solution. From the explicit characterization of $\equiv_\Lambda$ there exists $l_1, \ldots, l_t$ in $\mathbb{Q}$ such that

$$(\phi_1^{k_1} \cdots \phi_t^{k_t} a_1^{v_1} \cdots a_m^{v_m})^{e^*} = a_1^{s_1^*} a_2^{s_2^*} \cdots a_m^{s_m^*} \cdot \Pi_{i=1}^{t} \left(\phi_i^{e_i} \cdot \Pi_{k=1}^{m} a_k^{-s_k^i} \right)^{l_i} \tag{2}$$

Since equality is standard equality over $\mathcal{F}$, the relation above translates (via symbol by symbol matching of exponents) into the following requirement. Equation (1) has a solution if there exist $v_1 \cdots v_m, k_1 \cdots k_t$ in $\mathbb{Z}$ and $l_1, \ldots, l_t \in \mathbb{Q}$ such that:

1. $k_i e^* = e_i \cdot l_i$ (for all $1 \leq i \leq t$)
2. $v_i e^* = s_i^* - \sum_{j=1}^{t} l_j s_i^{(j)}$ (for all $1 \leq i \leq m$)

The converse of the above statement is also true: if integers $v_1, \cdots v_m$, $k_1, \ldots, k_t$ and rationals $l_1, \cdots, l_t$ exist such that Equation 2 holds then $\phi = \phi_1^{k_1} \cdots \phi_t^{k_t} a_1^{v_1} \cdots a_m^{v_m}$ is a solution for Equation (1) over $\mathcal{F}/\equiv_\Lambda$.

Finally, we express these two conditions in a more compact matrix form which will be simpler to use in our proofs. Given the set of equations $\Lambda = \left\{x^{e_k} = a_1^{s_1^k} \cdots a_m^{s_m^k}\right\}_{k=1}^{t}$ we define the following quantities:

$$\Sigma = \begin{bmatrix} s_1^1 & \cdots & s_1^t \\ \vdots & & \vdots \\ s_m^1 & \cdots & s_m^t \end{bmatrix} \quad \text{and} \quad E = \begin{bmatrix} 1/e_1 & & 0 \\ & 1/e_2 & \\ 0 & & \ddots \\ & & & 1/e_t \end{bmatrix}$$

These quantities are dependent on Λ but we do not show the dependency explicitly to avoid heavy notation.

Proposition 2 (Trivial equation w.r.t. a set of equations). *Equation* $\lambda^* : x^{e^*} = a_1^{s_1^*} \cdots a_m^{s_m^*}$ *is* trivial *w.r.t* Λ *if and only if:*

$$\exists k \in \mathbb{Z}^t, V \in \mathbb{Z}^m : e^*(\Sigma E k + V) = \boldsymbol{s^*}$$

where $\boldsymbol{s^*} = [s_1^* \cdots s_m^*]^T$.

The proposition follows by simply setting $l_i = k_i \frac{e^*}{e_i}$ for all $1 \leq i \leq t$.

3.3 A Definition of Adaptive Pseudo-free Groups

The definition of adaptive pseudo-freeness that we give below is for a set A of m generators, a computational group $\{\mathbb{G}_N\}_N$ and is parameterized by a distribution $\varphi(\cdot)$ as discussed in Section 3.1.

Setup. The Challenger chooses a random instance of the computational group $\mathbb{G}_N$ (by picking a random index $N \xleftarrow{\$} \mathcal{N}_k$) from a family $\mathcal{G} = \{\mathbb{G}_N\}_{N \in \mathcal{N}_k}$. Then he fixes an assignment $\alpha : A \to \mathbb{G}_N$ for the set A of generators and a specific parametric distribution φ for the exponents. The adversary is given in input the assignment $\alpha : A \to \mathbb{G}_N$ and the descriptions of the computational group and the parametric distribution φ.

Equations queries. In this phase the adversary is allowed to adaptively query the Challenger on equations and see their solutions. More precisely, $\mathcal{A}$ controls the queried equations via the parametric distribution φ. Namely, for

each query it chooses a parameter M_i and hands it to the Challenger. The Challenger runs $(e_i, \boldsymbol{s^i}) \leftarrow \varphi(M_i)$, computes the solution ψ_i for the equation λ_i, which is $x^{e_i} = a_1^{s_1^i} \cdots a_m^{s_m^i}$ and gives $(\psi_i, e_i, \boldsymbol{s^i})$ to $\mathcal{A}$.

Challenge. Once the adversary has seen the solutions, then it is supposed to output an equation λ^* (defined by $(e^*, \boldsymbol{s^*})$) together with a solution ψ^*. We say that $\mathcal{A}$ wins this game if λ^* is a non-trivial equation.

Definition 4 (Adaptive pseudo-free groups). *$\mathcal{G}$ is a family of adaptive pseudo-free groups w.r.t. distribution φ, if for any set A of polynomial size, any PPT adversary $\mathcal{A}$ wins in the game above with at most negligible probability.*

We restate several of the reasons that justify the above definition. Although the definition is parametrized by a distribution, we feel this is the right way of modeling an adversary who is adaptive but not all-powerful. As explained, by varying the distribution one obtains a large spectrum of potentially interesting instantiations, starting with static pseudo-freeness all the way to strong adaptive pseudo-freeness. Finally, we show that for some fixed distributions adaptive pseudo-freeness implies immediately secure signature schemes.

4 Applications of Adaptive Pseudo-free Groups

In this section we show that adaptive pseudo-free groups yield interesting cryptographic applications. Specifically, we prove that any group that is pseudo-free with respect to a distribution ϕ from a class of of parametric distributions that we specify immediately yields a secure signature scheme. We also explain how to adapt the distribution and the proof to obtain the analogous result for (non-strongly) unforgeable schemes.

4.1 Signatures from Adaptive Pseudo-free Groups

The class of parametric distributions φ_ℓ. In this section we introduce a specific class of parametric distributions $\varphi_\ell : \{0,1\}^\ell \rightarrow \mathbb{Z}^{1+m} \times \{0,1\}^{a(\ell)}$. For any input $M \in \{0,1\}^\ell$ and an integer ℓ, $\varphi_\ell(M)$ outputs a tuple $(e, \boldsymbol{s}, r)$ such that:

- r is a binary string taken according to some efficiently samplable distribution D_r (that may depend on M), for which collisions happen with at most negligible probability;
- $e = H(r)$ where $H : \{0,1\}^{a(\ell)} \rightarrow \{0,1\}^{b(\ell)}$ is a division intractable function (see Section 1.1) and $a(\cdot)$ and $b(\cdot)$ are polynomials;
- $s_1 = 1$;
- $s_i \in \mathbb{Z}_e$ (i.e. $s_i < e$) $\forall i = 2, \ldots, m$ for some efficiently samplable distribution D_{s_i}.

Also we require that $\varphi_\ell(M)$ produces an output $(e, \boldsymbol{s}, r)$ for which one can efficiently tell that it belongs to the support of $\varphi_\ell(M)$. Formally, we require that φ_ℓ

is equipped with an efficient algorithm $Ver_{\varphi_\ell}(\cdot,\cdot,\cdot,\cdot)$ that, on input $(e, \boldsymbol{s}, r, M)$, outputs 1 if $(e, \boldsymbol{s}, r)$ is in the support of $\varphi_\ell(M)$ and 0 otherwise. Moreover we require $Ver_{\varphi_\ell}(e, \boldsymbol{s}, r, M)$ to be such that, for all PPT adversaries $\mathcal{A}$ the following probability is at most negligible

$$\Pr\left[(e, \boldsymbol{s}, r, M_1, M_2) \leftarrow \mathcal{A}(\varphi_\ell) : \begin{array}{l} M_1 \neq M_2 \wedge Ver_{\varphi_\ell}(e, \boldsymbol{s}, r, M_1) = 1 \\ \wedge Ver_{\varphi_\ell}(e, \boldsymbol{s}, r, M_2) = 1 \end{array}\right]$$

Signature scheme construction. We now show how to build a signature scheme from any family of groups $\mathcal{G}$ that is adaptive pseudo-free w.r.t. $\hat{\varphi} \in \varphi_\ell$.

Let $\hat{\varphi}$ be a parametric distribution taken from the class φ_ℓ and let $\mathcal{G}$ be a family of groups that is adaptive pseudo-free w.r.t. $\hat{\varphi}$. Then we have the following signature scheme $\mathsf{PFSig} = (\mathsf{KG}, \mathsf{Sign}, \mathsf{Ver})$:

$\mathsf{KG}(1^k)$**.** Let $A = \{a_1, \ldots, a_m\}$ and $X = \{x\}$ be the sets of constants variable symbols. The key generation algorithm selects a random group $\mathbb{G}$ from $\mathcal{G}$, fixes an assignment $\alpha : A \to \mathbb{G}$ for the symbols in A and finally it sets $\mathsf{vk} = (X, A, \alpha, \mathbb{G}, \hat{\varphi})$ as the public verification key and $\mathsf{sk} = ord(\mathbb{G})$ as the secret signing key. The input space of $\hat{\varphi}$, $\mathcal{M}$, is taken as the message space of the signature scheme.

$\mathsf{Sign}(\mathsf{sk}, M)$**.** The signing algorithm proceeds as follows:

- $(e, \boldsymbol{s}, r) \leftarrow \hat{\varphi}(M)$
- Use $ord(\mathbb{G})$ to solve the equation $x^e = a_1^{s_1} \cdots a_m^{s_m}$. Let $\psi : X \to \mathbb{G}$ be the satisfying assignment for x. The algorithm outputs $\sigma = (e, \boldsymbol{s}, r, \psi)$ as the signature for M.

$\mathsf{Ver}(\mathsf{vk}, M, \sigma)$**.** To verify a signature σ for a message M, the verification algorithm proceeds as follows:

- Check if $Ver_{\hat{\varphi}}(e, \boldsymbol{s}, r, M) = 1$ and if the equation $x^e = a_1^{s_1} \cdots a_m^{s_m}$ is satisfied in $\mathbb{G}$ by $\psi(x)$.
- If both the checks are true, output 1, otherwise 0.

Security of the signature scheme. In this section we prove the security of the proposed signature scheme under the assumption that $\mathcal{G}$ is adaptive pseudo-free w.r.t. $\hat{\varphi}$. In particular we can state the following theorem (whose proof is omitted for lack of space):

Theorem 1. *If $\mathcal{G}$ is a family of adaptive pseudo-free groups w.r.t. distribution $\hat{\varphi} \in \varphi_\ell$, then the signature scheme* PFSig *is strongly-unforgeable under chosen-message attack.*

Notice that if one relaxes a bit the requirements on the parametric distribution $\hat{\varphi}$, Theorems 1 leads to different flavors of digital signature schemes. For instance, one might consider the distribution $\hat{\varphi}'$, which slightly generalizes the parametric distribution $\hat{\varphi}$ as follows. $\hat{\varphi}'$ is exactly as $\hat{\varphi}$ with the only difference that s_2 is chosen uniformly in $\mathbb{Z}_B$ for some value $B > e$. It is easy to rewrite the proof of Theorem 1 in order to show the following

Corollary 1. *If $\mathcal{G}$ is a family of adaptive pseudo-free groups w.r.t. distribution $\hat{\varphi}'$, then the signature scheme* PFSig *is unforgeable under chosen-message attack.*

Informally what this corollary is saying is that by (slightly) generalizing the parametric distribution one gets a signature scheme where unforgeability is guaranteed only for previously unsigned messages (i.e. the scheme is not strongly unforgeable).

4.2 Network Coding Signatures from Adaptive Pseudo-free Groups

In this section we show that our framework allows to encompass network coding signature schemes as defined and constructed by [4,12]. In particular, by combining previous theorems with ideas from [12] we construct the first RSA-based network coding homomorphic signature scheme provably secure without random oracle. In the following we will represent files V to be signed as collections $(v^{(1)}, \ldots, v^{(m)})$ where each $v^{(i)}$ is a n-dimensional vector of the form $(v_1, \ldots, v_n)$. To sign V the signer signs every single vector $v^{(i)}$ separately. Informally this is done using a signature scheme that allows some form of (controlled) malleability. In this way, if we interpret signatures as solutions of non trivial equations, one can easily compute solutions for any linear combination of the given equations. This simple observation, when combined with ideas from [12], can be used to construct a secure signature scheme for network coding without random oracles.

Our Network Coding Signature Scheme. For lack of space we defer the interested reader to the full version of this paper or to the works [4,12] for a background on network coding signatures. Here we describe our network coding signature scheme. First, however, we discuss some additional details required to properly present the scheme. As already mentioned, a file to be signed is expressed as a set of vectors $(v^{(1)}, \ldots, v^{(m)})$ of n components each. Such vectors will be prepended with m unitary vectors $u^{(i)}$ (of m components each). Let us denote with $w^{(i)}$ the resulting vectors.

Using a similar notation as [12] we denote with $Q = \{0, \ldots, q-1\}$ (for some prime q) the set from which coefficients are (randomly) sampled. We denote with L an upper bound on the path length from the source to any target. By these positions $B = mq^L$ denotes the largest possible value of u-coordinates in (honestly-generated) vectors. Moreover denoting with M an upper bound on the magnitude of the coordinates of initial vectors $v^{(1)}, \ldots, v^{(m)}$, we set $B^* = MB$.

Let φ_N be the following parametric distribution. It takes as input some, large enough, random identifier fid, a vector space V and a bound B^*. Let ℓ_s be a security parameter and ℓ be an integer such that $2^\ell > B^*$, compute $e = H(\mathsf{fid})$ where $H : \{0,1\}^* \to \{0,1\}^\ell$ is a division intractable function. Next, for each $v^{(i)} = (v_1^{(i)}, \ldots, v_n^{(i)}) \in V$ it proceeds as follows. First it samples (uniformly and at random) a $\ell + \ell_s$-bit random integer s_i and outputs $(s_i, u^{(i)}, v^{(i)})$. The global output of φ_N is then $(e, \{(s_i, u^{(i)}, v^{(i)})\}_{i=1}^m)$.

Notice that φ_N is a simple extension of distribution $\hat{\varphi}'$ described above. It is straightforward to show that it fits the requirements of corollary 1 as well.

Let $\mathcal{G}$ be a family of groups that is adaptive pseudo-free w.r.t. φ_N. Then we have the following signature scheme $\mathsf{NetPFSig}$:

$\mathsf{NetKG}(1^k, n)$. Let $A = \{g, g_1, \ldots, g_n, h_1, \ldots, h_m\}$ and $X = \{x\}$ be the sets of constants variable symbols. The key generation algorithm selects a random

group $\mathbb{G}$ from $\mathcal{G}$, fixes an assignment $\alpha : A \to \mathbb{G}$ for the symbols in A and finally it sets $\mathsf{vk} = (X, A, \alpha, \mathbb{G}, \varphi_N)$ as the public verification key and $\mathsf{sk} = ord(\mathbb{G})$ as the secret signing key. The input space of φ_N, $\mathcal{M}$, is taken as the set of m-dimensional vectors whose components are positive integers of magnitude at most M.

$\mathsf{Sign}(\mathsf{sk}, V)$**.** The signing algorithm proceeds as follows. A random identifier fid for the vector space V is chosen. Next, it runs $\varphi_N(V, B^*, \mathsf{fid})$ to get back $(e, \{(s_i, u^{(i)}, v^{(i)})\}_{i=1}^m)$. Finally, for $i = 1$ to m, it uses $ord(\mathbb{G})$ to solve the equation

$$x_i^e = g^{s_i} \prod_{j=1}^{m} h_j^{u_j^{(i)}} \prod_{j=1}^{n} g_j^{v_j^{(i)}}$$

Let $\psi : X \to \mathbb{G}$ be the satisfying assignment for x_i and $\sigma_i = (e, s_i, u^{(i)}, v^{(i)}, \mathsf{fid}, \psi)$ the signature for $w^{(i)}$. The algorithm outputs $\sigma = (\sigma_1, \ldots \sigma_m)$ as the signature for V.

$\mathsf{Ver}(\mathsf{vk}, V, \sigma)$**.** To verify a signature σ for a vector space V, the verification algorithm proceeds as follows

- Check if $Ver_{\varphi_N}(e, V, B^*, \mathsf{fid}, \{(s_i, u^{(i)}, v^{(i)})\}_{i=1}^m) = 1$[3] and if the equations $x_i^e = g^{s_i} g_1^{v_1^{(i)}} \cdots g_n^{v_n^{(i)}} h_1^{u^{(i)}} \cdots h_m^{u_m^{(i)}}$ are all satisfied in $\mathbb{G}$ by $\psi(x_i)$.
- If all the checks are true, output 1, otherwise 0.

$\mathsf{Combine}(\mathsf{vk}, \mathsf{fid}, w_1, \ldots, w_\ell, \sigma_1, \ldots, \sigma_\ell)$**.** To combine signatures σ_i, corresponding to vectors w_i sharing the same fid, a node proceeds as follows.

- It discards any w_i having u coordinates negative or larger than $B/(mq)$, or having v coordinates negative or larger than $B^*/(mq)$. Without loss of generality we keep calling $w_1, \ldots w_\ell$ the remaining vectors.
- It chooses random $\alpha_1, \ldots \alpha_\ell \in Q$, set $w = \sum_{i=1}^{\ell} \alpha_i w_i$ and it outputs the signature $\sigma = (e, s, w, \mathsf{fid}, \psi)$ on w which is obtained by computing

$$\psi = \prod_{i=1}^{\ell} \psi_i^{\alpha_i}, \qquad s = \sum_{i=1}^{\ell} \alpha_i s_i$$

One can easily rewrite the proof of corollary 1 to prove the following.

Theorem 2. *If $\mathcal{G}$ is a family of adaptive pseudo-free groups w.r.t. distribution φ_N, then the* NetPFSig *signature scheme described above is a secure (homomorphic) network coding signature.*

5 The RSA Group Is Adaptive Pseudo-free

In Section 3 we have defined the notion of adaptive pseudo-free groups and in Section 4 have shown a class of parametric distributions (called φ_ℓ) that allows

[3] We implicitly assume that the Ver_{φ_N} verification algorithm rejects immediately if any of the u coordinates is negative or larger than B, or if any of the v coordinates is negative or larger than B^*.

to build signatures from the sole assumption that a family of groups is adaptive pseudo-free w.r.t. $\hat{\varphi} \in \varphi_\ell$. At this stage, it is therefore interesting to find a computational group candidate to be proved adaptive pseudo-free. As proved by Micciancio in [16], the only group that we know to be pseudo-free is the RSA group $\mathbb{Z}_N^*$ of integers modulo N, where N is the product of two "safe" primes and the sampling procedure takes elements from QR_N. Therefore we aim to prove adaptive pseudo-freeness for the same group.

A PARAMETRIC DISTRIBUTION $\hat{\varphi}$. First of all we need to define the specific parametric distribution for which we will prove adaptive pseudo-freeness of the RSA group.

Let us consider the following $\hat{\varphi} : \mathcal{M} \to \mathbb{Z} \times \mathbb{Z}^m \times \{0,1\}^*$, where $\mathcal{M} = \{0,1\}^\ell$. For any input $M \in \mathcal{M}$, $\hat{\varphi}(M)$ outputs a tuple $(e, \boldsymbol{s}, r)$ that is defined as follows:

- r is a random binary string, taken from some sufficiently large input domain.
- $e = H(r)$ where $H : \{0,1\}^* \to \{0,1\}^\ell$ is a division intractable function.
- $s_1 = 1$.
- s_2 is uniformly distributed in $\mathbb{Z}_e$.
- For $3 \leq i \leq m$, each s_i is taken with an arbitrary (but efficiently samplable) distribution D_{s_i} in $\mathbb{Z}_e$ such that the tuple $s_3, \ldots, s_m$ is binding to M[4].

The verification algorithm $Ver_{\hat{\varphi}}(e, \boldsymbol{s}, r, M)$ checks that $e = H(r)$ and that $s_3, \ldots, s_m$ are binding w.r.t. M. It is straightforward to verify that $\hat{\varphi}$ is contained in the class φ_ℓ defined in section 4.1.

We state the following theorem (the proof is omitted for lack of space).

Theorem 3. *If the Strong-RSA Assumption holds, then $\mathbb{Z}_N^*$ is adaptive pseudo-free w.r.t. $\hat{\varphi}$.*

As a corollary of the above theorem we can prove adaptive pseudo-freeness of the RSA group w.r.t. two new parametric distributions $\hat{\varphi_s}, \hat{\varphi_{ch}} \neq \hat{\varphi}$ which still are within the class φ_ℓ defined in section 4.1. In particular $\hat{\varphi_s}$ is a variant of $\hat{\varphi}$ where: $s_2 = 0$ and for all $i = 3$ to m, $s_i \in \{0, \ldots, p\}$ such that p is at most polynomial in the security parameter (and of course $p < e$).

Corollary 2. *If the Strong-RSA Assumption holds, then $\mathbb{Z}_N^*$ is adaptive pseudo-free w.r.t. $\hat{\varphi_s}$.*

The proofs follows from that of Theorem 3. The intuition here is that when the s_i's are small they can be guessed in advance with non-negligible probability.

Instead $\hat{\varphi_{ch}}$ is a variant of $\hat{\varphi}$ where: $s_2 = 0$ and $s_3, \ldots, s_m \in \mathbb{Z}_e$ are obtained as output of a chameleon hash function $CH(M; R)$ computed on the parameter M and with randomness R.

Corollary 3. *If the Strong-RSA Assumption holds, and CH is a chameleon hash function, then $\mathbb{Z}_N^*$ is adaptive pseudo-free w.r.t. $\hat{\varphi_{ch}}$.*

[4] This means that there exists an efficient algorithm that on input $(M, s_3, \ldots, s_m)$ outputs 1 if $s_3, \ldots, s_m$ are created w.r.t. M.

The proof is the same as in Corollary 2. The intuition here is that one can use the chameleon property of CH in the simulation to "prepare" the s_i's in advance.

Weak adaptive pseudo-freeness of the RSA group. One may also consider a weaker notion of adaptive pseudo-freeness where the adversary is forced to choose the parameters $M^1, \ldots, M^t$ of its queries at the beginning of the game, i.e. before receiving the description of the group from the challenger. In the full version of the paper we show that the proof of Theorem 3 still holds even w.r.t. a slightly more general distribution than $\hat{\varphi}$ where the entire tuple $(e, s_2, \ldots, s_m)$ needs to be bound to M. It is then trivial to see that starting from a weak-adaptive pseudo-free group our results of section 4.1 lead to the construction of signature schemes that are weakly-secure.

6 A Framework for Strong RSA-Based Signatures

In this section we show that, by appropriately instantiating the parametric distribution $\hat{\varphi}$, Theorems 1 and 3 yield essentially all the known constructions of Strong RSA-based digital signatures in the standard model (to the best of our knowledge). Due to space limits we only briefly summarize these results. Precise details are in the full version.

Cramer-Shoup's signatures [8]. While Cramer-Shoup's scheme seems based on the difficulty of solving a system of two equations, we observe that for only one of these two equations the signing process is required to find a solution (using the secret key) while the other equation is, *de facto*, a chameleon hash function computed on the message. Their scheme is then a special case of our general framework applying via Corollary 3.

Fischlin's signatures [10]. Fischilin's scheme can be seen as a special case of our framework as the distribution of its exponents fits the case of $\hat{\varphi}$, for which Theorem 3 applies.

Camenisch-Lysyanskaya's signatures [6]. This signature can be seen as an instance of our framework since its distribution is an instance of $\hat{\varphi}'$, for which Corollary 1 applies.

Zhu's signatures [19,20]. Zhu's scheme is captured by our general framework as the distribution of its exponents is a special instance of $\hat{\varphi}$.

Hofheinz-Kiltz's signatures [13]. Hofheinz and Kiltz show in [13] how to use programmable hash functions to get a new efficient signature scheme based on Strong RSA. It is not hard to notice that the security of their scheme basic scheme[5] emerges from Corollary 2.

Gennaro-Halevi-Rabin's signatures [11]. The scheme in [11] fits our framework for weakly-secure signature scheme (see section 5) when using a distribution in which $e = H(m)$ and H is a division intractable hash function.

[5] By basic scheme here we mean the version of the HK scheme where the public exponents are set as 160-bit primes. The interesting thing about the analysis made in [13], is that, by using programmable hash functions, one can consider much smaller primes (i.e. 70 bit long ones). The present formulation of our framework, however, does not allow for such an optimization.

A new network signature from Strong RSA. It is easy to see that combining the results of Theorem 3 and Theorem 2 we obtain a concrete instantiation of the network coding signature scheme given in Section 4.2 whose security is thus based on Strong RSA in the standard model. We notice that our scheme is not as efficient as the one proposed by Gennaro *et al.* in [12], but it is secure in the standard model.

7 Conclusion

In this paper we have introduced a formal definition of adaptive pseudo-freeness. We have shown that under reasonable conditions the RSA group is adaptive pseudo-free for moduli that are products of safe primes, and exhibited the first direct cryptographic applications of adaptive pseudo-free groups: under some mild conditions, pseudo-free groups yield secure digital signature schemes.

There are several interesting problems that we have not addressed. Here we enumerate some of them. The first obvious one, originally posed by Rivest, is what other groups used in cryptography are pseudo-free. A new construction would lead, via our framework, to new signature schemes for example. Our results for RSA are only for univariate equations. It should be interesting to either justify this restriction through an analogue of Micciancio's Lemma, or, if this is not possible, extend our study to multi-variate equations. The one-more RSA inversion problem has a strong flavor of adaptive pseudo-freeness but does not fit our framework. In particular, its relation with the strong RSA problem is an interesting open problem. Nevertheless, studying the relation between these two problems within our framework seems to be an interesting direction. Finally, we manage to prove adaptive pseudo-freeness for a large class of parametric distributions sufficient for cryptographic applications. It should be interesting to understand how far one can go with the limitations that we impose on the adversary by trying to enlarge this class.

Acknowledgements. We thank Eike Kiltz for clarifications regarding [13] as well as for useful suggestions. The work described in this paper has been supported in part by the European Commission through the ICT programme under contract ICT-2007-216676 ECRYPT II.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology 20(3), 395 (2007)
2. Backes, M., Pfitzmann, B., Waidner, M.: A composable cryptographic library with nested operations. In: ACM CCS 2003, Washington D.C., USA, October 27-30, pp. 220–230. ACM Press, New York (2003)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM CCS 1993, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press, New York (1993)

4. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 68–87. Springer, Heidelberg (2009)
5. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
6. Camenisch, J., 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)
7. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, Las Vegas, Nevada, USA, October 14-17, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
8. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. In: ACM CCS 1999, Kent Ridge Digital Labs, Singapore, November 1-4, pp. 46–51. ACM Press, New York (1999)
9. Dolev, D., Yao, A.C.: On the security of public key protocols. In: FOCS, pp. 350–357 (1981)
10. Fischlin, M.: The Cramer-Shoup strong-RSA signature scheme revisited. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 116–129. Springer, Heidelberg (2002)
11. 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)
12. Gennaro, R., Katz, J., Krawczyk, H., Rabin, T.: Secure network coding over the integers. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 142–160. Springer, Heidelberg (2010)
13. 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)
14. Hohenberger, S.: The cryptographic impact of groups with infeasible inversion. Master's thesis, Massachusetts Institute of Technology, EECS Dept. (2003)
15. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000, San Diego, California, USA, February 2-4. The Internet Society, San Diego (2000)
16. Micciancio, D.: The RSA group is pseudo-free. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 387–403. Springer, Heidelberg (2005)
17. Micciancio, D., Warinschi, B.: Soundness of formal encryption in the presence of active adversaries. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 133–151. Springer, Heidelberg (2004)
18. 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)
19. Zhu, H.: New digital signature scheme attaining immunity to adaptive chosen-message attack. Chinese Journal of Electronics 10(4), 484–486 (2001)
20. H. Zhu. A formal proof of Zhu's signature scheme. Cryptology ePrint Archive, Report 2003/155 (2003), http://eprint.iacr.org/

Commuting Signatures and Verifiable Encryption

Georg Fuchsbauer*

Dept. Computer Science, University of Bristol, UK
georg@cs.bris.ac.uk

Abstract. Verifiable encryption allows one to encrypt a signature while preserving its public verifiability. We introduce a new primitive called *commuting signatures and verifiable encryption* that extends this in multiple ways, such as enabling encryption of both signature and message while proving validity. More importantly, given a ciphertext, a signer can create a verifiably encrypted signature on the encrypted (unknown) message, which leads to the same result as first signing the message and then verifiably encrypting the message/signature pair; thus, signing and encrypting commute. Our instantiation is based on the recently introduced *automorphic signatures* and Groth-Sahai proofs, which we show to be homomorphic. We also prove a series of other properties and provide a novel approach to simulation.

As an application, we give an instantiation of *delegatable anonymous credentials*, a primitive introduced by Belenkiy et al. Our construction is arguably simpler than theirs and it is the first to provide *non-interactive* (and thus concurrently secure) issuing and delegation protocols, which are significantly more efficient. Moreover, the size of our credentials and the cost of verification are less than half of those of the previous instantiation. All our constructions are proven secure in the standard model under known non-interactive assumptions.

Keywords: Verifiably encrypted signatures, blind signatures, anonymous credentials, Groth-Sahai proofs.

1 Introduction

Verifiably encrypted signatures let us sign a message, encrypt the signature, and make a proof asserting that the ciphertext contains a valid signature. Suppose the message is only available as an encryption. We cannot make a signature on the plaintext, as this would contradict the security of the encryption scheme[1]. But could we instead, given a ciphertext, produce a *verifiable encryption* of a signature on the plaintext?

We show that such a functionality is feasible and moreover give a practical instantiation of it. We then use this new primitive to build the first non-interactively delegatable anonymous credential scheme: given an encrypted public key, a delegator can make a verifiably encrypted certificate for the key, which acts as a credential.

* Work done while at École normale supérieure, Paris, France. The author has been supported by the French ANR 07-TCOM-013-04 PACE Project, the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II and EPSRC Grant EP/H043454/1.

[1] Given two messages and the encryption of one of them, a signature on the plaintext could be used to decide which one was encrypted.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 224–245, 2011.

Delegatable Anonymous Credentials. Access control that respects users' privacy concerns is a challenging problem in security. To gain access to resources, a participant must prove to possess the required credential issued by an authority. To increase manageability of the system, the authority usually does not issue credentials directly to each user, but relies on intermediate layers in the hierarchy. For example, a system administrator issues credentials to webmasters for using his server; the latter may then create forums and delegate rights to moderators, who can give posting privileges to users.

Web-based social network services are enjoying a huge popularity and represent another area of application for credentials. Registered users can be given credentials to access services, which they delegate to introduce and recommend friends and friend of friends. The recent rise of concern about protection of privacy in such networks motivates *anonymous* credentials: a user can obtain a credential and prove possession of it without revealing neither her identity nor that of the user who delegated it to her.

In practice, (non-anonymous) delegation of rights is usually realized by certifying (i.e., signing) the public key of the delegated user. Consecutive delegation leads to a certification chain, consisting of public keys and certificates linking them, starting with the *original issuer* of the credential. A user in the chain can delegate the credential by signing the delegatee's public key and appending the certificate to the credential.

Anonymous credentials [Cha85, Dam90, Bra99, LRSW00, CL01, CL04, BCKL08] aim to provide a functionality similar to certificates without revealing information about the user's identity when obtaining or showing a credential. However, the goal of reconciling delegatability and anonymity remained elusive—until recently. Chase and Lysyanskaya [CL06] show theoretical feasibility of delegatable anonymous credentials, but their size is exponential in the number of delegations. A breakthrough was then made in [BCC+09], where Belenkiy et al. (BCCKLS) introduce a new approach using a non-interactive zero-knowledge (NIZK) proof system [BFM88] with *randomizable* proofs: anyone can transform such a proof into a new proof of the same statement that cannot be linked to the original one. A credential is a *proof of knowledge* of a certification chain that can be randomized before being delegated or shown, which guarantees anonymity and unlinkability.

In their model, each user holds a secret key which can be used to produce multiple unlinkable pseudonyms *Nym*. A user A can be known to user O as $Nym_A^{(O)}$ and to B as $Nym_A^{(B)}$. Given a credential issued by O for $Nym_A^{(O)}$, A can transform it into a credential from O for $Nym_A^{(B)}$ and show it to B. Moreover, A can delegate the credential to user C, known to A as $Nym_C^{(A)}$. C can then show a credential from O for $Nym_C^{(D)}$ to user D (without revealing neither $Nym_A^{(C)}$ nor $Nym_C^{(A)}$), or redelegate it. Delegation preserves anonymity, in that delegator and delegatee learn nothing more about each other than their respective pseudonyms. This is formalized by requiring that there exist a *simulator* that can produce pseudonyms and credentials for them without knowing any secrets.

In the instantiation of [BCC+09], delegation is fairly complex and interactive—in contrast to non-anonymous credentials, where it suffices to know a user's public key in order to issue or delegate a credential to her. We bridge this gap by giving an instantiation that enables *non-interactive* delegation: given a pseudonym *Nym*, a delegator can produce a ready credential for the holder of *Nym* without any interaction. Note that a non-interactive delegation protocol immediately yields security against

concurrent attacks, where an adversary might simultaneously run protocols for delegating and being delegated credentials with honest users. This was not considered in the BCCKLS model.

Commuting Signatures. Our main building block for non-interactively delegatable anonymous credentials is a new primitive we call *commuting signature and verifiable encryption* (or *commuting signature* for short), which we sketch in the following and formally define in Section 3. It combines a digital signature scheme, an encryption scheme and a proof system with the following properties: given a verification key *vk*, a message M and a signature Σ on M under *vk*, we can encrypt any subset of $\{vk, M, \Sigma\}$ and add a proof (which leaks no more information) that the plaintexts are a key, a message and a valid signature—which makes the encryptions verifiable.

For consistency with the Groth-Sahai methodology [GS08], we also say *commitment* instead of *encryption*, as their commitments to group elements, which we will use, are *extractable*, i.e., we can recover the committed value (and thus "decrypt") using an *extraction key*. An extractable commitment to a signature together with a proof of validity is a *proof of knowledge* (PoK) of a signature, and at the same time a *verifiably encrypted signature* (VES) [BGLS03, RS09][2].

We denote committing to signatures by Com, committing to messages by $\mathsf{Com}_{\mathcal{M}}$, and proving validity by Prove. A proof for a committed signature is denoted by $\widetilde{\pi}$, and for a committed message by $\bar{\pi}$. If both are committed, we write π, and if the verification key is committed too, we write $\widehat{\pi}$ ("$\sim$" for signature and "$\wedge$" for $\underline{vk}$). Besides allowing us to prove validity of committed values, a commuting signature scheme provides the following functionalities, neither of which requires knowledge of the extraction key:

SigCom. Given a commitment $\mathbf{C}_M$ to a message M and a signing key *sk*, SigCom produces a commitment $\mathbf{c}_\Sigma$ to a signature Σ on M under *sk*, and a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$.

$\mathsf{AdC}_{\mathcal{S}}$. Given a commitment $\mathbf{C}_M$ to M, a signature Σ on M and a proof $\bar{\pi}$ of validity of Σ on the content of $\mathbf{C}_M$, we can make a commitment $\mathbf{c}_\Sigma$ to Σ using randomness ρ_Σ. Then $\mathsf{AdC}_{\mathcal{S}}$ ("adapt when committing to signature") allows us to adapt $\bar{\pi}$ to a proof for $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$: given $(\mathbf{C}_M, \Sigma, \rho_\Sigma, \bar{\pi})$ it returns a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$. $\mathsf{AdD}_{\mathcal{S}}$ ("adapt proof when *de*committing") does the converse: given a committed message $\mathbf{C}_M$, a committed signature $\mathbf{c}_\Sigma$ together with the used randomness ρ_Σ, and a proof π for $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$, $\mathsf{AdD}_{\mathcal{S}}$ outputs a proof $\bar{\pi}$ of validity of the signature Σ on the committed message.

$\mathsf{AdC}_{\mathcal{M}}$. Analogously we define proof adaptation for the *message*. Given M, a commitment $\mathbf{c}_\Sigma$ to a signature on M and a proof of validity $\widetilde{\pi}$, $\mathsf{AdC}_{\mathcal{M}}$ transforms the proof to the case when the message is committed as well. $\mathsf{AdD}_{\mathcal{M}}$ is given commitments $\mathbf{C}_M$ to a message M and $\mathbf{c}_\Sigma$ to a signature, the randomness ρ_M for $\mathbf{C}_M$ and a proof π. It adapts π to a proof $\widetilde{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a valid signature on M.

$\mathsf{AdC}_{\mathcal{K}}$. We can also adapt proofs when (de)committing to the *verification key*. Given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$, a proof of validity π w.r.t. a verification key *vk*, and

[2] While for VES, encryption suffices to be one-way (*opacity* means it is hard to extract a signature), we require verifiable encryptions of different signatures to be *indistinguishable*.

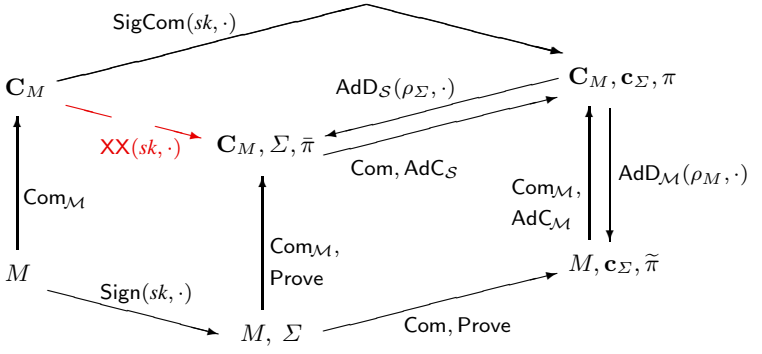


Fig. 1. Diagram representing a system of commuting signatures and verifiable encryption

randomness ρ_{vk}, $\mathsf{AdC}_{\mathcal{K}}$ outputs a proof $\widehat{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a signature on the content of $\mathbf{C}_M$ valid under vk given as a commitment $\mathbf{c}_{vk}$ with randomness ρ_{vk}. $\mathsf{AdD}_{\mathcal{K}}$ is given $(vk, \rho_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma, \widehat{\pi})$ and adapts the proof $\widehat{\pi}$ for $(\mathbf{c}_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma)$, where $\mathbf{c}_{vk}$ commits to vk with randomness ρ_{vk}, to a proof for $(vk, \mathbf{C}_M, \mathbf{c}_\Sigma)$.

We require that committing, signing and the above functionalities all *commute* with each other, that is, it does not matter in which order they are executed; e.g., signing a message, committing to the message and the signature, and proving validity yields the same as committing to the message and then running SigCom. Thus, the diagram in Figure 1 commutes. Note that due to the argument given in Footnote 1, there cannot exist a functionality XX that is given a commitment $\mathbf{C}_M$ to a message M and a secret key sk, and outputs a signature Σ on M.

Besides verifiably encrypted signatures, commuting signatures imply *blind signatures*, and moreover *CL signatures* [CL02] and *P-signatures* [BCKL08], both building blocks for protocols providing privacy. They let a user obtain a signature on a committed value from a signer by running an *issuing* protocol. The user can then make a proof of knowledge of that signature, which is verifiable given the commitment. SigCom provides a non-interactive issuing, which directly gives the user a (randomizable) proof of knowledge of such a signature (see the full version [Fuc10]).

Instantiating Commuting Signatures. Blind signatures [Cha83, PS96] enable a user to obtain a signature on a message in a way that the signer cannot link the resulting message/signature pair to its issuing. In [Fuc09, AFG+10] the author gives an efficient implementation with *round-optimal* issuing [Fis06], where after sending information to the signer, the user can immediately derive the blind signature from the signer's response. In this scheme, the user *randomizes* the message, makes (extractable) commitments to the message and the randomness, and adds a witness-indistinguishable (WI) proof that the commitments contain the correct values.

The user sends these values to the signer, who learns nothing about the message from them. The signer fabricates a "pre-signature", which the user, knowing the values used to randomize the message, can transform into a signature on the message. The actual

blind signature is a WI proof of knowledge (PoK) of this signature, which prevents the signer from linking it to the signing session. This PoK is instantiated with Groth-Sahai (GS) commitments and proofs [GS08] for *pairing-product equations* (PPE), and the message space consists of pairs of group elements.

We require a lot more: the signer, without knowing the randomness used to hide the message, should not only make a commitment to a signature (which he cannot know—Footnote 1) on an unknown message, but in addition give a proof that this signature is valid. While the described blind signature scheme has the nice property that during issuing the user obtains an actual signature on the message, we show that its true potential has not yet been exploited. We first observe that the values the user sends to the signer for a blind signature can be seen as a commitment to the message. We then show that they actually suffice for the signer to *directly*—without the help of the user—construct a proof of knowledge of a signature on the message.

This is made possible by the specific structure of the signature, the fact that the commitments are homomorphic and a series of properties of the proof system. We prove that, besides being randomizable, Groth-Sahai proofs are *homomorphic*[3] w.r.t. the statement they prove (the product of two proofs is a proof for the product of the equations they prove), they are independent of parts of the statement—in some cases even of the committed value—, and there are ways to "blindly" transform a proof for one statement into a proof for another statement.

Instantiating Delegatable Anonymous Credentials. Belenkiy et al. [BCC+09] show that Groth-Sahai (GS) proofs can be randomized and combine them with an authentication scheme for secret keys to construct delegatable credentials. A pseudonym *Nym* is a commitment to the user's secret key and a credential is a proof of knowledge of an authentication chain. To issue or delegate, the issuer and the user jointly compute a PoK of an authenticator on the content of the user's pseudonym. In the case of delegation, the issuer prepends her own credential, after randomizing it. Their authentication scheme must satisfy strong security notions (*F-unforgeability* and *certification security*), since secret keys cannot be extracted from the commitments, and an adversary against it must be allowed to ask for authenticators *on* as well as *under* the attacked key.

We avoid these notions and interactivity of delegation by following a more modular approach replacing the authenticators on secret keys by commuting signatures on verification keys. The underlying signatures are *automorphic* [Fuc09], which means that they are Groth-Sahai compatible and their verification keys lie in the message space—which is a requirement for delegation. A credential is then a chain of *verification keys* and *certificates* (as in the non-anonymous case), which are all given as commitments completed with proofs of validity.

Commuting signatures enable non-interactive issuing and delegation: given a user's pseudonym Nym_U (i.e., a commitment to his verification key), the issuer can produce a commitment $\mathbf{c}_\Sigma$ to a signature on the value committed in Nym_U and a proof π of validity using SigCom. In the case of issuing, the credential is $(\mathbf{c}_\Sigma, \pi)$ and is verified by checking π on the issuer's public key, Nym_U and $\mathbf{c}_\Sigma$. In the case of delegation, the issuer also randomizes her own credential $cred_I$, yielding a credential $cred_I{}'$ on her pseudonym Nym_I that is unlinkable to $cred_I$. Running $\mathsf{AdC}_\mathcal{K}$, the issuer adapts the proof π (which

[3] For linear equations the homomorphic property of GS proofs was also noted in [DHLW10].

is valid under her verification key) to a proof $\widehat{\pi}$ of validity of the signature contained in $\mathbf{c}_\Sigma$ on the content of the pseudonym Nym_U under the content of the issuer's pseudonym Nym_I. The credential for the user is then $cred_I{}' \parallel Nym_I \parallel (\mathbf{c}_\Sigma, \widehat{\pi})$.

Comparing Our Results to Previous Ones. Replacing the authenticators from the BCCKLS scheme with our automorphic signatures already more than doubles the efficiency. In the full version [Fuc10] we revise the approach to achieving simulatability of credentials. Groth and Sahai show how to simulate *proofs of satisfiability* of equations, consisting of commitments and proofs for the committed values, which are produced by the simulator. However, in order to simulate credentials for a given pseudonym, the simulator has to construct proofs for *given* commitments. Belenkiy et al. therefore double some of the commitments and provide proofs of consistency. We show that our credentials can be directly simulated even if some of the commitments are fixed beforehand.

Finally, our issuing (and delegation) protocol is significantly more efficient. While in [BCC+09], the issuer and the user run a complex two-party protocol using homomorphic encryption and interactive ZK proofs, in our instantiation the issuer simply sends a PoK of a signature. Both schemes are proven secure under the SXDH assumption and different "hidden" variants of the *strong Diffie-Hellman* assumption [BB04] (which are thus "q-type" assumptions): BB-CDH and BB-HSDH, introduced in [BCC+09], for their scheme and ADH-SDH [AFG+10] for ours (see Section 4.1).

Automorphic signatures were combined with GS proofs in [AFG+10] to construct *anonymous proxy signatures* (APS) [FP08]. They also allow one to prove rights in an anonymous way, but there is no anonymity between the delegator and the delegated user. If in our credential scheme we give the extraction key to a tracing authority, and define a *proxy signing algorithm* similar to delegation but outputting a committed signature on a *clear* message, we get an instantiation of APS with *mutually anonymous* delegation.

Subsequent to our work, Blazy et al. [BFPV11] defined a primitive similar to commuting signatures, called *extractable signatures on randomizable ciphertexts*. While their instantiation solely relies on the *decision linear* assumption (DLIN) [BBS04], it is only efficient for small message spaces due to bit-by-bit techniques.

2 Preliminaries

We briefly recall the definitions and security requirements for the relevant primitives from the literature (and refer to the full version [Fuc10] for more details).

Commitments. We will use a (non-interactive) *randomizable extractable commitment scheme* $\mathbf{Com}$ which is composed of the algorithms Setup, Com, RdCom, ExSetup, Extr, and WISetup. By $\mathcal{V}$ we denote the space of "committable" values, by $\mathcal{R}$ the randomness space and by $\mathcal{C}$ the space of commitments. On input the security parameter 1^λ, Setup and WISetup output a *commitment key* ck, and ExSetup outputs (ck, ek), where ck is distributed as the output of Setup; ek is called the *extraction key*. On input ck, a message $M \in \mathcal{V}$ and randomness $\rho \in \mathcal{R}$, Com outputs a commitment $\mathbf{c} \in \mathcal{C}$.

The scheme is *perfectly binding*, i.e., for any $ck \leftarrow$ Setup and any $\mathbf{c} \in \mathcal{C}$ there exists exactly one $M \in \mathcal{V}$ s.t. $\mathbf{c} = \mathsf{Com}(ck, M, \rho)$ for some ρ. If $(ck, ek) \leftarrow$ ExSetup then $\mathsf{Extr}(ek, \mathbf{c})$ extracts that value M from $\mathbf{c}$. The keys output by WISetup are computationally indistinguishable from those output by Setup and generate *perfectly hiding*

commitments: for any $ck^* \leftarrow \mathsf{WISetup}$, $\mathbf{c} \in \mathcal{C}$ and $M \in \mathcal{V}$, there exists a $\rho \in \mathcal{R}$ s.t. $\mathbf{c} = \mathsf{Com}(ck^*, M, \rho)$. Finally, we have $\mathsf{RdCom}(ck, \mathsf{Com}(ck, M, \rho), \rho') = \mathsf{Com}(ck, M, \rho + \rho')$; thus RdCom *randomizes* commitments[4].

Proofs for Committed Values. We define a proof system that allows one to prove that committed values satisfy an equation. The proofs are constructed from the committed values and the used randomness, and they are witness indistinguishable, which means they do not reveal which satisfying values were used. Given a proof for a set of commitments, the proof can be adapted to a randomization of the commitments without knowledge of the committed values.

A *randomizable witness-indistinguishable proof system* $\mathbf{Proof}$ for a commitment scheme $\mathbf{Com}$ for a class $\mathcal{E}$ of equations consists of the algorithms Prove, Verify and RdProof. On input *ck*, an equation $\mathrm{E} \in \mathcal{E}$, values $M_1, \ldots, M_n \in \mathcal{V}$ satisfying E and $\rho_1, \ldots, \rho_n \in \mathcal{R}$, Prove outputs a proof π for the values $\mathsf{Com}(ck, M_1, \rho_1), \ldots, \mathsf{Com}(ck, M_n, \rho_n)$. On input *ck*, E, $\mathbf{c}_1, \ldots, \mathbf{c}_n$ and π, Verify outputs 0 or 1, indicating rejection or acceptance of π. Every proof generated for commitments to values satisfying an equation is accepted by Verify. Given *ck*, $\mathbf{c}_1, \ldots, \mathbf{c}_n$, a proof π for $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$ and E, and $\rho'_1, \ldots, \rho'_n \in \mathcal{R}$, algorithm RdProof outputs a proof for the randomizations $\mathbf{c}'_i := \mathsf{RdCom}(ck, \mathbf{c}_i, \rho'_i)$; in particular, $\mathsf{RdProof}(ck, \mathrm{E}, (\mathbf{c}_1, \rho'_1), \ldots, (\mathbf{c}_n, \rho'_n), \pi)$ is distributed as $\mathsf{Prove}(ck, \mathrm{E}, (M_1, \rho_1 + \rho'_1), \ldots, (M_n, \rho_n + \rho'_n))$.

Soundness states that if there is a valid proof for a set of commitments for $\mathrm{E} \in \mathcal{E}$ then Extr extracts a set of values satisfying E. *Witness indistinguishability* is defined as follows: if the commitment key is output by WISetup then a set of commitments and a valid proof for them for an equation E reveals nothing, in an information-theoretical sense, about the committed values, except that they satisfy E.

Signatures. A signature scheme $\mathbf{Sig}$ consists of the following algorithms: $\mathsf{Setup}_\mathsf{S}$ takes as input the security parameter 1^λ and outputs parameters *pp*, which define a message space $\mathcal{M}$. On input *pp*, $\mathsf{KeyGen}_\mathsf{S}$ outputs a pair (vk, sk) of verification and signing key. For $M \in \mathcal{M}$, $\mathsf{Sign}(sk, M)$ outputs a signature Σ, which is verified by $\mathsf{Ver}(vk, M, \Sigma)$. If $pp \leftarrow \mathsf{Setup}_\mathsf{S}$ and $(vk, sk) \leftarrow \mathsf{KeyGen}_\mathsf{S}(pp)$ then $\mathsf{Ver}(vk, M, \mathsf{Sign}(sk, M)) = 1$ for all $M \in \mathcal{M}$. *Strong unforgeability* means that given *vk* and an oracle that queried on a message M_i returns a signature Σ_i on M_i, it is infeasible to output a pair (M, Σ), s.t. $\mathsf{Ver}(vk, M, \Sigma) = 1$ and $(M, \Sigma) \neq (M_i, \Sigma_i)$ for all i.

We require that $\mathbf{Sig}$ be *compatible* with $\mathbf{Com}$ and $\mathbf{Proof}$: the messages, verification keys and signatures are composed of values in $\mathcal{V}$ (the value space of $\mathbf{Com}$) and the signature verification predicate is a conjunction of equations from $\mathcal{E}$ (the class of equations for $\mathbf{Proof}$). We note that from a compatible triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ one can easily construct a *verifiably encrypted signature* scheme; see [Fuc10].

For our application to delegatable credentials we require furthermore that $\mathbf{Sig}$ be *automorphic*, that is, besides being compatible, its verification keys must lie in its message space $\mathcal{M}$. We let $\mathrm{E}_{\mathsf{Ver}}$ denote the verification equations for $\mathbf{Sig}$. When, for example, Σ is considered a variable we write $\mathrm{E}_{\mathsf{Ver}(vk, M, \cdot)}(\Sigma)$.

[4] Commitment schemes with two types of keys were called *perfectly hiding with extraction* in [GOS06] and *strongly computationally hiding* in [BCKL08]. Note that a scheme $\mathbf{Com}$ with the described properties is at the same time a *lossy encryption scheme* [BHY09].

3 Commuting Signatures and Verifiable Encryption

Commuting signatures extend a commitment scheme $\mathbf{Com}$, an associated proof system $\mathbf{Proof}$ and a compatible signature scheme $\mathbf{Sig}$ by the following functionalities: $\mathbf{Com}_{\mathcal{M}}$ is a commitment scheme with the same keys as $\mathbf{Com}$ and whose message space is that of $\mathbf{Sig}$. SigCom takes a $\mathbf{Com}_{\mathcal{M}}$ commitment and a signing key, and produces a commitment to a signature on the committed message and a proof of validity. SmSigCom simulates SigCom and is given a signature instead of the signing key. Moreover, the algorithms AdC and AdD *adapt proofs when (de)committing* to a signature (subscript $\mathcal{S}$), a message (subscript $\mathcal{M}$) or a verification key (subscript $\mathcal{K}$).

Definition 1. *A system of* ***commuting signatures and verifiable encryption*** *consists of an extractable commitment scheme* $\mathbf{Com} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{RdCom}, \mathsf{ExSetup}, \mathsf{Extr}, \mathsf{WISetup})$ *with value space* $\mathcal{V}$ *and randomness space* $\mathcal{R}$*, a randomizable WI proof system* $\mathbf{Proof} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{RdProof})$ *for* $\mathbf{Com}$*, a compatible signature scheme* $\mathbf{Sig} = (\mathsf{Setup}_{\mathrm{S}}, \mathsf{KeyGen}_{\mathrm{S}}, \mathsf{Sign}, \mathsf{Ver})$ *and the following algorithms. We let* $ck \leftarrow \mathsf{Setup}$*,* $pp_{\mathrm{S}} \leftarrow \mathsf{Setup}_{\mathrm{S}}$*,* $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(pp_{\mathrm{S}})$*,* $M \in \mathcal{M}$*,* $\mu \in \mathcal{R}_{\mathcal{M}}$ *and* $pp := (ck, pp_{\mathrm{S}})$*.*

$\mathbf{Com}_{\mathcal{M}}$*. On input pp, a message* $M \in \mathcal{M}$ *and* $\mu \in \mathcal{R}_{\mathcal{M}}$*, algorithm* $\mathsf{Com}_{\mathcal{M}}$ *outputs a commitment* $\mathbf{C}$ *in* $\mathcal{C}_{\mathcal{M}}$*, the space of commitments.* $\mathsf{RdCom}_{\mathcal{M}}$ *takes inputs pp,* $\mathbf{C}$ *and* $\mu' \in \mathcal{R}_{\mathcal{M}}$ *and outputs a randomized commitment* $\mathbf{C}'$*. On input ek output by* $\mathsf{ExSetup}$*, and* $\mathbf{C}$*,* $\mathsf{Extr}_{\mathcal{M}}$ *outputs the committed value* M*. We require* $\mathbf{Com}_{\mathcal{M}} := (\mathsf{Setup}, \mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}, \mathsf{ExSetup}, \mathsf{Extr}_{\mathcal{M}}, \mathsf{WISetup})$ *to be a commitment scheme as defined in Section 2 and to be* compatible *with* $\mathbf{Proof}$*, i.e.,* Prove *and* RdProof *accept inputs from* $\mathcal{R}_{\mathcal{M}}$ *and* Verify *accepts* $\mathbf{Com}_{\mathcal{M}}$ *commitments.*

$\mathsf{AdC}_{\mathcal{S}}(pp, vk, \mathbf{C}, (\Sigma, \rho), \bar{\pi})$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, \mathbf{C}, \bar{\pi}) = 1$ *then the algorithm outputs* π *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))]$*, where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdD}_{\mathcal{S}}(pp, vk, \mathbf{C}, (\Sigma, \rho), \pi)$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathsf{Com}(ck, \Sigma, \rho), \pi) = 1$ *then the algorithm outputs* $\bar{\pi}$ *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, (M, \mu))]$*, where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdC}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_{\Sigma}, \widetilde{\pi})$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_{\Sigma}, \widetilde{\pi}) = 1$ *then the algorithm outputs* π *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))]$*, where* Σ *and* ρ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdD}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_{\Sigma}, \pi)$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathsf{Com}_{\mathcal{M}}(pp, M, \mu), \mathbf{c}_{\Sigma}, \pi) = 1$*, the algorithm outputs* $\widetilde{\pi}$ *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, (\Sigma, \rho))]$*, where* Σ *and* ρ *are such that* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdC}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \pi)$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathbf{c}_{\Sigma}, \pi) = 1$*, the algorithm outputs* $\widehat{\pi}$ *which is distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, (vk, \xi), (M, \mu), (\Sigma, \rho)))]$*, where* M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{AdD}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \widehat{\pi})$*. If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, \mathsf{Com}(ck, vk, \xi), \mathbf{C}, \mathbf{c}_{\Sigma}, \widehat{\pi}) = 1$*, the algorithm outputs* π*, distributed as* $[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))]$*, where* M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_{\Sigma} = \mathsf{Com}(ck, \Sigma, \rho)$*.*

$\mathsf{SigCom}(pp, sk, \mathbf{C})$. *If* $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$ *then the algorithm outputs a commitment to a signature and a proof of validity* $(\mathbf{c}_\Sigma, \pi)$ *which is distributed as*

$$\left[\Sigma \leftarrow \mathsf{Sign}(sk, M);\ \rho \leftarrow \mathcal{R} :\ \left(\mathsf{Com}(ck, \Sigma, \rho), \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right)\right]$$

where M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$.

$\mathsf{SmSigCom}(pp, ek, vk, \mathbf{C}, \Sigma)$. *Assume* $(ck, ek) \leftarrow \mathsf{ExSetup}$. *If* $\mathsf{Ver}(vk, \mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}), \Sigma) = 1$ *then the algorithm outputs* $(\mathbf{c}_\Sigma, \pi)$ *which is distributed as* $[\rho \leftarrow \mathcal{R} : (\mathsf{Com}(ck, \Sigma, \rho), \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))]$, *where* M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$.

By $\mathbf{Algs} := (\mathsf{AdC}_{\mathcal{S}}, \mathsf{AdD}_{\mathcal{S}}, \mathsf{AdC}_{\mathcal{M}}, \mathsf{AdD}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{K}}, \mathsf{AdD}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom})$ we denote the algorithms of the system that extend $\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}$ and $\mathbf{Com}_{\mathcal{M}}$. When verifying a signature Σ on a message M by $\mathsf{Ver}(vk, M, \Sigma)$, we implicitly assume that Ver also checks whether $M \in \mathcal{M}$. Analogously, we assume that when verifying a proof of validity by running Verify on $\mathrm{E}_{\mathsf{Ver}}$ and $\mathbf{C}$, it checks whether $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$.

Definition 1 implies that running SigCom on a commitment to M yields the same (the output is distributed identically) as running $\Sigma \leftarrow \mathsf{Sign}(sk, M)$, $\mathsf{Com}_{\mathcal{M}}$ on M, Com on Σ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$; or running Sign, $\mathsf{Com}_{\mathcal{M}}$ on M and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$, and then Com on Σ and $\mathsf{AdC}_{\mathcal{S}}$, etc. This means that the diagram in Figure 1 *commutes*.

SmSigCom allows us to prove the following unforgeability property: Consider an adversary that is given (pp, ek, vk) for $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathsf{S}}$ and access to an oracle that on input $\mathbf{C}_i$ returns $(\mathbf{c}_i, \pi_i) \leftarrow \mathsf{SigCom}(pp, sk, \mathbf{C}_i)$. Then it cannot output a valid triple $(\mathbf{C}, \mathbf{c}_\Sigma, \pi)$ such that the committed message/signature pair is different from every pair committed in $(\mathbf{C}_i, \mathbf{c}_i)$ from the oracle calls. Commuting signatures moreover yield a round-optimal blind signature scheme: The user sends a commitment to the message to the signer, who runs SigCom to produce $(\mathbf{c}_\Sigma, \pi)$. The user computes a proof $\widetilde{\pi}$ for $\mathbf{c}_\Sigma$ and M via $\mathsf{AdD}_{\mathcal{M}}$, and outputs $(\mathbf{c}_\Sigma, \widetilde{\pi})$ as the blind signature (see [Fuc10]).

4 Instantiation of the Building Blocks

4.1 Bilinear Groups and Assumptions

A *bilinear group* is a tuple $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p; G_1 and G_2 generate $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively; and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear map: $\forall X \in \mathbb{G}_1\ \forall Y \in \mathbb{G}_2\ \forall a, b \in \mathbb{Z} : e(X^a, Y^b) = e(X, Y)^{ab}$, and $e(G_1, G_2)$ generates $\mathbb{G}_T$.

We denote group elements by capital letters and assume two fixed generators G and H of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. We call a pair $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$ a *Diffie-Hellman pair* (w.r.t. (G, H)), if there exists $a \in \mathbb{Z}_p$ such that $A = G^a$ and $B = H^a$. We let $\mathcal{DH} := \{(G^a, H^a) \mid a \in \mathbb{Z}_p\}$ denote the set of DH pairs. Using the bilinear map e, $\mathcal{DH}$ is efficiently decidable by checking $e(G^{-1}, \underline{B})\, e(\underline{A}, H) = 1$. We will make the following assumptions.

Assumption 1 (SXDH). *The* Symmetric External Diffie-Hellman assumption *for* $\mathcal{G}$ *states that given* $(G_1, G_1^r, G_1^s, G_1^t)$ *for random* $r, s \in \mathbb{Z}_p$, *it is hard to decide whether* $t = rs$ *or* t *is random; likewise, given* $(G_2, G_2^{r'}, G_2^{s'}, G_2^{t'})$ *for random* $r', s' \in \mathbb{Z}_p$, *it is hard to decide whether* $t' = r's'$ *or* t' *is random.*

The *q-Asymmetric Double Hidden Strong Diffie-Hellman assumption* (ADH-SDH) was introduced in [AFG+10] and proven to hold in the generic-group model for any type of pairing. It was shown in [FPV09] that under the q-SDH assumption [BB04], given $q-1$ tuples $((K\cdot G^{v_i})^{1/(x+c_i)}, c_i, v_i)$ for random $c_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to produce a new tuple of this form. Similarly to q-HSDH [BW07], ADH-SDH states that if c_i and v_i are given in a *hidden* form $(F^{c_i}, H^{c_i}, G^{v_i}, H^{v_i})$, it is intractable to produce another such tuple $((K\cdot G^{v})^{1/(x+c)}, F^{c}, H^{c}, G^{v}, H^{v})$.

Assumption 2 (q-ADH-SDH). *For randomly chosen $G, F, K \leftarrow \mathbb{G}_1, H \leftarrow \mathbb{G}_2$ and $x, c_i, v_i \leftarrow \mathbb{Z}_p$, given $(G, F, K, X=G^x;\ H, Y=H^x)$ and, for $1 \le i \le q-1$,*

$$\left(A_i = (K\cdot G^{v_i})^{\frac{1}{x+c_i}},\ B_i = F^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right)\ ,$$

it is hard to output $((K\cdot G^{v})^{1/(x+c)}, F^{c}, H^{c}, G^{v}, H^{v})$ with $(c, v) \neq (c_i, v_i)$ for all i.

The next assumption is a *weak* variant of the various *flexible CDH* assumptions, generalized to *asymmetric* bilinear groups. It was also introduced in [AFG+10] and shown to be implied by DDH in $\mathbb{G}_1$, and thus by SXDH.

Assumption 3 (AWF-CDH). *Given random generators $G \leftarrow \mathbb{G}_1$ and $H \leftarrow \mathbb{G}_2$, and $A = G^a$ for $a \leftarrow \mathbb{Z}_p$, it is hard to output $(G^r, G^{ra}, H^r, H^{ra})$ with $r \neq 0$.*

4.2 Groth-Sahai Proofs and Automorphic Signatures

Commitments. We instantiate $\mathbf{Com}$, defined in Section 2, by the commitment scheme based on SXDH from [GS08]. Setup, on input $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$, outputs a commitment key $ck \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. Value and randomness space are defined as $\mathcal{V} := \mathbb{G}_1 \cup \mathbb{G}_2$ and $\mathcal{R} := \mathbb{Z}_p^2$. $\mathsf{Com}(ck, X, r)$ takes randomness $r \in \mathcal{R}$ and an element $X \in \mathcal{V}$; commitments to $\mathbb{G}_1$-elements are in $\mathbb{G}_1^2$ and commitments to $\mathbb{G}_2$-elements are in $\mathbb{G}_2^2$. We have $\mathsf{Com}(ck, X, r) \circ \mathsf{Com}(ck, X', r') = \mathsf{Com}(ck, X\cdot X', r+r')$, where "$\circ$" denotes component-wise multiplication; the commitments are thus *homomorphic*.

$\mathsf{RdCom}(ck, \mathbf{c}, r')$ returns $\mathbf{c}' := \mathbf{c} \circ \mathsf{Com}(ck, 1, r')$, which for $\mathbf{c} = \mathsf{Com}(ck, X, r)$ is $\mathbf{c}' = \mathsf{Com}(ck, X, r+r')$. $\mathsf{ExSetup}$ constructs ck as in Setup and in addition outputs the used randomness as ek, and $\mathsf{Extr}(ek, \mathbf{c})$ outputs the committed value. $\mathsf{WISetup}$ produces a commitment key ck^* that is indistinguishable from outputs of Setup under the SXDH assumption. Commitments under ck^* are independent of the committed value.

Proofs for Committed Values. In order to instantiate $\mathbf{Proof}$ for $\mathbf{Com}$, we use the proof system from [GS08], which was shown to be randomizable in [BCC+09]. The class of equations $\mathcal{E}$ for our proof system are *pairing-product equations* (PPE). A PPE over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$ is an equation of the form[5]

$$\mathrm{E}(X_1, \ldots, X_m; Y_1, \ldots, Y_n)\ :\ \prod_{j=1}^{n} e(A_j, \underline{Y_j}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} = \mathbf{t}_T\ , \quad (1)$$

[5] For a more concise exposition we will underline the variables of an equation.

defined by $A_j \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $\gamma_{i,j} \in \mathbb{Z}_p$, for $1 \leq i \leq m$ and $1 \leq j \leq n$, and $\mathbf{t}_T \in \mathbb{G}_T$. We refer to [GS08] or [Fuc10] for a description of the implementations of the following: Prove, which chooses internal randomness $Z \leftarrow \mathbb{Z}_p^{2\times 2}$, and outputs $\pi \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ (we write $\mathsf{Prove}(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n; Z)$ if we want to make Z explicit); $\mathsf{RdProof}(ck, \mathrm{E}, (\mathbf{c}_i, r_i)_{i=1}^m, (\mathbf{d}_j, s_j)_{j=1}^n, \pi)$, which adapts a proof π to the new commitments output by $\mathsf{RdCom}(ck, \mathbf{c}_i, r_i)$ and $\mathsf{RdCom}(ck, \mathbf{d}_j, s_j)$; and $\mathsf{Verify}(ck, \mathrm{E}, \vec{\mathbf{c}}, \vec{\mathbf{d}}, \pi)$.

Signatures. We instantiate **Sig** with the automorphic signature scheme presented in [AFG+10]. It is compatible, as signature components are in the space for committed values $\mathcal{V} = \mathbb{G}_1 \cup \mathbb{G}_2$, and the verification equations are pairing-product equations, thus in $\mathcal{E}$. Moreover, the verification keys lie in its message space, the set of Diffie-Hellman pairs. Under q-ADH-SDH and AWF-CDH (which is implied by SXDH), **Sig** is strongly unforgeable against adversaries making up to $q-1$ adaptive chosen-message queries, as shown in [Fuc09].

Scheme 1 (Sig). $\mathsf{Setup_S}$ has input a bilinear group $\mathcal{G}$ and outputs random generators $F, K, T \leftarrow \mathbb{G}_1$. The message space is $\mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$.

$\mathsf{KeyGen_S}$ chooses $x \leftarrow \mathbb{Z}_p$ and outputs $vk = (G^x, H^x)$ and $sk = x$.

Sign has input a secret key x and a message $(M, N) \in \mathcal{DH}$. It chooses $c, r \leftarrow \mathbb{Z}_p$ and outputs

$$\left(A := (K \cdot T^r \cdot M)^{\frac{1}{x+c}},\ B := F^c,\ D := H^c,\ R := G^r,\ S := H^r\right) .$$

Ver on input a verification key $(X, Y) \in \mathcal{DH}$, a message $(M, N) \in \mathcal{DH}$ and a signature (A, B, D, R, S) outputs 1 if and only if the following equalities hold:

$$e(A, Y \cdot D) = e(K \cdot M, H)\, e(T, S) \qquad \begin{aligned} e(B, H) &= e(F, D) \\ e(R, H) &= e(G, S) \end{aligned} \tag{2}$$

Under SXDH and ADH-SDH, $\mathbf{Com}, \mathbf{Proof}$ and **Sig** are instantiations of the primitives defined in Section 2. Note that if we based GS proofs on DLIN instead of SXDH, the security of our constructions would follow from DLIN, ADH-SDH and AWF-CDH, which can all be made for bilinear groups of every type (1, 2 and 3) from [GPS08].

5 Additional Properties of Groth-Sahai Proofs

We identify four properties of Groth-Sahai (GS) proofs which will allow us to instantiate commuting signatures. We refer to [Fuc10] for the proofs and further results. First, proofs are independent of the right-hand side of the equation, and if the equation does not contain pairings of two variables, i.e., $\gamma_{ij} = 0$ for all i, j in (1), then they are even independent of the committed values.

Lemma 1. *For any equation* $\mathrm{E} \in \mathcal{E}$ *the output of* $\mathsf{Prove}(\cdot, \mathrm{E}, \cdot, \cdot)$ *is independent of* $\mathbf{t}_T$.

Lemma 2. *Proofs for equations for which* $\gamma_{ij} = 0$ *for all* i, j *depend only on the randomness of the commitments.*

Groth-Sahai (GS) proofs are homomorphic w.r.t. the equations, in that the product of two proofs is a proof for the "product of the respective equations". More precisely, given two equations

$$\mathrm{E} \;:\; \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \mathbf{t}_T$$
$$\mathrm{E}' \;:\; \prod_{i=1}^{n'} e(A'_i, \underline{Y'_i}) \prod_{i=1}^{m'} e(\underline{X'_i}, B'_i) \prod_{i=1}^{m'} \prod_{j=1}^{n'} e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathbf{t}'_T$$

and a proof π for commitments $(\vec{\mathbf{c}}, \vec{\mathbf{d}})$ for E and a proof π' for commitments $(\vec{\mathbf{c}}', \vec{\mathbf{d}}')$ for E′, then $\pi'' := \pi \circ \pi'$ is a proof for commitments $((\vec{\mathbf{c}}, \vec{\mathbf{c}}'), (\vec{\mathbf{d}}, \vec{\mathbf{d}}'))$ and equation E″ defined as $\prod e(A_i, \underline{Y_i}) \prod e(A'_i, \underline{Y'_i}) \prod e(\underline{X_i}, B_i) \prod e(\underline{X'_i}, B'_i) \prod\prod e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \cdot \prod\prod e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathbf{t}''_T$ (for arbitrary $\mathbf{t}''_T \in \mathbb{G}_T$).

Lemma 3. *For* E, E′ *and* E″ *as above, if* $\pi = \mathsf{Prove}(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n; Z)$ *and* $\pi' = \mathsf{Prove}(ck, \mathrm{E}', (X'_i, r'_i)_{i=1}^{m'}, (Y'_j, s'_j)_{j=1}^{n'};\ Z')$ *then the following equation holds:* $\pi \circ \pi' = \mathsf{Prove}(ck, \mathrm{E}'', (X_i, r_i)_{i=1}^m, (X'_i, r'_i)_{i=1}^{m'}, (Y_j, s_j)_{j=1}^n, (Y'_j, s'_j)_{j=1}^{n'};\ Z + Z')$.

Given a proof for an equation, one can commit to its constants and adapt the proof. Consider $\mathrm{E}(X_1, \ldots, X_m;\ Y_1, \ldots, Y_n)$ as in (1) and a proof π for E and commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m;\ \mathbf{d}_1, \ldots, \mathbf{d}_n)$. Then π is also a proof for $\mathrm{E}'(\vec{X}, A_k;\ \vec{Y})$ defined as

$$\prod_{\substack{i=1 \\ i \neq k}}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}}\, e(\underline{A_k}, \underline{Y_k}) \;=\; \mathbf{t}_T$$

and commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathsf{Com}(ck, A_k, 0);\ \mathbf{d}_1, \ldots, \mathbf{d}_n)$. We have thus:

Lemma 4. *Let* $\pi \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$, $\mathbf{c}_i = \mathsf{Com}(ck, X_i, r_i)$ *and* $\mathbf{d}_j = \mathsf{Com}(ck, Y_j, s_j)$ *for all* i, j. *Then* $\mathsf{RdProof}\big(ck, \mathrm{E}', (\mathbf{c}_i, 0)_{i=1}^m, (\mathsf{Com}(ck, A_k, 0), r), (\mathbf{d}_j, 0)_{j=1}^n, \pi\big)$ *yields a proof that is distributed as* $\mathsf{Prove}\big(ck, \mathrm{E}', (X_i, r_i)_{i=1}^m, (A_k, r), (Y_j, s_j)_{j=1}^n\big)$. *An analogous result holds for committing to a constant* $B_k \in \mathbb{G}_2$.

6 Instantiation of Commuting Signatures

We explain how to implement commuting signatures; due to space constraints we refer to [Fuc10] for more details and the proofs. In [Fuc09, AFG+10], a blind signature scheme is constructed from the scheme **Sig** (Scheme 1) as follows. Given parameters (G, H, F, K, T) and a message $(M, N) \in \mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$, the user chooses a random $t \leftarrow \mathbb{Z}_p$ and *blinds* the first message component by the factor T^t. He then sends the following to the signer: $U := T^t \cdot M$, commitments $\mathbf{c}_M, \mathbf{c}_N, \mathbf{c}_P$ and $\mathbf{c}_Q$ to $M, N, P := G^t$ and $Q := H^t$, respectively; and proofs π_M, π_P and π_U proving $(M, N), (P, Q) \in \mathcal{DH}$ and well-formedness of U. With

$$\mathrm{E}_{\mathcal{DH}}(M, N) \;:\; e(G^{-1}, \underline{N})\, e(\underline{M}, H) = 1 \tag{3}$$
$$\mathrm{E}_U(M, Q) \;:\; e(T^{-1}, \underline{Q})\, e(\underline{M}, H^{-1}) = e(U, H)^{-1} \tag{4}$$

π_M proves $\mathrm{E}_{\mathcal{DH}}(M, N)$, π_P proves $\mathrm{E}_{\mathcal{DH}}(P, Q)$, and π_U proves $\mathrm{E}_U(M, Q)$, which asserts $U = T^t \cdot M$.

The signer replies with a "pre-signature" on U, defined in (6) below, which the user converts into a signature and outputs a GS proof of knowledge of it. Now to turn this into a commuting signature, there are two key observations.

1. The values $\mathbf{C} := (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ the user sends to the signer can be considered as a *commitment* to the message (M, N), which is extractable and randomizable, and which perfectly hides the message when the values are produced under a key $ck^* \leftarrow \mathsf{WISetup}$.
2. As $\mathbf{Com}$ is homomorphic, the values $\mathbf{c}_P$ and $\mathbf{c}_Q$ contained in the commitment $\mathbf{C}$ to (M, N) can be used by the signer to compute commitments to an actual signature. Moreover, below we show how π_P and π_U can be used to make a proof of validity using Lemmas 1, 2, 3 and 4.

For the blind signature scheme in [Fuc09], the values $\mathbf{c}_P, \mathbf{c}_Q, \pi_P$ and π_U are mainly needed in the proof of unforgeability, when the simulator extracts the message, queries it to its signing oracle and then uses P and Q to turn the signature into a pre-signature. We show that these values can be directly used by the signer to produce commitments to the signature components and even a proof of validity.

Commitments to Messages. To instantiate $\mathbf{Com}_{\mathcal{M}}$, we define a commitment to a message $(M, N) \in \mathcal{DH}$ as $\mathbf{C}$ discussed above. For parameters $pp = (\mathcal{G}, ck, F, K, T)$ the space of valid commitments is thus defined as

$$\mathcal{C}_{\mathcal{M}}(pp) := \big\{(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \;\big|\; \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_M, \mathbf{c}_N, \pi_M) \\ \wedge\ \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_P, \mathbf{c}_Q, \pi_P) \wedge \mathsf{Verify}(ck, \mathrm{E}_U, \mathbf{c}_M, \mathbf{c}_Q, \pi_U)\big\}\ ,$$

and the randomness space is $\mathcal{R}_{\mathcal{M}} := \mathbb{Z}_p \times \mathcal{R}^4$. The algorithms $\mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}$ and $\mathsf{Extr}_{\mathcal{M}}$ defining $\mathbf{Com}_{\mathcal{M}}$ are given in Figure 2.

Committing to a Signature on a Committed Message and Proving Validity. We now show how the signer can use the values in $\mathbf{C}$ to produce a proof of knowledge $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$ of a signature (A, B, D, R, S) (i.e., a verifiably encrypted signature) on the message committed in $\mathbf{C}$. The proofs π_A, π_B and π_R attest that the values committed in $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ and $\mathbf{c}_M$ contained in $\mathbf{C}$ satisfy the verification equations in (2):

$$\begin{aligned} \mathrm{E}_A(A, M;\, S, D) \;&:\; e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\ \mathrm{E}_B(B;\, D) \;&:\; e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1 \\ \mathrm{E}_R(R;\, S) \;&:\; e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1 \end{aligned} \tag{5}$$

In the blind signature scheme, after receiving $\mathbf{C}$, the signer checks the proofs contained in it, and then produces a pre-signature, which is constructed as a signature on U, but on a message that lacks the second component: choose $c, r \leftarrow \mathbb{Z}_p$ and compute

$$A := (K \cdot T^r \cdot U)^{1/(x+c)} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r \tag{6}$$

Since $U := T^t \cdot M$, we have $A = (K \cdot T^r \cdot U)^{1/(x+c)} = (K \cdot T^{r+t} \cdot M)^{1/(x+c)}$, which is the first component of a signature on (M, N) with randomness $r + t$. Knowing t,

$\mathbf{Com}_{\mathcal{M}}$ on input pp, $(M,N) \in \mathcal{DH}$, $(t,\mu,\nu,\rho,\sigma) \in \mathcal{R}_{\mathcal{M}}$ sets $P = G^t$, $Q = H^t$ and returns

$$\begin{array}{lll} \mathbf{c}_M := \mathsf{Com}(ck, M, \mu) & \mathbf{c}_N := \mathsf{Com}(ck, N, \nu) & \pi_M \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}},\ (M,\mu),(N,\nu)) \\ \mathbf{c}_P := \mathsf{Com}(ck, P, \rho) & \mathbf{c}_Q := \mathsf{Com}(ck, Q, \sigma) & \pi_P \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}},\ (P,\rho),(Q,\sigma)) \\ U := T^t \cdot M & & \pi_U \leftarrow \mathsf{Prove}(ck, \mathrm{E}_U, (M,\mu),(Q,\sigma)) \end{array}$$

$\mathbf{RdCom}_{\mathcal{M}}$ has input pp, $\mathbf{C}$ and $(t',\mu',\nu',\rho',\sigma') \in \mathcal{R}_{\mathcal{M}}$, and returns $\mathbf{C}'$. It replaces t by $t+t'$ setting $U' := U \cdot T^{t'}$, $\hat{\mathbf{c}}_P := \mathbf{c}_P \circ \mathsf{Com}(ck, G^{t'}, 0)$, and $\hat{\mathbf{c}}_Q := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^{t'}, 0)$. It then replaces the remaining randomness (μ,ν,ρ,σ) by $(\mu+\mu', \nu+\nu', \rho+\rho', \sigma+\sigma')$, setting

$$\begin{array}{ll} \mathbf{c}'_M := \mathsf{RdCom}(ck, \mathbf{c}_M, \mu') & \pi'_M \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\mathbf{c}_M, \mu'), (\mathbf{c}_N, \nu'), \pi_M) \\ \mathbf{c}'_N := \mathsf{RdCom}(ck, \mathbf{c}_N, \nu') & \\ \mathbf{c}'_P := \mathsf{RdCom}(ck, \hat{\mathbf{c}}_P, \rho') & \pi'_P \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\hat{\mathbf{c}}_P, \rho'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_P) \\ \mathbf{c}'_Q := \mathsf{RdCom}(ck, \hat{\mathbf{c}}_Q, \sigma') & \pi'_U \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_U, (\mathbf{c}_M, \mu'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_U) \end{array}$$

$\mathbf{Extr}_{\mathcal{M}}(ek, \mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U))$ outputs $(\mathsf{Extr}(ek, \mathbf{c}_M), \mathsf{Extr}(ek, \mathbf{c}_N))$.

$\mathbf{SigCom}(pp, sk, \mathbf{C})$. Parse $\mathbf{C}$ as $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and sk as x. If π_M, π_P and π_U are valid then choose $c, r \leftarrow \mathbb{Z}_p$ and $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathbb{Z}_p^2$ and compute the following values:

$$\begin{array}{lll} A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} & \mathbf{c}_B := \mathsf{Com}(ck, F^c, \beta) & \mathbf{c}_R := \mathbf{c}_P \circ \mathsf{Com}(ck, G^r, \rho') \\ \mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) & \mathbf{c}_D := \mathsf{Com}(ck, H^c, \delta) & \mathbf{c}_S := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^r, \sigma') \\ \pi'_A := \pi_U \circ \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A,\alpha), (H^c,\delta);\ 0) & & \text{(with } \mathrm{E}_{A^\dagger} \text{ being Equation (9))} \\ \pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_A, (\mathbf{c}_A, 0), (\mathbf{c}_D, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, \sigma'), \pi'_A) & & \\ \pi_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R, \rho'), (\mathbf{c}_S, \sigma'), \pi_P) & \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (F^c, \beta), (H^c, \delta)) & \end{array}$$

Return $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$.

Fig. 2. Committing to messages and making commitments to a signature on a committed value

the user can fabricate an actual signature on (M,N) from the pre-signature by setting $R := R' \cdot G^t = G^{r+t}$ and $S := S' \cdot H^t = H^{r+t}$. (A,B,D,R,S) is then a signature on (M,N) with randomness $(c, r+t)$.

Let μ, ρ and σ denote the randomness of the respective commitments $\mathbf{c}_M, \mathbf{c}_P$ and $\mathbf{c}_Q$, contained in $\mathbf{C}$. Since the commitments are homomorphic, the signer can—without knowing $P = G^t$ and $Q = H^t$—compute commitments to R and S from $\mathbf{c}_P$ and $\mathbf{c}_Q$:

$$\begin{aligned} \mathbf{c}_R := \mathsf{Com}(ck, R', 0) \circ \mathbf{c}_P = \mathsf{Com}(ck, R, \rho) \\ \mathbf{c}_S := \mathsf{Com}(ck, S', 0) \circ \mathbf{c}_Q = \mathsf{Com}(ck, S, \sigma) \end{aligned} \tag{7}$$

The signer then chooses $\alpha, \beta, \delta \leftarrow \mathcal{R}$, and makes the remaining commitments:

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_B := \mathsf{Com}(ck, B, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, D, \delta) \tag{8}$$

The vector $\vec{\mathbf{c}}_\Sigma := (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ is thus a commitment to the signature $\Sigma = (A,B,D,R,S)$ on (M,N). It remains to construct proofs π_A, π_B and π_R for the 3

equations in (5)—without knowledge of the randomness μ, ρ and σ of the commitments $\mathbf{c}_M, \mathbf{c}_R$ and $\mathbf{c}_S$! This can be done by observing the following:

1. Equation $\mathrm{E}_R(R;\ S)$ is actually $\mathrm{E}_{\mathcal{DH}}(R;\ S)$ from (3). Since by (7) $\mathbf{c}_R$ and $\mathbf{c}_P$ have the same randomness ρ, and $\mathbf{c}_S$ and $\mathbf{c}_Q$ have the same randomness σ, and since by Lemma 2 proofs for $\mathrm{E}_{\mathcal{DH}}$ are independent of the committed values, π_P (the proof for $\mathbf{c}_P$ and $\mathbf{c}_Q$ for $\mathrm{E}_{\mathcal{DH}}$) is also a proof for $\mathbf{c}_R$ and $\mathbf{c}_S$; we thus set $\pi_R := \pi_P$.
2. Lemmas 1 and 2 yield that proofs for E_U (Equation (4)) only depend on the randomness of the commitments. Since $\mathbf{c}_S = \mathsf{Com}(ck, S, \sigma)$ and $\mathbf{c}_Q = \mathsf{Com}(ck, Q, \sigma)$ have the same randomness, π_U is not only a proof for $\mathrm{E}_U(M; Q)$ but also for

$$\mathrm{E}_{U^\dagger}(M;\ S)\ :\ e(T^{-1}, \underline{S})\, e(\underline{M}, H^{-1}) = \mathbf{t}_T$$

(for an arbitrary $\mathbf{t}_T \in \mathbb{G}_T$)[6] for $\mathbf{c}_M$ and $\mathbf{c}_S$. Moreover, the signer, knowing A, D, α and δ, can produce a proof $\pi_{A^\dagger} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (D, \delta))$ for

$$\mathrm{E}_{A^\dagger}(A;\ D)\ :\ e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = \mathbf{t}_T \tag{9}$$

(for any $\mathbf{t}_T$). Since the product of the left-hand sides of $\mathrm{E}_{U^\dagger}(M; S)$ and $\mathrm{E}_{A^\dagger}(A; D)$ is the left-hand side of $\mathrm{E}_A(A, M;\ S, D)$ from (5), Lemma 3 yields that $\pi_A := \pi_U \circ \pi_{A^\dagger}$ is a proof for E_A.

We have thus shown how the signer can construct π_A and π_R. The remaining proof π_B can be made regularly, since the required randomness (β, δ) was chosen by the signer. Finally, to get a *random* proof of knowledge, the signer randomizes all commitments and proofs using RdCom and $\mathsf{RdProof}$ as defined in Section 4.2. Algorithm SigCom, with some optimizations, is summarized in Figure 2.

Instantiation of SmSigCom. This algorithm is similar to SigCom but instead of the signing key sk it is given a signature (A, B, D, R, S) and the extraction key. It proceeds like SigCom but starting from a signature instead of producing a pre-signature: choose $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathcal{R}$ and set $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ as in (8); use ek to extract P and Q from $\mathbf{C}$ and set $\mathbf{c}_R := \mathbf{c}_P \circ \mathsf{Com}(ck, R \cdot P^{-1}, \rho') = \mathsf{Com}(ck, R, \rho + \rho')$ and $\mathbf{c}_S := \mathbf{c}_Q \circ \mathsf{Com}(ck, S \cdot Q^{-1}, \sigma') = \mathsf{Com}(ck, S, \sigma + \sigma')$. Now π_A, π_B and π_R can be computed as in SigCom in Figure 2.

Instantiations of Proof Adaptation for Committing and Decommitting. We define equations $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ and recall E_A, which all represent the first verification equation in (2) but with different elements being variables.

$$\begin{aligned}
\mathrm{E}_A(A, M;\ S, D)\ &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H)\\
\mathrm{E}_{\tilde{A}}(A;\ S, D)\ &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = e(K \cdot M, H)\\
\mathrm{E}_{\bar{A}}(M)\ &:\ e(\underline{M}, H^{-1}) = e(A, Y \cdot D)^{-1} e(K, H)\, e(T, S)
\end{aligned}$$

[6] Technically, π_U is only a *proof* for $\mathrm{E}_{U^\dagger}$ when $\mathbf{t}_T$ is s.t. M and S satisfy it. However, since the proofs are independent of the right-hand side, the prover need not know the appropriate $\mathbf{t}_T$.

With E_B and E_R defined in (5) we can write the following

$$\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,\cdot)}((M,N),(A,B,D,R,S)) \equiv \mathrm{E}_A(A,M;\ S,D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \quad (10)$$

$$\mathrm{E}_{\mathsf{Ver}((X,Y),(M,N),\cdot)}(A,B,D,R,S) \equiv \mathrm{E}_{\tilde{A}}(A;\ S,D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \quad (11)$$

$$\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,(A,B,D,R,S))}(M,N) \equiv \mathrm{E}_{\bar{A}}(M) \quad (12)$$

$\mathsf{AdC}_{\mathcal{S}}$ transforms proofs $\bar{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$ (12) into proofs π for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$ (10), $\mathsf{AdC}_{\mathcal{M}}$ transforms proofs $\tilde{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,(M,N),\cdot)}$ (11) into proofs π for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$, whereas $\mathsf{AdD}_{\mathcal{S}}$ and $\mathsf{AdD}_{\mathcal{M}}$ do the converse.

Since the product of the left-hand sides of $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ is the left-hand side of E_A, by Lemma 3 we have $\pi_A = \pi_{\tilde{A}} \circ \pi_{\bar{A}}$, which lets us transform proofs for equations $\mathrm{E}_{\tilde{A}}$ and $\mathrm{E}_{\bar{A}}$ into proofs for E_A and vice versa, and thus implement the four algorithms. Note that when a proof is multiplied by a freshly generated proof, it is uniformly distributed: by Lemma 3, if Z is the internal randomness of the proof and Z' that of the fresh proof then the randomness of the product of proofs is $Z + Z'$. However, when reusing a proof it must be randomized first.

$\mathsf{AdC}_{\mathcal{S}}(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \bar{\pi})$. Proof $\bar{\pi}$ being for (12), it sets

$$\pi_{\tilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\tilde{A}}, (A,\alpha),(S,\sigma),(D,\delta)) \qquad \begin{array}{l} \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_B, (B,\beta),(D,\delta)) \\ \pi_R \leftarrow \mathsf{Prove}(ck, \mathrm{E}_R, (R,\rho),(S,\sigma)) \end{array}$$

for E_B and E_R defined in (5). It then returns $\pi := (\pi_{\tilde{A}} \circ \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdD}_{\mathcal{S}}(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \pi)$. The proof π is of the form (π_A, π_B, π_R). The algorithm sets $\pi_{\tilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\tilde{A}}, (A,\alpha),(S,\sigma),(D,\delta))$ and returns $\bar{\pi} := \pi_A \oslash \pi_{\tilde{A}}$ (where "$\oslash$" denotes componentwise division, i.e., multiplying every component of π_A with the inverse of that component of $\pi_{\tilde{A}}$).

$\mathsf{AdC}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \tilde{\pi})$. The proof $\tilde{\pi}$ is of the form $(\pi_{\tilde{A}}, \pi_B, \pi_R)$. The algorithm returns $\pi := (\pi_{\tilde{A}} \circ \pi_{\bar{A}}, \pi'_B, \pi'_R)$ with

$$\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu)) \qquad \begin{array}{l} \pi'_B \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_B, (\mathbf{c}_B,0),(\mathbf{c}_D,0),\pi_B) \\ \pi'_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R,0),(\mathbf{c}_S,0),\pi_R) \end{array}$$

$\mathsf{AdD}_{\mathcal{M}}(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \pi)$. The proof π is of the form (π_A, π_B, π_R). The algorithm computes $\pi_{\bar{A}}, \pi'_B$ and π'_R as for $\mathsf{AdC}_{\mathcal{M}}$ above, and returns $\tilde{\pi} := (\pi_A \oslash \pi_{\bar{A}}, \pi'_B, \pi'_R)$.

Instantiation of $\mathsf{AdC}_{\mathcal{K}}$ and $\mathsf{AdD}_{\mathcal{K}}$. A commitment $\mathbf{c}_{vk}$ to $vk = (X,Y) \in \mathcal{DH}$ is defined as $(\mathsf{Com}(ck,X,\xi), \mathsf{Com}(ck,Y,\psi), \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X,\xi),(Y,\psi)))$. The equations for $\mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}((X,Y),(M,N),(A,B,D,R,S))$, i.e., when the key, the message and the signature are committed, are represented by

$$\mathrm{E}_{\widehat{A}}(A,M;\ S,Y,D)\ :\ e(T^{-1},\underline{S})\, e(\underline{M},H^{-1})\, e(\underline{A},\underline{Y}) e(\underline{A},\underline{D}) = e(K,H)\ ,$$

and E_B and E_R from (5). Given a commitment $\mathbf{C}$ to a message, a commitment $\mathbf{c}_\Sigma = (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ to a signature, (X, Y) and a proof $\pi = (\pi_A, \pi_B, \pi_R)$ of validity, by Lemma 4 the component π_A can be adapted to $\pi_{\widehat{A}}$ for $\mathbf{c}_Y = \mathsf{Com}(ck, Y, \psi)$ setting

$$\pi_{\widehat{A}} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), \psi), (\mathbf{c}_D, 0), \pi_A\big) \ .$$

To adapt a proof to a decommitment of $\mathbf{c}_{vk}$, we have to reset the randomness of $\mathbf{c}_Y$ to 0. $\mathsf{AdD}_{\mathcal{K}}$ does thus the converse: it sets

$$\pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), -\psi), (\mathbf{c}_D, 0), \pi_{\widehat{A}}) \ .$$

We conclude this section by summarizing the results by the following theorem.

Theorem 1. *Under the ADH-SDH and the SXDH assumption,* $\big(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}, \mathbf{Com}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{S}}, \mathsf{AdD}_{\mathcal{S}}, \mathsf{AdC}_{\mathcal{M}}, \mathsf{AdD}_{\mathcal{M}}, \mathsf{AdC}_{\mathcal{K}}, \mathsf{AdD}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom}\big)$ *is a system of commuting signatures and verifiable encryption as defined in Definition 1.*

7 Non-interactively Delegatable Anonymous Credentials

7.1 The BCCKLS Model

Functionality. We give an overview of the model for delegatable credentials defined in [BCC+09]. The system parameters are set up by a trusted party. Every user generates a secret key *sk*, of which they can publish *pseudonyms Nym*. Any user can become *originator* of a credential by publishing a pseudonym Nym_O as the *public key*. To issue or delegate a credential, the issuer and the user (both known to each other under their respective pseudonyms) run a protocol at the end of which the user holds a credential. The holder can produce a *credential proof* for any of his pseudonyms, which proves that the owner of that pseudonym holds a credential rooted at a public key Nym_O.

In our *non-interactive* instantiation we have the following: To delegate (or to issue) a credential to a user known to the delegator under Nym_U, the delegator produces (without interacting with the user) a ready credential proof for Nym_U. The user can turn this credential proof into a credential, which (as in the BCCKLS model) she can then use to make a credential proof for another pseudonym.

A (non-interactively) delegatable anonymous credential system consists of the following algorithms. On input 1^λ, $\mathsf{Setup}_{\mathrm{C}}$ generates the parameters *pp*, which are input to all other algorithms. $\mathsf{KeyGen}_{\mathrm{C}}$ generates user secret keys *sk*, of which NymGen outputs pseudonyms *Nym* and auxiliary information *aux* related to *Nym*.

Issuing and delegation is done via Issue, which on input the issuer's secret key sk_I, pseudonym Nym_I and corresponding aux_I, a level-L credential *cred* for the issuer rooted at Nym_O (if $L = 0$ then $cred = \varepsilon$) and a user pseudonym Nym_U, outputs a credential proof *credproof* for Nym_U. From this the user can obtain a credential *cred* by running Obtain on his secret key sk_U, pseudonym Nym_U and aux_U, the issuer's and delegator's pseudonyms Nym_O and Nym_I, respectively, and *credproof*. Running CredProve on $(pp, Nym_O, cred, sk, Nym, aux, L)$ permits a user to make a credential proof for the pseudonym *Nym* related to *sk* and *aux*. CredVerify verifies a proof

credproof rooted at Nym_O for *Nym*. Note that CredProve outputs a *credproof* for the user that runs it, while Issue outputs a *credproof* for the user one issues or delegates to.

Security. Security is defined by *correctness*, *anonymity* and *unforgeability*. Run honestly, Issue and Obtain must produce credentials on which, for all user pseudonyms, CredProve outputs a proof that is accepted by CredVerify.

Anonymity means that an adversary interacting with honest users cannot distinguish the real game from an ideal game: There are *simulated parameters* which are indistinguishable from the real ones but lead to pseudonyms, credentials and proofs that are *independent* of users' secret keys. Given a trapdoor for these parameters, Issue, Obtain and CredProve can be simulated without the secret inputs *cred*, *sk* and *aux*.

To break unforgeability, an adversary must produce a proof that some *Nym* has a credential although such a credential has never been issued to any pseudonym of the owner of *Nym*. To formalize the notion of "owner", Belenkiy et al. define an algorithm that extracts from a pseudonym a user identity *vk*, which is uniquely defined by the secret key. Moreover, from a credential proof it extracts the identities that represent the underlying delegation chain. We say that a forgery occurs if the adversary produces a credential for authority vk_0 from which are extracted $(vk_1, \ldots, vk_{L-1}, vk_L)$ such that vk_{L-1} is an honest user that never delegated a level-L credential rooted at vk_0 to vk_L.

7.2 Our Instantiation

In the instantiation from [BCC+09] the system parameters are a Groth-Sahai (GS) commitment key and parameters for an authentication scheme. Each user holds a secret key *sk* for the authentication scheme, and a pseudonym is a GS commitment to $f(sk)$ for a one-way function f. To issue and delegate, the issuer and the user run an interactive two-party protocol to compute a proof of knowledge of an *authenticator* on the user's secret key, which is valid under the issuer's secret key. A credential is then a chain of pseudonyms and committed authenticators with GS proofs of validity.

We replace the authenticators (consisting of 11 group elements and verified by 8 pairing-product equations) by automorphic signatures (5 group elements satisfying 3 PPEs). A non-anonymous level-L credential for vk_L rooted at vk_0 is a chain of verification keys and signatures $(\Sigma_1, vk_1, \Sigma_2 \ldots, vk_{L-1}, \Sigma_L)$, where Σ_i is a signature on vk_i under vk_{i-1}. To achieve anonymity, the keys and signatures in the credential are committed to and proofs of validity are added. Using commuting signatures, given a commitment to a key, the issuer can directly make a commitment to a signature on it and a validity proof. This is what enables non-interactive delegation.

However, merely signing user keys does not suffice, as the issuer of a credential might want to add public information to the credential, such as attributes. For delegatable credentials it is also required to include the originator's pseudonym and the delegation level in each certificate to prevent combining different credentials and changing the order within a credential.

In the full version [Fuc10] we therefore give a simple extension of $\mathbf{Sig}$: Scheme $\mathbf{Sig}''$ has message space $\mathbb{Z}_p \times \mathcal{M}$, allowing the signer to specify a public value in addition to M. Its parameters contain one additional group element, but the signatures have the same size as those of $\mathbf{Sig}$. We also define SigCom″, an adaptation of SigCom which

has the public value in $\mathbb{Z}_p$ as additional input, and show that all the other algorithms defined in Definition 1 and instantiated in Section 6 work equally for $\mathbf{Sig}$ and $\mathbf{Sig}''$[7].

Our Scheme. Let $\mathcal{H}\colon \mathcal{C}_{\mathcal{M}} \times \mathbb{N} \to \mathbb{Z}_p$ be a collision-resistant hash function. Then our scheme $\mathbf{Cred}$ can be sketched as follows: $\mathsf{Setup}_{\mathrm{C}}$ generates a key for $\mathbf{Com}$ and parameters for $\mathbf{Sig}''$; $\mathsf{KeyGen}_{\mathrm{C}}$ outputs a signing key for $\mathbf{Sig}''$; and given such a key, NymGen outputs a commitment to the corresponding verification key and the used randomness as auxiliary information. A level-L credential proof from Nym_0 for Nym_L has the form

$$credproof = (\mathbf{c}_1, \pi_1, Nym_1, \mathbf{c}_2, \pi_2, \ldots, Nym_{L-1}, \mathbf{c}_L, \pi_L) \ ,$$

where $\mathbf{c}_i$ is a commitment to a $\mathbf{Sig}''$ signature Σ_i on the public value $\mathcal{H}(Nym_0, i)$ and the key committed in Nym_i, valid under the key committed in Nym_{i-1}; and π_i is a proof of validity of Σ_i. We call *credproof* a credential if it is valid on a "trivial" Nym_L, i.e., when $Nym_L = \mathsf{Com}(ck, vk_L, 0)$.

$\mathsf{CredProve}$ takes a credential and turns it into a credential proof for Nym_L by randomizing all its components, using as randomness for the last component the value *aux* s.t. $Nym_L = \mathsf{Com}(ck, vk_L, aux)$. $\mathsf{CredVerify}$ verifies a *credproof* by checking the proofs contained in it. Given a level-L credential, Issue extends it by one level making a credential proof for the delegatee's pseudonym Nym_{L+1}: In case of a delegation ($L > 0$), it first makes a *credproof* for the issuer's pseudonym Nym_I; otherwise *credproof* $:= \varepsilon$. Running SigCom'' on sk_I, $\mathcal{H}(Nym_0, L+1)$ and Nym_{L+1}, it produces $(\mathbf{c}_{L+1}, \pi'_{L+1})$, a verifiably encrypted signature under vk_I. It then runs $\mathsf{AdC}_{\mathcal{K}}$ on (vk_I, aux_I) to adapt π'_{L+1} to a proof π_{L+1}, which is valid for the committed verification key Nym_I. Finally, Issue outputs $credproof \,\|\, Nym_I \,\|\, (\mathbf{c}_{L+1}, \pi_{L+1})$. Obtain turns a level-L credential proof into a credential by adapting the randomness to make it valid for a trivial Nym_L.

Simulatability. While unforgeability of our scheme is implied by the soundness of $\mathbf{Proof}$ and unforgeability of $\mathbf{Sig}''$, anonymity is shown as follows. We generate the simulated parameters by replacing Setup with $\mathsf{WISetup}$; this makes the commitments perfectly hiding and thus pseudonyms and credential proofs independent of secret keys.

We then have to simulate issuing and $\mathsf{CredProve}$ for pseudonyms of the adversary's choice. This essentially means to simulate credential proofs for given *Nym*'s; thus, simulating commitments to a signature and a proof of validity w.r.t. commitments to a key and to a message which are both given to the simulator.

While Groth and Sahai [GS08] show simulation of *proofs of satisfiability* of equations, where the simulator produces all the commitments, we require a novel type of simulation: given commitments to certain variables of an equation, we have to simulate the remaining commitments and the proof of validity. In the full version [Fuc10] we show that this can be done for a class $\mathcal{E}' \subset \mathcal{E}$ of equations, in which the equations for validity of committed signatures fall. We call a commuting signature scheme *simulatable* if given a commitment to a key and a commitment to a message the simulator can create a verifiably encrypted signature for them.

In [Fuc10] we define a simulatable variant of our scheme from Section 6. We also give a formal description of our credential scheme $\mathbf{Cred}$ and a proof of the following.

[7] Note that replacing SigCom by SigCom'' in the construction of a blind signature at the end of Section 3 yields a *partially blind signature* [AF96].

Theorem 2. *Let* $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}'', \mathbf{Com}_{\mathcal{M}}, \mathbf{Algs})$ *be a system of commuting signatures which is automorphic and simulatable, and let* $\mathcal{H}$ *be a collision-resistant hash function. Then* $\mathbf{Cred}$ *is a secure delegatable anonymous credential scheme.*

8 Conclusion

We introduced and instantiated *commuting signatures and verifiable encryption*. Given an encryption $\mathbf{C}$ of M and a signing key, they let us produce an encryption $\mathbf{c}_\Sigma$ of a signature on M and a proof that $\mathbf{c}_\Sigma$ contains a valid signature on the content of $\mathbf{C}$. We used them to give the first instantiation of delegatable anonymous credentials with non-interactive issuing and delegation and believe that they are a useful tool in the construction of privacy-preserving primitives that will find further applications.

Acknowledgements

The author would like to thank David Pointcheval, Elizabeth Quaglia and Damien Vergnaud for many helpful discussions and the anonymous referees of CCS 2010 and EUROCRYPT for their valuable comments.

References

[AF96] Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)

[AFG+10] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)

[BB04] Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BCC+09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)

[BCKL08] Belenkiy, M., Chase, M., Kohlweiss, M., Lysyanskaya, A.: P-signatures and noninteractive anonymous credentials. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 356–374. Springer, Heidelberg (2008)

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

[BFPV11] Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Gennaro, R. (ed.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (2011)

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

[BHY09] Bellare, M., Hofheinz, D., Yilek, S.: Possibility and impossibility results for encryption and commitment secure under selective opening. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 1–35. Springer, Heidelberg (2009)

[Bra99] Brands, S.: Rethinking public key infrastructure and digital certificates—building privacy. PhD thesis, Eindhoven Inst. of Tech., The Netherlands (1999)

[BW07] Boyen, X., Waters, B.: Full-domain subgroup hiding and constant-size group signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)

[Cha83] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)

[Cha85] Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Commun. ACM 28(10), 1030–1044 (1985)

[CL01] Camenisch, J., 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)

[CL02] Camenisch, J., 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)

[CL04] Camenisch, J., 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)

[CL06] Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Heidelberg (2006)

[Dam90] Damgård, I.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 328–335. Springer, Heidelberg (1990)

[DHLW10] Dodis, Y., Haralambiev, K., López-Alt, A., Wichs, D.: Cryptography against continuous memory attacks. In: FOCS, pp. 511–520. IEEE Computer Society, Los Alamitos (2010)

[Fis06] Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)

[FP08] Fuchsbauer, G., Pointcheval, D.: Anonymous proxy signatures. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 201–217. Springer, Heidelberg (2008)

[FPV09] Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Transferable constant-size fair E-cash. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 226–247. Springer, Heidelberg (2009)

[Fuc09] Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), http://eprint.iacr.org/2009/320, an extended abstract appeared as part of [AFG+10]

[Fuc10] Fuchsbauer, G.: Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. Cryptology ePrint Archive, Report 2010/233 (2010), http://eprint.iacr.org/2010/233

[GOS06] 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)

[GPS08] Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

[GS08] 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)

[LRSW00] Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym systems. In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)

[PS96] Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (1996)

[RS09] Rückert, M., Schröder, D.: Security of verifiably encrypted signatures and a construction without random oracles. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 17–34. Springer, Heidelberg (2009)

Secure Authentication from a Weak Key, without Leaking Information

Niek J. Bouman and Serge Fehr

Centrum Wiskunde & Informatica (CWI), Amsterdam, The Netherlands
{n.j.bouman,serge.fehr}@cwi.nl

Abstract. We study the problem of authentication based on a weak key in the information-theoretic setting. A key is weak if its min-entropy is an arbitrary small fraction of its bit length. This problem has recently received considerable attention, with different solutions optimizing different parameters. We study the problem in an extended setting, where the weak key is a one-time *session key* that is derived from a public source of randomness with the help of a (potentially also weak) *long-term* key. Our goal now is to authenticate a message by means of the weak session key in such a way that (nearly) no information on the long-term key is leaked. Ensuring privacy of the long-term key is vital for the long-term key to be re-usable. Previous work has not considered such a privacy issue, and previous solutions do not seem to satisfy this requirement.

We show the existence of a practical four-round protocol that provides message authentication from a weak session key and that avoids non-negligible leakage on the long-term key. The security of our scheme also holds in the quantum setting where the adversary may have limited quantum side information on the weak session key. As an application of our scheme, we show the existence of an identification scheme in the bounded quantum storage model that is secure against a man-in-the-middle attack and that is truly password-based: it does not need any high entropy key, in contrast to the scheme proposed by Damgård *et al.*

1 Introduction

1.1 The Problem

We consider the problem of achieving authentic communication over a public channel that might be under the control of an active adversary. We study this problem in the information-theoretic setting, i.e. we assume the adversary to be computationally unbounded.

Specifically, we consider the following scenario. Alice and Bob share a *long-term* key W. When needed, Alice and Bob can extract a weak *session key* X_W from an auxiliary source of randomness with the help of W. It should be guaranteed by the property of the auxiliary source that a potential adversary Eve who does not know W has limited information on the weak session key X_W. This is formalized by requiring that $H_{\min}(X_W|WE) \geq k$ for some parameter k, where E denotes Eve's side information. Examples of where this scenario occurs

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 246–265, 2011.

naturally are the bounded storage model, where W determines which part of the huge string to read, or the quantum setting, where W determines in which basis to measure some quantum state.

The goal now is to authenticate a message μ from Alice to Bob with the help of the weak session key X_W, in such a way that (1) Eve cannot tamper with μ without being detected, and (2) Eve learns (nearly) no information on the long-term key W. We stress that property (2) is vital for Alice and Bob to be able to re-use W. Note that once Alice and Bob can do message authentication with a weak key, then they can also do key agreement, simply by doing standard randomness extraction where the seed for the extractor is communicated in an authentic way.

We want to emphasize that, by assumption, every new session key X_W for the same long-term key W contains fresh randomness, provided by the auxiliary source. Therefore, the goal above does not contradict the well-known impossibility result of re-using an authentication key without refreshing. Also note that we do not specify how exactly the auxiliary source of randomness produces X_W from W; on the contrary, we want security no matter how X_W is obtained, as long as X_W contains enough min-entropy (given the adversary's information and W).

1.2 Related Work

Let n be the bitsize of the key (in our case, the session key) and k its min-entropy (in bits). It was proved by Dodis and Wichs [9] that non-interactive authentication is impossible when $k \leq n/2$, even when the parties have access to local non-shared randomness, which we will assume. For a good overview of earlier work on the case where $k > n/2$, we refer to [9].

The first protocol for interactive authentication from arbitrarily weak keys is due to Renner and Wolf [15]. It requires $\Theta(\ell)$ rounds of interaction to authenticate an ℓ-bit message. In [9], an authentication protocol from arbitrarily weak keys is described that only needs two rounds of interaction, which is optimal (in terms of the number of rounds). Chandran *et al.* [2] focus on minimizing entropy loss and describe a privacy amplification protocol that is optimal with respect to entropy loss (up to constant factors). Their construction needs a linear number of rounds (linear in the security parameter).

The case where Alice and Bob share highly-correlated, but possibly unequal keys – the "fuzzy" case – is addressed in [16] and improved upon by Kanukurthi and Reyzin [11], but also covered by [9] and [2].

We stress that none of these works address the case where the weak key is obtained from a long-term key and where security of the long-term key needs to be guaranteed.

1.3 Our Contributions

We propose a new four-round protocol for message authentication with a weak session key X_W. We prove that our protocol satisfies *security* and *long-term key*

privacy, meaning that the adversary Eve cannot tamper with the authenticated message without being detected, nor does she learn any (non-negligible amount of) information on the long-term key W. Our proofs also apply in the quantum setting, where Eve's bounded knowledge on X_W may be in the form of a quantum state.

We also discuss how our techniques can be applied in the fuzzy case, where there are some errors between Alice and Bob's weak session keys. Finally, we outline how our scheme can be used to improve an existing password-based identification scheme in the bounded-quantum-storage model (more details on this application are given below).

1.4 Application

Our main application is to password-based identification in the bounded quantum storage model, as proposed by Damgård *et al.* [4]. Two identification schemes were proposed in [4], *Q-ID*, which is only secure against dishonest Alice or Bob, and *Q-ID*$^+$, which is also secure against against a man-in-the-middle (MITM) attack. However, only *Q-ID* is truly password-based; in *Q-ID*$^+$, Alice and Bob, in addition to the password, also need to share a high-entropy key. By incorporating our new techniques into *Q-ID*$^+$, we show the existence of a truly password-based identification scheme in the bounded-quantum-storage model with security against MITM attacks.

Based on *Q-ID*$^+$, Damgård *et al.* also propose an *authenticated* quantum key distribution scheme in the bounded quantum storage model, which, in contrast to standard quantum key distribution schemes, does not require authenticated communication but has the authentication "built in"[1]. Our relaxation on the required key material in *Q-ID*$^+$ also affects their authenticated quantum key distribution scheme and circumvents the need for a high entropy key. As a result, we obtain a truly password-based authenticated quantum key distribution scheme in the bounded-quantum-storage model.

1.5 Organization of the Paper

The paper is structured as follows. In Section 2 we introduce notation, give some standard definitions and introduce the security definition that our authentication protocol should fulfill. Then, in Section 3, we describe an existing authentication protocol that we use as a basis for our protocol. We also explain there why this existing protocol does not fulfill our security definition, and we discuss some steps how we extend that protocol. This ultimately leads to our own protocol `AUTH`, which is introduced in Section 4. In the same section, we present an important lemma that is used in the security proof to deal with a certain circularity issue. Section 5 consists of the proofs for security and privacy. In Section 6 we argue that our authentication protocol can also be used in the fuzzy case and finally

[1] Furthermore, in contrast to using standard quantum key distribution in combination with standard authentication, in the authenticated quantum key distribution scheme the authentication keys can be re-used.

Section 7 discusses our application. Most results related to instantiating our protocol can be found in the full version of this paper [1].

2 Notation and Preliminaries

We prove security of our scheme in the presence of a *quantum* adversary with *quantum* side information, and below we introduce some suitable notations. However, we stress that most of the notation and the proofs can also be understood from a purely classical information-theoretical point of view.

The *state* of a quantum system X is given by a *density matrix* ρ_X, i.e., a positive-semidefinite trace-1 matrix acting on some Hilbert space $\mathcal{H}_X$. We denote the set of all such matrices, acting on $\mathcal{H}_X$, by $\mathcal{P}(\mathcal{H}_X)$. In the special case where ρ_X is diagonal, X is called *classical*, and in this case we can understand X as a random variable, where its distribution P_X is given by the diagonal entries of ρ_X. In this case, we tend to slightly abuse notation and write $X \in \mathcal{X}$ to indicate that the range of the random variable X is $\mathcal{X}$.

If X is part of a bi-partite system XE, then X is called classical if the density matrix ρ_{XE} of XE is of the form $\rho_{XE} = \sum_x P_X(x)|x\rangle\langle x| \otimes \rho_{E|X=x}$, where P_X is a probability distribution, $\{|x\rangle\}_x$ forms an orthonormal basis of $\mathcal{H}_X$, and $\rho_{E|X=x} \in \mathcal{P}(\mathcal{H}_E)$. In this case, X can be understood as random variable, and system E is in state $\rho_{E|X=x}$ exactly if X takes on the value x. We therefore sometimes also speak of a random variable X and a quantum system E. To simplify notation, we often write ρ_E^x instead of $\rho_{E|X=x}$. Readers that are unfamiliar with quantum information can safely think of E as being classical as well, in which case the $\rho_{E|X=x}$'s are all diagonal, with the probabilities of the conditional distributions $P_{E|X}(\cdot|x)$ as diagonal entries.

The distance between two states $\rho_X, \sigma_X \in \mathcal{P}(\mathcal{H}_X)$ is measured by their *trace distance* $\frac{1}{2}\|\rho_X - \sigma_X\|_1$, where $\|\cdot\|_1$ is the L_1 norm[2]. In case of classical states, i.e., ρ_X and σ_X correspond to distributions P_X and Q_X, the trace distance coincides with the statistical distance $\frac{1}{2}\sum_x |P_X(x) - Q_X(x)|$.

In the following definitions, we consider a bi-partite system XE with classical X. X is said to be *random and independent* of E if $\rho_{XE} = \rho_U \otimes \rho_E$, where ρ_U is the fully mixed state on $\mathcal{H}_X$ (i.e., U is classical and - as random variable - uniformly distributed). In case of classical E, this is equivalent to $P_{XE} = P_U \cdot P_E$ (in the sense that $P_{XE}(x,e) = P_U(x) \cdot P_E(e)\ \forall x, e$). The following definition measures how far away XE is from such an ideal situation.

Definition 1 (Distance to Uniform). *The* distance to uniform *of X given E is defined as*

$$d(X|E) := \tfrac{1}{2}\|\rho_{XE} - \rho_U \otimes \rho_E\|_1.$$

If also E is classical, then $d(X|E)$ simplifies to

$$d(X|E) = \tfrac{1}{2}\sum_{x,e} |P_{XE}(x,e) - P_U(x)P_E(e)| = \sum_e P_E(e)\ \tfrac{1}{2}\sum_x \left|P_{X|E}(x|e) - P_U(x)\right|.$$

[2] Defined by $\|A\|_1 := \mathrm{trace}(\sqrt{A^\dagger A})$, where $A^\dagger$ denotes the Hermitian transpose.

It is not too hard to show that for a tri-partite system XYE with classical X and Y

$$d(X|YE) = \sum_{y \in \mathcal{Y}} P_Y(y)\, d(X|E, Y=y).$$

From this, the following lemma follows immediately.

Lemma 1. *For any y: $d(X|E, Y=y) \leq d(X|YE)/P_Y(y)$.*

Definition 2 (Guessing Probability). *The* guessing probability *of X given E is defined as*

$$\mathsf{Guess}(X|E) := \sup_{\{M_x\}_x} \sum_x P_X(x) \mathrm{tr}(M_x\, \rho_E^x),$$

where the supremum is over all POVMs $\{M_x\}_x$ on $\mathcal{H}_E$.

In case also E is classical, $\mathsf{Guess}(X|E)$ simplifies to the standard average guessing probability

$$\mathsf{Guess}(X|E) = \sum_e P_E(e) \max_x P_{X|E}(x|e).$$

Definition 3 (Min-Entropy). *The min-entropy of X given E is defined as*

$$H_{\min}(X|E) := -\log \mathsf{Guess}(X|E).$$

This definition coincides with the definition introduced by Renner [13], as shown by [12]; in case of a classical E, it coincides with the classical definition of conditional min-entropy (see e.g. [7]).

Definition 4. *A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ε)-strong extractor, if for any bipartite quantum system XE with classical X and with $H_{\min}(X|E) \geq k$, and for a uniform and independent seed Y, we have*

$$d\big(\mathsf{Ext}(X,Y)\big|YE\big) \leq \varepsilon\,.$$

Note that we find "extractor against quantum adversaries" a too cumbersome terminology; thus we just call Ext a (strong) extractor, even though it is a stronger notion than the standard notion of a (strong) extractor. When necessary, we distinguish between the two notions by saying that an extractor is or is not *secure against quantum side information*.

A well-known example of a strong extractor (that is secure against quantum side information) is a two-universal hash function $h : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^q$. Indeed, for any XE with classical X, and for Y an independent seed, uniformly distributed on $\{0,1\}^d$ privacy amplification [14] guarantees that

$$d(h(X,Y)|YE) \leq \frac{1}{2}\sqrt{2^{q - H_{\min}(X|YE)}} = \frac{1}{2}\sqrt{2^q\, \mathsf{Guess}(X|YE)}.$$

2.1 Security Definition

In the scope of this paper, an authentication protocol is understood as a classical protocol between two parties Alice and Bob. Alice inputs a message μ and a weak session key X_W, and Bob inputs a message μ' and the same session key X_W. At the end of the protocol, Bob announces a Boolean decision whether to "accept" or "reject". The weak session key X_W may depend arbitrarily on a long-term key W. During the execution of the protocol, an adversary Eve has full control over the communication between Alice and Bob.

We require the protocol to fulfill the following formal definition.

Definition 5. *Let $E_\circ, E$ denote Eve's respective a priori and a posteriori quantum systems, where the latter includes Bob's decision on whether to accept or reject. A $(n, k, m, \delta, \varepsilon)$ message authentication protocol with long-term-key privacy is defined to satisfy the following properties:*

Correctness: *If there is no adversary Eve present, then for any message $\mu \in \{0,1\}^m$ and $\mu' = \mu$, and for any (distribution of the) key $X_W \in \{0,1\}^n$, Bob accepts with certainty.*

Security: *If $H_{\min}(X_W|WE_\circ) > k$, then for any $\mu, \mu' \in \{0,1\}^m$ with $\mu \neq \mu'$, the probability that Bob accepts is at most δ.*

Long-Term-Key Privacy: *If $\rho_{WE_\circ} = \rho_W \otimes \rho_{E_\circ}$ and $H_{\min}(X_W|WE_\circ) > k$, then*

$$\frac{1}{2}\|\rho_{WE} - \rho_W \otimes \rho_E\|_1 \leq \varepsilon\,.$$

3 The Dodis-Wichs Authentication Scheme

Here, we describe a slightly modified version of the two-round message authentication protocol due to Dodis and Wichs [9]. Our construction will be based on this protocol. We start by giving a few definitions that are crucial for the understanding of the protocol by Dodis and Wichs.

Definition 6 (Epsilon Look-Aheadness). *Let t, ℓ be positive integers. Let $A := (A_1, \ldots, A_t)$ and $B := (B_1, \ldots, B_t)$ be random variables over $(\{0,1\}^\ell)^t$, and let E be a quantum system. For all $i \in \{0, \ldots, t-1\}$ let ε_i be defined as*

$$\varepsilon_i := d\big(A_{i+1} \ldots A_t \big| B_1 \ldots B_i E\big)\,.$$

*The ordered pair (A, B) is ε-*look-ahead conditioned on *E if $\varepsilon \geq \max_i \varepsilon_i$.*

Definition 7 (Look-Ahead Extractor). $\mathsf{laExt} : \{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^\ell)^t$ *is called a (k, ε)-*look-ahead extractor *if for any random variable $X \in \{0,1\}^n$ and quantum system E with $H_{\min}(X|E) \geq k$ the following holds. Let $S \in \{0,1\}^d$ be a independent and uniformly distributed seed, and let $\tilde{S} \in \{0,1\}^d$ be adversarially chosen given S and E; this may involve a (partial) measurement of E, resulting in the new state E'. Then, the ordered pair $(R, \tilde{R})$ where $R = (R_1, \ldots, R_t) := \mathsf{laExt}(X; S)$ and $\tilde{R} = (\tilde{R}_1, \ldots, \tilde{R}_t) := \mathsf{laExt}(X; \tilde{S})$ is ε-look-ahead conditioned on S, $\tilde{S}$ and E'.*

Informally, a look-ahead extractor has the property that even if the adversary is allowed to modify the seed, when given the first i blocks of the key that is extracted using the modified seed, the remaining blocks of the key that is extracted using the correct seed still look random.

Definition 8 (Look-ahead security). *A family of functions*

$$\{\mathsf{MAC}_k : \{0,1\}^m \to \{0,1\}^s\}$$

indexed by keys $k \in (\{0,1\}^\ell)^t$ *is an* (ε, δ) look-ahead secure *MAC if for any pair of fixed and distinct messages* $\mu_\mathrm{A}, \mu_\mathrm{B} \in \{0,1\}^m, \mu_\mathrm{A} \neq \mu_\mathrm{B}$*, and any ordered pair of random variables* $(K, K') \in (\{0,1\}^\ell)^{2t}$ *satisfying the look-ahead property with parameter* ε *conditioned on quantum system* E*,*

$$\mathsf{Guess}\big(\mathsf{MAC}_K(\mu_\mathrm{B}) \,\big|\, \mathsf{MAC}_{K'}(\mu_\mathrm{A})E\big) < \delta\,.$$

We are now ready to present the Dodis and Wichs message authentication protocol DW-MAC. The protocol we present here is slightly modified in that we assume that Alice has already sent her message μ_A to Bob, who has received it as μ_B (possibly $\neq \mu_\mathrm{A}$). This modification is for simplicity, and because we do not aim at minimizing the number of rounds. X_W is the weak key, known to both Alice and Bob. The function $\mathsf{laExt}\colon \{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^\ell)^t$ is a (k, ε)-look-ahead extractor and $\mathsf{MAC}_k : \{0,1\}^m \to \{0,1\}^s$ is a (ε, δ) look-ahead secure MAC.

Protocol DW-MAC

$Alice(X_W, \mu_\mathrm{A})$		$Bob(X_W, \mu_\mathrm{B})$
		$R \in_\mathrm{R} \{0,1\}^d$
	$\xleftarrow{R}$	
$K := \mathsf{laExt}(X_W; R)$		$K := \mathsf{laExt}(X_W; R)$
$T_\mathrm{A} := \mathsf{MAC}_K(\mu_\mathrm{A})$		$T_\mathrm{B} := \mathsf{MAC}_K(\mu_\mathrm{B})$
	$\xrightarrow{T_\mathrm{A}}$	
		accept if: $T_\mathrm{A} = T_\mathrm{B}$
		else: abort

Security of DW-MAC follows immediately from the definitions of the underlying building blocks: laExt ensures that Alice and Bob's versions of the key K satisfy the look-ahead property, and in this case it is guaranteed that MAC acts as a secure MAC, even when Alice's key was modified.

However, in our setting where we additionally want to maintain privacy of the long-term key W, which may arbitrarily depend on X_W, DW-MAC does not seem to be good enough — unless Eve remains passive. Indeed, if Eve does not manipulate the communicated seed R, then by the assumed lower bound on $H_\mathrm{min}(X_W|WE)$, it follows that the extracted K on Bob's side is close to random and independent of W (and E), and thus T leaks no information on W.

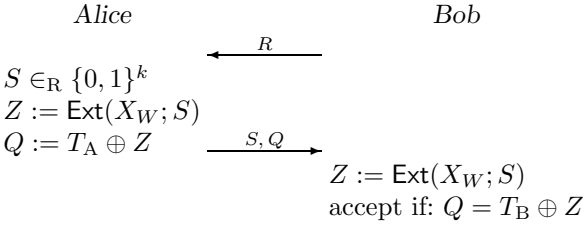
However, if Eve manipulates the seed R (for instance replaces it by a value of her choice), then there is no guarantee anymore that K, and thus T, does no leak information on W.

Another and more subtle way for Eve to (potentially) learn information on W is by not manipulating the message, i.e., have $\mu_A = \mu_B$, but manipulate the seed R and try to obtain information on W by observing if Bob accepts or not.

3.1 Towards Achieving Key-Privacy

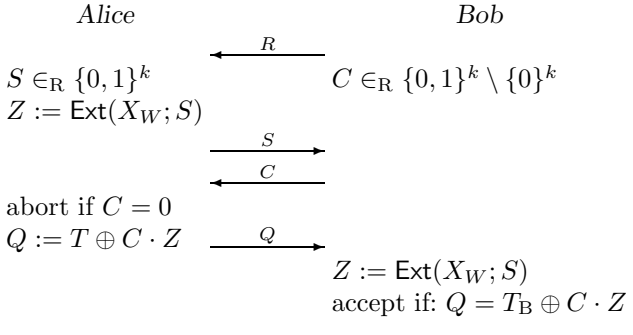
We give here some intuition on how we overcome the above privacy issues of DW-MAC with respect to the long-term key W. Similarly to our notation T_A and T_B to distinguish between the tag computed by Alice and by Bob, respectively, we write R_A and R_B etc. to distinguish between Alice and Bob's values of R etc., which may be different if Eve actively manipulates communicated messages.

A first approach to prevent leakage through T_A is to one-time-pad encrypt T_A. The key for the one-time-pad is extracted by means of a strong extractor Ext from X_W, where Alice chooses the seed:



In the above protocol (and also below), we understand T_A and T_B to be computed as in DW-MAC. Note that since it is Alice who chooses the seed S and since $H_{\min}(X_W|WE)$ is lower bounded, Z_A is guaranteed to be (close to) random and independent of W (and E), and thus hides all information that T_A might leak on W. However, this modification renders the *security* of the scheme invalid. For instance, we cannot exclude that by modifying the seed S appropriately, Eve can enforce $Z_B = T_B$, so that she only needs to send $Q = 0$ to have Bob convinced.

In order to re-gain security while still preventing information to leak through T_A, we let Bob choose a random non-zero "multiplier" for the one-time pad key Z:



The multiplication $C \cdot Z$ is to be understood in the corresponding binary field. Leakage through T_{A} is still prevented since a non-zero multiple of a good one-time-pad key is still a good one-time-pad key. Furthermore, for security, we can intuitively argue as follows. Consider a snapshot of an execution of the protocol after S has been communicated. We now give Eve the value T_{A} for free; this only makes her stronger. By the security of the underlying DW-MAC scheme, we know that it is hard for Eve to guess T_{B}. Now, assuming that there exist two distinct values for C for which Eve can predict the corresponding value $Q_{\mathrm{B}} = T_{\mathrm{B}} \oplus C \cdot Z_{\mathrm{B}}$, it follows immediately that Eve can actually predict T_{B}; a contradiction. Hence, there can be at most one value for Bob's choice of C for which Eve can guess Q_{B} reasonably well.

We point out that the above intuitive reasoning involves *rewinding*; this is fine in the classical but fails in the quantum setting (see e.g. [17]). Thus, in our formal security proof where we allow Eve to maintain a quantum state, we have to reason in a different way. As a consequence, in the actual protocol, Q is computed in a slightly different way.

One issue that is still unsolved is that Bob's decision to accept or reject may also leak information on W when $\mu_{\mathrm{A}} = \mu_{\mathrm{B}}$ and Eve modifies one (or both) of the seeds R and S. Note that this is not an issue if $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$ because then, by the security, Bob rejects with (near) certainty. For instance it might be that changing the first bit of S changes Z or not, depending on what the first bit of X_W is. Thus, by changing the first bit of S and observing Bob's decision, Eve can learn the first bit of X_W, which may give one bit of information on W. The solution to overcome this problem is intuitively very simple: we use MAC not only to authenticate the actual message, but also to authenticate the two seeds R and S. Then, like in the case $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$, if Eve changes one of the seeds then Bob's decision is determined to be reject. Note that this modification introduces a circularity: the key K, which is used to authenticate the seed R (amongst the message and S) is extracted from X_W by means of the seed R. However, it turns out that we can deal with this.

4 Main Construction

We now turn to our construction for the message authentication protocol with long-term-key privacy (Definition 5). In the construction, we will use DW-MAC as a building block. Informally speaking, the basic idea is to encrypt the authentication tag from DW-MAC using a one-time pad, which prevents key leakage. The key for this one-time pad is established in a challenge-response sequence, from a mix of local and shared randomness. Additionally, we use the DW-MAC protocol to authenticate some of the extractor seeds that appear in the construction, to prevent key-leakage from Bob's accept/reject decision.

Let $\mathsf{laExt} : \{0,1\}^n \times \{0,1\}^d \to (\{0,1\}^\ell)^t$ be a (k_K, ε_K) look-ahead extractor. Let $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^v \to \{0,1\}^q$ be a (k_Z, ε_Z)-strong extractor. Let $\mathsf{MAC} : (\{0,1\}^\ell)^t \times (\{0,1\}^m \times \{0,1\}^d \times \{0,1\}^v) \to \{0,1\}^s$ be a $(\epsilon, \lambda + \epsilon)$ look-ahead secure MAC, for any $\epsilon > 0$. Let X_W be the session key, shared among Alice and Bob,

and satisfy $H_{\min}(X_W|WE_\circ) > \max(k_K + q, k_Z)$. The "$\oplus$" symbol represents bit-wise addition modulo 2. Multiplication, denoted by "$\cdot$", should be understood as multiplication in the corresponding finite field: $\mathrm{GF}(2^s)$ or $\mathrm{GF}(2^q)$. We write $[b]_q$ for the q most significant bits of the bit-string b. Protocol AUTH is shown below.

Protocol AUTH

$Alice(X_W, \mu_\mathrm{A})$		$Bob(X_W, \mu_\mathrm{B})$
		$R \in_\mathrm{R} \{0,1\}^d$
	$\xleftarrow{R}$	
$K := \mathsf{laExt}(X_W; R)$		$K := \mathsf{laExt}(X_W; R)$
$S \in_\mathrm{R} \{0,1\}^v$		
	$\xrightarrow{S}$	
$Z := \mathsf{Ext}(X_W; S)$		$Z := \mathsf{Ext}(X_W; S)$
$T_\mathrm{A} := \mathsf{MAC}_K((\mu_\mathrm{A}, R, S))$		$T_\mathrm{B} := \mathsf{MAC}_K((\mu_\mathrm{B}, R, S))$
		$U \in_\mathrm{R} \{0,1\}^s, V \in_\mathrm{R} \{0,1\}^q \setminus \{0\}^q$
	$\xleftarrow{U,V}$	
if $U, V = 0$: abort		
$Q := [U \cdot T_\mathrm{A}]_q \oplus V \cdot Z$		
	$\xrightarrow{Q}$	
		accept if: $Q = [U \cdot T_\mathrm{B}]_q \oplus V \cdot Z$
		else: abort

In the full version of this paper [1], we show how to instantiate the building blocks (due to space restrictions we have only included the part about instantiating the MAC in the present version, in Appendix A) to obtain a scheme with reasonable parameters. In doing so, we use similar techniques as [9], except that we replace the strong extractors that are part of the look-ahead extractor construction by extractors that are proven secure against quantum side information (by [6]).

Depending on the parameters of an instantiation of AUTH and on the bitsize of μ_A, it might be beneficial, or could even be necessary, to authenticate a hash of the tuple (μ_A, R, S), instead of authenticating the tuple itself. In this case, we let Alice choose a small seed for an almost universal hash function and apply MAC_K to this seed and the hash of the the tuple (μ_A, R, S) (with respect to this seed). We will actually make use of this suggested modification when instantiating AUTH.

Before going into the security proof for protocol AUTH, we resolve here the circularity issue obtained by authenticating the seed R that was used to extract the authentication key K.

Lemma 2. *Consider a MAC that is $(\epsilon, \lambda + \epsilon)$-look-ahead-secure for any ϵ. Let $K, K', M_\mathrm{A}, M_\mathrm{B}$ be arbitrary random variables and E a quantum state, and let the ordered pair $(K, K') \in (\{0,1\}^\ell)^{2t}$ satisfy the look-ahead property with parameter ε conditioned on $M_\mathrm{A}, M_\mathrm{B}, E$ and the event $M_\mathrm{A} \neq M_\mathrm{B}$. Then,*

$$\mathsf{Guess}\big(\mathsf{MAC}_K(M_\mathrm{B}) \,\big|\, \mathsf{MAC}_{K'}(M_\mathrm{A}) M_\mathrm{A} M_\mathrm{B} E, M_\mathrm{A} \neq M_\mathrm{B}\big) < \lambda + t\varepsilon.$$

Proof. We condition on $M_\mathrm{A} = m_\mathrm{A}$ and $M_\mathrm{B} = m_\mathrm{B}$ and assume throughout the proof that $m_\mathrm{A} \neq m_\mathrm{B}$. Because (K, K') may depend on $(M_\mathrm{A}, M_\mathrm{B})$, conditioning on fixed values for the latter implies that (K, K') is not necessarily ε-look-ahead anymore. Let $\varepsilon_{m_\mathrm{A},m_\mathrm{B}}$ be the maximum over $i \in [t]$ of the following expression,

$$\varepsilon_{m_\mathrm{A},m_\mathrm{B},i} := d(K_{i+1} \dots K_t \big| K'_1 \dots K'_i E, M_\mathrm{A} = m_\mathrm{A}, M_\mathrm{B} = m_\mathrm{B}).$$

Hence, by Definition 6, (K, K') is $\varepsilon_{m_\mathrm{A},m_\mathrm{B}}$-look-ahead conditioned on E and the events $M_\mathrm{A} = m_\mathrm{A}$ and $M_\mathrm{B} = m_\mathrm{B}$. Note that averaging $\varepsilon_{m_\mathrm{A},m_\mathrm{B},i}$ over m_A and m_B (conditioned on them being distinct) results in

$$\varepsilon_i = d(K_{i+1} \dots K_t | K'_1 \dots K'_i M_\mathrm{A} M_\mathrm{B} E, M_\mathrm{A} \neq M_\mathrm{B}) \leq \varepsilon\,.$$

Furthermore, note that by conditioning on fixed and distinct values for M_A and M_B, we fulfill the requirements for MAC look-ahead security from Definition 8. I.e. we can conclude that

$$\mathsf{Guess}\big(\mathsf{MAC}_K(M_\mathrm{B}) \,\big|\, \mathsf{MAC}_{K'}(M_\mathrm{A}) E, M_\mathrm{A} = m_\mathrm{A}, M_\mathrm{B} = m_\mathrm{B}\big) < \lambda + \varepsilon_{m_\mathrm{A},m_\mathrm{B}}.$$

It now follows that

$$\begin{aligned}
&\mathsf{Guess}\big(\mathsf{MAC}_K(M_\mathrm{B}) \,\big|\, \mathsf{MAC}_{K'}(M_\mathrm{A}) M_\mathrm{A} M_\mathrm{B} E, M_\mathrm{A} \neq M_\mathrm{B}\big) \\
&= \sum_{m_\mathrm{A},m_\mathrm{B}} P_{M_\mathrm{A} M_\mathrm{B} | M_\mathrm{A} \neq M_\mathrm{B}}(m_\mathrm{A}, m_\mathrm{B}) \\
&\qquad \cdot \mathsf{Guess}\big(\mathsf{MAC}_K(M_\mathrm{B}) \,\big|\, \mathsf{MAC}_{K'}(M_\mathrm{A}) E, M_\mathrm{A} = m_\mathrm{A}, M_\mathrm{B} = m_\mathrm{B}\big) \\
&< \sum_{m_\mathrm{A},m_\mathrm{B}} P_{M_\mathrm{A} M_\mathrm{B} | M_\mathrm{A} \neq M_\mathrm{B}}(m_\mathrm{A}, m_\mathrm{B}) \, (\lambda + \max_{i \in [t]} \varepsilon_{m_\mathrm{A},m_\mathrm{B},i}) \\
&\leq \lambda + \sum_{m_\mathrm{A},m_\mathrm{B}} P_{M_\mathrm{A} M_\mathrm{B} | M_\mathrm{A} \neq M_\mathrm{B}}(m_\mathrm{A}, m_\mathrm{B}) \sum_{i \in [t]} \varepsilon_{m_\mathrm{A},m_\mathrm{B},i} \\
&= \lambda + \sum_{i \in [t]} \sum_{m_\mathrm{A},m_\mathrm{B}} P_{M_\mathrm{A} M_\mathrm{B} | M_\mathrm{A} \neq M_\mathrm{B}}(m_\mathrm{A}, m_\mathrm{B}) \, \varepsilon_{m_\mathrm{A},m_\mathrm{B},i} \\
&= \lambda + \sum_{i \in [t]} \varepsilon_i \leq \lambda + \sum_{i \in [t]} \varepsilon = \lambda + t\varepsilon.
\end{aligned}$$

This concludes the proof.

5 Proofs of Security and Privacy

In this section we show that protocol `AUTH` fulfills the properties listed in Definition 5. First of all, note that it is easy to see from the protocol description that the correctness property is satisfied, we do not elaborate further on this here.

Throughout the proofs, let $E_\circ$ be Eve's quantum side information before executing `AUTH`. E_i, where $i \in \{1, \dots, 4\}$, represents Eve's (quantum) side information after the ith round of communication, and hence includes the communicated

random variables up to this ith round. E represents Eve's side information after executing AUTH, including Bob's decision to accept or reject (E_4 does not include this decision). Furthermore, like in Section 3.1, we write R_{A} and R_{B} etc. for Alice and Bob's respective values for R etc.

Theorem 1 (Security). *Assuming that* $H_{\min}(X_W|WE_\circ) > k_K + q$*, Protocol* AUTH *fulfills the security property defined in Definition 5 with*

$$\delta \leq 3 \cdot 2^{-q} + \frac{1}{2}\sqrt{2^q(\lambda + t\,\varepsilon_K)}.$$

In fact, we will prove a slightly stronger statement than the security statement, which will be of use also in the proof of the key privacy statement. Let $M_{\mathrm{A}} := (\mu_{\mathrm{A}}, R_{\mathrm{A}}, S_{\mathrm{A}})$ and $M_{\mathrm{B}} := (\mu_{\mathrm{B}}, R_{\mathrm{B}}, S_{\mathrm{B}})$. We will prove that in protocol AUTH, if $H_{\min}(X_W|WE_\circ) > k_K + q$, and conditioned on the event $M_{\mathrm{A}} \neq M_{\mathrm{B}}$, Bob rejects except with probability

$$\delta' \leq 3 \cdot 2^{-q} + \frac{1}{2}\sqrt{2^q(\lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}])}.$$

Note that this expression reduces to the simpler expression of Theorem 1 when proving security, because in that case $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$ (by Definition 5) which implies that $\Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}] = 1$.

Proof. Consider the phase in protocol AUTH after the second round of communication. Assume that Z_{A} and T_{A} are given to the adversary (this will only make her stronger). Let $K_{\mathrm{A}} := \mathsf{laExt}(X_W; R_{\mathrm{A}})$ and $K_{\mathrm{B}} := \mathsf{laExt}(X_W; R_{\mathrm{B}})$.

From the chain rule, and by subsequently using that R_{B} and S_{A} are sampled independently, it follows that

$$H_{\min}(X_W|Z_{\mathrm{A}}WE_2) \geq H_{\min}(X_W|WE_2) - q \geq H_{\min}(X_W|WE_\circ) - q.$$

By assumption on the parameters, i.e. $H_{\min}(X_W|WE_\circ) > k_K + q$, it follows that $(K_{\mathrm{B}}, K_{\mathrm{A}})$ is ε_K-look-ahead conditioned on Z_{A}, W and E_2. In order to apply Lemma 2, we additionally condition on the event $M_{\mathrm{A}} \neq M_{\mathrm{B}}$. By Lemma 1, it is guaranteed that ε_K grows at most by a factor $1/\Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}]$ as a result of this conditioning. We now apply Lemma 2 and conclude that

$$\mathsf{Guess}(T_{\mathrm{B}}|T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}}) \leq \lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}].$$

The next step is to view $Q_{\mathrm{B}} := [U_{\mathrm{B}} \cdot T_{\mathrm{B}}]_q \oplus V_{\mathrm{B}} \cdot Z_{\mathrm{B}}$ as the output of a strong extractor, with seed $(U_{\mathrm{B}}, V_{\mathrm{B}})$. Indeed, it is straightforward to verify that $h : \{0,1\}^s \times \{0,1\}^q \times \{0,1\}^s \times \{0,1\}^q \to \{0,1\}^q$, which maps (t, z, u, v) to $[u \cdot t]_q \oplus v \cdot z$, is a universal hash function (with random seed (u, v)). Thus, we can apply privacy amplification. One subtlety is that in protocol AUTH, V_{B} is random in $\{0,1\}^q \setminus \{0\}^q$, rather than in $\{0,1\}^q$. However, this affects the overall state by at most an additive term 2^{-q}, and thus, by triangle inequality, the distance-to-uniform by at most $2 \cdot 2^{-q}$:

$$\begin{aligned} &d(Q_{\mathrm{B}}|U_{\mathrm{B}}V_{\mathrm{B}}T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}}) \\ &\quad\leq \tfrac{1}{2}\sqrt{2^q \mathsf{Guess}(T_{\mathrm{B}}Z_{\mathrm{B}}|T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}})} + 2 \cdot 2^{-q} \\ &\quad\leq \tfrac{1}{2}\sqrt{2^q \mathsf{Guess}(T_{\mathrm{B}}|T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}})} + 2 \cdot 2^{-q} \\ &\quad\leq \tfrac{1}{2}\sqrt{2^q(\lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}])} + 2 \cdot 2^{-q}. \end{aligned}$$

Finally, we have that

$$\begin{aligned} \delta' &= \mathsf{Guess}\big(Q_{\mathrm{B}}\big|Q_{\mathrm{A}}WE_3, M_{\mathrm{A}} \neq M_{\mathrm{B}}\big) \\ &\leq \mathsf{Guess}\big(Q_{\mathrm{B}}\big|U_{\mathrm{B}}V_{\mathrm{B}}T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}}\big) \\ &\leq 2^{-q} + d(Q_{\mathrm{B}}|U_{\mathrm{B}}V_{\mathrm{B}}T_{\mathrm{A}}Z_{\mathrm{A}}WE_2, M_{\mathrm{A}} \neq M_{\mathrm{B}}) \\ &\leq 3 \cdot 2^{-q} + \tfrac{1}{2}\sqrt{2^q(\lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}])}. \end{aligned}$$

Theorem 2 (Long-Term-Key Privacy). *Assuming that* $H_{\min}(X_W|WE_\circ) > \max(q+k_K, k_Z)$*, Protocol* AUTH *fulfills the long-term-key privacy property defined in Definition 5 with*

$$\varepsilon \leq 6 \cdot 2^{-q} + \sqrt{2^q\big(\lambda + t\,\varepsilon_K\big)} + \varepsilon_K + 2\,\varepsilon_Z.$$

Proof. We first prove that none of the messages exchanged during the protocol leaks information about W. Then, we show that in our protocol Bob's decision on whether to accept or reject neither leaks information about W.

Because R_{B} is sampled independently of X_W, and by the chain rule, it follows that

$$H_{\min}(X_W|WE_1[U_{\mathrm{A}} \cdot T_{\mathrm{A}}]_q) \geq H_{\min}(X_W|WE_\circ) - q.$$

By assumption on the parameters in the statement of the proposition, i.e. that $H_{\min}(X_W|WE_\circ) > q + k_Z$, and by the properties of Ext it follows that

$$d(Z_{\mathrm{A}}|WE_2[U_{\mathrm{A}} \cdot T_{\mathrm{A}}]_q) \leq d(Z_{\mathrm{A}}|S_{\mathrm{A}}WE_1[U_{\mathrm{A}} \cdot T_{\mathrm{A}}]_q) \leq \varepsilon_Z.$$

By the fact that U_{B} and V_{B} are sampled independently, the following also holds

$$d(Z_{\mathrm{A}}|WE_3[U_{\mathrm{A}} \cdot T_{\mathrm{A}}]_q) \leq \varepsilon_Z.$$

Then, by security of the one-time pad, by the fact that Eve cannot gain information on W by computing Q_{B}, and by assumption that $\rho_{WE_\circ} = \rho_W \otimes E_\circ$,

$$\tfrac{1}{2}\|\rho_{WE_4} - \rho_W \otimes \rho_{E_4}\|_1 \leq \tfrac{1}{2}\|\rho_{WE_3Q_{\mathrm{A}}} - \rho_W \otimes \rho_{E_3Q_{\mathrm{A}}}\|_1 \leq \varepsilon_Z.$$

This completes the first part of the proof.

It remains to show that Bob's decision to accept or reject cannot leak (a substantial amount of) information about W. To show this, we make the following case distinction. In case $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$, the security proof applies and Bob rejects except with probability $\delta \leq 3 \cdot 2^{-q} + \frac{1}{2}\sqrt{2^q(\lambda + t\,\varepsilon_K)}$. It now immediately follows that

$$\tfrac{1}{2}\|\rho_{WE_4} - \rho_{WE}\|_1 \leq \delta, \quad \text{and} \quad \tfrac{1}{2}\|\rho_W \otimes \rho_{E_4} - \rho_W \otimes \rho_E\|_1 \leq \delta.$$

Hence, in case $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$ (by the triangle inequality),

$$\tfrac{1}{2}\|\rho_{WE} - \rho_W \otimes \rho_E\|_1 \leq \varepsilon_Z + 2\delta.$$

We now turn to the case $\mu_{\mathrm{A}} = \mu_{\mathrm{B}}$ and we analyze for two disjoint events. Conditioned on $M_{\mathrm{A}} \neq M_{\mathrm{B}}$, the strengthened version of the security statement applies, i.e.

$$\delta' \leq 3 \cdot 2^{-q} + \tfrac{1}{2}\sqrt{2^q\big(\lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}]\big)},$$

and again by applying the triangle inequality, we obtain

$$\tfrac{1}{2}\|\rho_{WE|M_{\mathrm{A}}\neq M_{\mathrm{B}}} - \rho_W \otimes \rho_{E|M_{\mathrm{A}}\neq M_{\mathrm{B}}}\|_1 \leq \varepsilon_Z + 2\delta'.$$

Secondly, we analyze for the event $M_{\mathrm{A}} = M_{\mathrm{B}}$. Nevertheless, we start this analysis without conditioning on $M_{\mathrm{A}} = M_{\mathrm{B}}$. (We'll condition on this event later in the proof.) Since S_{A} is sampled at random and independently of X_W, and since $H_{\min}(X_W|WE_\circ) > k_Z$, it follows that

$$d(Z_{\mathrm{A}}|S_{\mathrm{A}}WE_\circ) < \varepsilon_Z\,.$$

By the chain rule (and the independent choice of S_{A}),

$$H_{\min}(X_W|Z_{\mathrm{A}}WE_2) \geq H_{\min}(X_W|WE_\circ) - q > k_K\,,$$

and thus

$$d(K_{\mathrm{B}}|R_{\mathrm{B}}Z_{\mathrm{A}}S_{\mathrm{A}}WE_\circ) < \varepsilon_K\,.$$

From the above, and the independent choices of R_{B} and S_{A}, it follows that

$$\tfrac{1}{2}\|\rho_{K_{\mathrm{B}}Z_{\mathrm{A}}R_{\mathrm{B}}S_{\mathrm{A}}WE_\circ} - \rho_U \otimes \rho_{U'} \otimes \rho_{R_{\mathrm{B}}} \otimes \rho_{S_{\mathrm{A}}} \otimes \rho_W \otimes \rho_{E_\circ}\|_1 \leq \varepsilon_K + \varepsilon_Z.$$

where ρ_U is the fully mixed state on $\mathcal{H}_{K_{\mathrm{B}}}$ and $\rho_{U'}$ is the fully mixed state on $\mathcal{H}_{Z_{\mathrm{A}}}$, and therefore that

$$\tfrac{1}{2}\|\rho_{K_{\mathrm{B}}Z_{\mathrm{A}}WE_2} - \rho_U \otimes \rho_{U'} \otimes \rho_W \otimes \rho_{E_2}\|_1 \leq \varepsilon_K + \varepsilon_Z.$$

We now condition on $M_{\mathrm{A}} = M_{\mathrm{B}}$. Note that conditioned on this event, $K_{\mathrm{A}} = K_{\mathrm{B}}$ and $Z_{\mathrm{A}} = Z_{\mathrm{B}}$, and therefore, from here on, we omit the subscripts for these random variables and simply write K and Z. From Lemma 1 (noting that whether the event $M_{\mathrm{A}} = M_{\mathrm{B}}$ holds is determined by E_2), we get

$$\tfrac{1}{2}\|\rho_{KZWE_2|M_{\mathrm{A}}=M_{\mathrm{B}}} - \rho_U \otimes \rho_{U'} \otimes \rho_W \otimes \rho_{E_2|M_{\mathrm{A}}=M_{\mathrm{B}}}\|_1 \leq \frac{\varepsilon_K + \varepsilon_Z}{\Pr[M_{\mathrm{A}} = M_{\mathrm{B}}]}.$$

U_{B} and V_{B} are chosen uniformly at random and independent of the rest (and also independently of the event $M_{\mathrm{A}} = M_{\mathrm{B}}$). Furthermore, since E is computed from (KZE_4) alone, it follows that

$$\tfrac{1}{2}\|\rho_{WE|M_{\mathrm{A}}=M_{\mathrm{B}}} - \rho_W \otimes \rho_{E|M_{\mathrm{A}}=M_{\mathrm{B}}}\|_1 \leq \frac{\varepsilon_K + \varepsilon_Z}{\Pr[M_{\mathrm{A}} = M_{\mathrm{B}}]}.$$

We now combine the analyses for the two disjoint events, and conclude that in case $\mu_{\mathrm{A}} = \mu_{\mathrm{B}}$,

$$\begin{aligned}
&\tfrac{1}{2}\|\rho_{WE} - \rho_W \otimes \rho_E\|_1 \\
&\quad \leq \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}]\, \tfrac{1}{2}\|\rho_{WE|M_{\mathrm{A}}\neq M_{\mathrm{B}}} - \rho_W \otimes \rho_{E|M_{\mathrm{A}}\neq M_{\mathrm{B}}}\|_1 \\
&\qquad\quad + \Pr[M_{\mathrm{A}} = M_{\mathrm{B}}]\, \tfrac{1}{2}\|\rho_{WE|M_{\mathrm{A}}= M_{\mathrm{B}}} - \rho_W \otimes \rho_{E|M_{\mathrm{A}}= M_{\mathrm{B}}}\|_1 \\
&\quad = \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}]\, (\varepsilon_Z + 2\delta') + \varepsilon_K + \varepsilon_Z \\
&\quad \leq \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}] \left[\varepsilon_Z + 6 \cdot 2^{-q} + \sqrt{2^q\big(\lambda + t\,\varepsilon_K / \Pr[M_{\mathrm{A}} \neq M_{\mathrm{B}}]\big)}\right] + \varepsilon_K + \varepsilon_Z \\
&\quad \leq 6 \cdot 2^{-q} + \sqrt{2^q\big(\lambda + t\,\varepsilon_K\big)} + \varepsilon_K + 2\,\varepsilon_Z.
\end{aligned}$$

Note that we have computed two upper bounds on $\frac{1}{2}\|\rho_{WE} - \rho_W \otimes \rho_E\|_1$, for two distinct cases: $\mu_{\mathrm{A}} \neq \mu_{\mathrm{B}}$ and $\mu_{\mathrm{A}} = \mu_{\mathrm{B}}$. Obviously, the weaker (larger) upper bound holds in both cases, and we finally conclude that

$$\tfrac{1}{2}\|\rho_{WE} - \rho_W \otimes \rho_E\|_1 \leq 6 \cdot 2^{-q} + \sqrt{2^q\big(\lambda + t\,\varepsilon_K\big)} + \varepsilon_K + 2\,\varepsilon_Z.$$

6 The Fuzzy Case

Up to here, we assumed a scenario where Alice and Bob share *identical* copies of the session key X_W. Let us now consider the "fuzzy" case, where Alice and Bob hold keys that are only close in some sense, but not necessarily equal. This kind of scenario naturally arises when Alice and Bob obtain their session keys in the presence of noise. For simplicity and with our application (Section 7) in mind, we use the Hamming distance to measure closeness between keys.

Consider the following simple approach. Let Bob's key be called X_W. Before executing the authentication scheme, Bob sends some error correcting information (like the syndrome of X_W with respect to some error correcting code) to Alice, so that she can correct the errors in her key, X'_W, or vice versa. Unfortunately, Eve may of course also modify this error-correcting information, so that Alice might not correct X'_W correctly, in which case our scheme is not guaranteed to work. However, as proved in [9], this approach *does* work if one uses alternating-extraction-based instantiations of look-ahead extractors. For this solution to work it is important that both X_W and X'_W have sufficient min-entropy, and that Bob sends the error correcting information to Alice (i.e. the error-correction information must be sent in the same direction as the seed for the look-ahead extractor). The same holds in our setting where Eve is allowed to have quantum side information.

One subtlety is that the error correcting information must not leak information about W, to preserve the privacy property. Exactly this problem is addressed in [8], and is generalized to the quantum setting in [10]. Note that it is straightforward to upper bound the min-entropy loss in X_W (and X'_W) due to error correction: by the chain rule this is at most the bitsize of the error-correction information.

7 Application: Password-Based Identification in the Bounded Quantum Storage Model

Our main application is to password-based identification in the bounded quantum storage model. Damgård *et al.* proposed in [4] two password-based identification schemes, *Q-ID* and *Q-ID*$^+$. The former is truly password based but does not protect against a man-in-the-middle attack, whereas the latter is secure against a man-in-the-middle attack but is not truly password-based, because the "User" U and "Server" S need to additionally share a secret high-entropy key[3]. We sketch here how our authentication scheme leads to a truly password-based identification scheme in the bounded quantum storage model with security against man-in-the-middle attacks.

The idea of *Q-ID* and *Q-ID*$^+$ is as follows. U sends n BB84 qubits $H^\theta|x\rangle = H^{\theta_1}|x_1\rangle \otimes \cdots \otimes H^{\theta_n}|x_n\rangle$ to S, who measures them in basis $c(w) \in \{0,1\}^n$, where w is the common password and c is some appropriate code with large minimal distance d. Then, U announces the basis $\theta \in \{0,1\}^n$ used for the BB84 qubits. This allows U and S to compute the string x_w consisting of all the positions of x with $\theta_i = c(w)_i$, i.e., where U and S used the same basis. Then, U needs to convince S that he indeed knows (the same) string x_w. Damgård *et al.* show a way to do this which is guaranteed to not leak any information on w to a potentially dishonest U or S. Security against a dishonest U holds unconditionally, whereas security against a dishonest S holds in the bounded quantum storage model (where S is assumed to have limited quantum storage). At the core of the latter proof is a lower bound on the min-entropy of x_w from the dishonest server's point of view, which follows from the uncertainty relation from [5].

To make the protocol secure against man-in-the-middle attacks, some way is needed to protect the (classical and quantum) communication against tampering. In order to detect tampering with the communicated qubits, U and S choose a random sample of the qubits and verify that on those no tampering took place. In order to detect tampering with the classical communication, Damgård *et al.* propose to use a so-called extractor MAC. Such a MAC is similar to a standard information-theoretic MAC, and as such requires a high-entropy key, but is also an extractor. The way in which this extractor MAC is used in *Q-ID*$^+$ allows to re-use the high-entropy key.

7.1 Our Approach

Our approach of obtaining security against man-in-the-middle attacks without a high-entropy key is now simply to do the authentication of the classical communication by applying protocol `AUTH` of Section 4, using x_w as weak session key. Our privacy property guarantees that the authentication does not leak information on the password w. We stress that previous schemes for authentication based on weak keys would (potentially) leak here information on w.

[3] The high entropy key is only needed to protect against a man-in-the-middle attack, security against dishonest U and S only relies on the password and holds even if the dishonest party knows the high entropy key.

There are a couple of subtleties to be taken care of with our approach. If the quantum communication is noisy (which it is in realistic scenarios) or if the man-in-the-middle attacker modifies some of the qubits (but few enough so that he is not detected) or θ, then U and S's versions of x_w are not identical. Thus, we are in the fuzzy case. As discussed in Section 6, this is not a problem as long as the error-correcting information is sent from Bob to Alice, which means from S to U in the identification setting, and as long as we have lower bounds on both U and S's versions of x_w (from the attacker's point of view). The first requirement is easily taken care of, we just perform the error correction in the required direction; from S to U. In order to guarantee that both versions of x_w have sufficient min-entropy (the analysis of Damgård *et al.* only guarantees min-entropy in U's version), we modify the scheme as follows. Instead of measuring the BB84 qubits in basis $c(w)$, S measures them in a *random* basis $\hat{\theta}$ and announces the difference $r = c(w) \oplus \hat{\theta}$. Then, U and S update the code c by shifting every code word by r, so that with respect to the updated code c', S has actually measured the BB84 qubits in basis $c'(w)$. This trick has also been used in [3], though for a different reason, and has no real effect on the analysis of the scheme. However, as we show below, it enables us to argue that also S's version of x_w has lower-bounded min-entropy, and therefore the authentication of the classical messages is guaranteed to work, which implies security of our password-based identification scheme.

Recall that security against a dishonest U or a dishonest S requires that the dishonest party can exclude at most one possibility for the password w (in one execution of the attack); indeed, this is the best we can hope for, because the dishonest party can always try to guess w. For password-based man-in-the-middle security, we require that the attacker can exclude at most *two* possibilities for the password. Again, this is the best we can hope for, because in a man-in-the-middle attack, the attacker can (but of course does not have to) individually attack U and S, and in both attacks he can try to guess w. This is the man-in-the-middle security that we achieve with our scheme.

We first outline our scheme below and then argue (informally) why it is secure. From here, we use upper case letters for the random variables that describe the values x, θ, w, etc. in a (purified) execution of the protocol. It follows from the analysis of Q-ID$^+$ in [4] (which still applies under the shifted-codeword modification outlined above) that there exists a W' (independent of W) such that unless $W' = W$, there is min-entropy in X restricted to I_W from Eve's point of view.

For X' we reason as follows. Consider two possibilities for W; say w_1 and w_2. We focus on the positions where $c(w_1) \neq c(w_2)$ (which will be the same positions when replacing c by c'). We now fix θ; the following will hold for any choice of θ (chosen by U). From the uncertainty relation of [5] it follows that, approximately,

$$H_{\min}(X'_{12}|\hat{\Theta}) \geq d/2,$$

where X'_{12} is the restriction of X' to the positions where $c(w_1) \neq c(w_2)$, and remember that d represents the minimum distance of c. Because X' and $\hat{\Theta}$ are independent of W, and, in turn, R is determined by $\hat{\Theta}$ and W, we have that

1. U picks $x, \theta \in_R \{0,1\}^n$ and sends the n-qubit state $H^\theta|x\rangle$ to S.
2. S picks $\hat{\theta} \in_R \{0,1\}^n$ and measures $H^\theta|x\rangle$ in basis $\hat{\theta}$. Let x' be the outcome. S computes and sends $r := \hat{\theta} \oplus c(w)$ to U. We define $c'(w) := c(w) \oplus r$ and $I_w := \{i : \theta_i = c'(w)_i\}$.
3. U sends θ and $f \in_R \mathcal{F}$ to S.
4. S picks $g \in \mathcal{G}$, $j \in_R \mathcal{J}$ and a random subset $T \subset \{1, \ldots, n\}$ of size ℓ, computes $s := syn_j(x'|_{I_w})$ and $test' := x'|_T$, and sends g, j, s and T to U.
5. U sets $test := x|_T$, recovers $x'|_{I_w}$ from $x|_{I_w}$ with the help of s, and sends $test$ and $z := f(x'|_{I_w}) \oplus g(w)$ to S.
6. Using weak key $x'|_{I_w}$, U authenticates all communicated classical messages, i.e. $r, \theta, f, g, j, s, T, test, z$, using `AUTH`, towards S.
7. S accepts if and only if (1) `AUTH` accepts, (2) $test$ coincides with $test'$ wherever the bases coincide (up to some allowed noise level), and (3) $z = f(x'|_{I_w}) \oplus g(w)$.

$H_{\min}(X'_{12}|\hat{\Theta}WR) \geq d/2$. To additionally condition on Eve's quantum system E, we apply the storage bound q and conclude that $H_{\min}(X'_{12}|\hat{\Theta}WRE) \geq d/2 - q$. Since F is independently chosen, we may additionally condition on F.

Let $\tilde{\Theta}$ be the (possibly) adversarially modified version of θ, which is sent in step 3. The adversary obtains $\tilde{\Theta}$ as a quantum measurement on FRE, so we may condition on $\tilde{\Theta}$ instead of E without lowering the bound. Now, because of the conditioning on $\hat{\Theta}\tilde{\Theta}WFR$, we can replace X'_{12} by the pair $X'_1X'_2$ in the min-entropy bound, where X'_1 consists of the positions where $\tilde{\Theta} = c'(w_1)$ and similarly X'_2. (The entropy cannot decrease by not restricting to the positions $c(w_1) \neq c(w_2)$ anymore.) Thus,

$$H_{\min}(X'_1X'_2|\hat{\Theta}\tilde{\Theta}WFR) \geq d/2 - q,$$

and therefore in particular

$$H_{\min}(X'_1X'_2|\tilde{\Theta}FR) \geq d/2 - q.$$

This holds for any w_1 and w_2, so that the entropy splitting lemma [4] implies the existence of W'' (independent of W), so that unless $W'' = W$, there is lower-bounded min-entropy in X' restricted to I_W from Eve's point of view after step 3. Note that at the point where $X'|_{I_W}$ is actually used to run protocol `AUTH`, Eve will have obtained additional information (i.e. during step 4 and 5). However, it is not hard to upper-bound the min-entropy loss in $X'|_{I_W}$ due to this additional information, so that with the right choice of parameters there is still lower bounded min-entropy.

We have argued that both $X|_{I_W}$ and $X'|_{I_W}$, or, respectively X'_W and X_W in the terminology of Section 6, have lower-bounded min-entropy from Eve's point of view. Furthermore, in our proposed identification scheme above, S sends the error-correcting information to U. Together, this guarantees the security of `AUTH` when applied in the fuzzy case. Although in the original protocol (*Q-ID*$^+$) the error-correction information is sent in the other direction, reversing this direction is allowed because the authentication makes sure that no message is modified.

Now, security follows from the analysis of $Q\text{-}ID^+$ [4] (as well as from [3] regarding the shifted-codeword modification).

Acknowledgment

We would like to thank Krzysztof Pietrzak for interesting discussions and valuable comments regarding this work.

References

1. Bouman, N.J., Fehr, S.: Secure authentication from a weak key, without leaking information (full version). Cryptology ePrint Archive (2011), http://eprint.iacr.org/2011/034
2. Chandran, N., Kanukurthi, B., Ostrovsky, R., Reyzin, L.: Privacy amplification with asymptotically optimal entropy loss. In: STOC 2010: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 785–794. ACM, New York (2010)
3. Damgård, I., Fehr, S., Lunemann, C., Salvail, L., Schaffner, C.: Improving the security of quantum protocols via commit-and-open. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 408–427. Springer, Heidelberg (2009)
4. Damgård, I., Fehr, S., Salvail, L., Schaffner, C.: Secure identification and QKD in the bounded-quantum-storage model. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 342–359. Springer, Heidelberg (2007)
5. Damgård, I.B., Fehr, S., Renner, R., Salvail, L., Schaffner, C.: A tight high-order entropic quantum uncertainty relation with applications. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 360–378. Springer, Heidelberg (2007)
6. De, A., Portmann, C., Vidick, T., Renner, R.: Trevisan's extractor in the presence of quantum side information. arXiv (2009), http://arxiv.org/abs/0912.5514
7. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
8. Dodis, Y., Smith, A.: Correcting errors without leaking partial information. In: STOC 2005: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 654–663. ACM, New York (2005)
9. Dodis, Y., Wichs, D.: Non-malleable extractors and symmetric key cryptography from weak secrets. In: STOC 2009: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, pp. 601–610 (2009)
10. Fehr, S., Schaffner, C.: Randomness extraction via δ-biased masking in the presence of a quantum attacker. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 465–481. Springer, Heidelberg (2008)
11. Kanukurthi, B., Reyzin, L.: Key agreement from close secrets over unsecured channels. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 206–223. Springer, Heidelberg (2009)
12. König, R., Renner, R., Schaffner, C.: The operational meaning of min- and max-entropy. IEEE Transactions on Information Theory 55(9), 4337–4347 (2009)
13. Renner, R.: Security of Quantum Key Distribution. Ph.D. thesis, ETH Zürich (Switzerland) (2005)
14. Renner, R., König, R.: Universally composable privacy amplification against quantum adversaries. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 407–425. Springer, Heidelberg (2005)

15. Renner, R., 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)
16. Renner, R., Wolf, S.: The exact price for unconditionally secure asymmetric cryptography. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 109–125. Springer, Heidelberg (2004)
17. Van De Graaf, J.: Towards a formal definition of security for quantum protocols. Ph.D. thesis, Univ. de Montreal, Quebec, Canada (1998)

A Security and Instantiation of MAC

To construct a MAC with look-ahead security, we adopt the construction given in [9]. Because our look-ahead security definition, Definition 8, is slightly weaker than the one given in [9] (in that both μ_{A} and μ_{B} are fixed), we obtain a better security parameter, as argued below.

With respect to a different aspect, the requirement on the MAC for our construction is somewhat stronger, because we need a "universal" MAC which is $(\epsilon, \lambda+\epsilon)$-look-ahead secure *for any* $\epsilon \geq 0$ (and some λ). (This requirement stems from the proof of Lemma 2.) It turns out that the construction from [9] in the light of our weaker security definition does satisfy this property.

Proposition 1. *For any positive integers m and ℓ, there exists a family of functions $\{\mathsf{MAC}_k : \{0,1\}^m \to \{0,1\}^s\}$, indexed by keys $k \in (\{0,1\}^\ell)^t$, that is $(\epsilon, 2^{-\ell}+\epsilon)$ look-ahead secure for any $\epsilon > 0$, where $t = 4m$ and $s = 2m\ell$.*

The proof of the statement that MAC_k is $(\epsilon, 2^{-\ell}+\epsilon)$ look-ahead secure for any $\epsilon > 0$ largely follows the proof of Lemma 15 Appendix E.3 of [9] (and still applies in the quantum setting). However, our modification (of fixing both μ_{A} and μ_{B} before executing `DW-MAC`) overcomes the need for a union bound over all possible messages μ_{B} in that original proof, and hence saves us a factor of 2^m.

For completeness, we very briefly describe the idea of the construction here. $\mathsf{MAC}_k(\mu)$ outputs some of the blocks k_i of the key $k = (k_1, \ldots, k_t)$; where the choice of this subset is determined by μ. Furthermore, the construction guarantees that for any two distinct messages μ and μ', there exists an index $i_\circ < t$ such that $\mathsf{MAC}_k(\mu)$ outputs more blocks k_i with $i > i_\circ$ than $\mathsf{MAC}_k(\mu')$ does. From the look ahead property, it follows that given $k'_1, \ldots, k'_{i_\circ}$, the remaining blocks $k_{i_\circ+1}, \ldots, k_t$ are (close to) random. Then, from the choice of $i_\circ$ and from the chain rule we conclude that when given $\mathsf{MAC}_{k'}(\mu')$, the tag $\mathsf{MAC}_k(\mu)$ still contains at least (nearly) ℓ bits of min-entropy.

Secret Keys from Channel Noise

Hadi Ahmadi and Reihaneh Safavi-Naini

Department of Computer Science, University of Calgary, Canada
{hahmadi,rei}@ucalgary.ca

Abstract. We study the problem of unconditionally secure Secret Key Establishment (SKE) when Alice and Bob are connected by two noisy channels that are eavesdropped by Eve. We consider the case that Alice and Bob do not have any sources of initial randomness at their disposal. We start by discussing special cases of interest where SKE is impossible and then provide a simple SKE construction over binary symmetric channels that achieves some rates of secret key. We next focus on the Secret Key (SK) capacity and provide lower and upper bounds on this capacity. We prove the lower bound by proposing a multi-round SKE protocol, called the *main protocol*. The main protocol consists of an initialization round and the repetition of a two-round SKE sub-protocol, called the *basic protocol*. We show that the two bounds coincide when channels do not leak information to the adversary. We apply the results to the case that communicants are connected by binary symmetric channels.

1 Introduction

In cryptography, it is commonly assumed that parties have access to sources of randomness for their randomized protocols. It is also common to assume that this randomness is *perfect*, represented as a sequence of independently and uniformly random bits. Noting that, in many scenarios, the distribution of the random source is either biased or unknown, Dodis and Spencer [10] initiated the study of building cryptographic primitives using *imperfect* random sources. They focussed on symmetric-key encryption and message authentication and showed that in both cases the corresponding sources do not require perfect randomness.

In practice, generating randomness with high entropy needs specialized hardware and/or software as well as access to complex processes that could be hard to obtain, e.g., when devices with low computational resources are considered. A natural question is then whether the need for a separate random source can be eliminated from a particular cryptographic task. Obviously, cryptography is not possible without randomness. For devices with communication capability however, channel noise is an attractive *resource* for providing randomness.

Physical communication channels are noisy and can be viewed as potential resources to produce randomness. Wyner's pioneering work [18] showed that channel noise can be used to provide perfect security in message transmission. This work started a long line of research that relies on channel noise for constructing cryptographic primitives and it shares the vision of Crépeau and Kilian [7] that,

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 266–283, 2011.

"Noise, on the other hand, breeds disorder, uncertainty, and confusion. Thus, it is the cryptographer's natural ally."

Wyner's work and, to our knowledge, all cryptographic systems that use noisy channels as a resource also assume access to sources of initial randomness. In this paper, we initiate the study of cryptographic systems without making this assumption. We consider the case that the algorithms have hardwired and public constant strings, such as IDs, and the only resource for randomness is channel noise. One may ask whether, in such a setting, a particular cryptographic primitive exists and, if it does, whether it is sufficiently efficient to be of practical interest. We focus on the basic task of Secret Key Establishment (SKE) in the presence of a passive adversary and pose the following question:

Question 1. *Can Alice and Bob establish a shared secret key, without having access to initial randomness, by communicating over noisy channels that leak information to an eavesdropping adversary, Eve? In the case of a positive answer, are there efficient constructions to generate secret keys in practice?*

To the best of our knowledge, this paper is the first work to consider SKE with no initial randomness.

1.1 Our Work

We focus on Question 1 and study SKE over a pair of independent Discrete Memoryless Broadcast Channels (DMBCs). We refer to this setup as 2DMBC. SKE in this setup has been studied in [2]; however, again, it was assumed that Alice and Bob have access to initial randomness. We assume Alice and Bob have fixed strings, **a** and **b**, respectively. We also assume a full-duplex model of communication where, in *each communication round*, Alice and Bob send sequences of the same length. This communication model is used to simplify the presentation of our results; the results can be easily adapted to half duplex-channels where, in each communication round, either Alice or Bob sends a sequence.

Impossibility results: Beyond doubt, SKE without initial randomness is impossible if the channels between the parties are noise free. In Section 3, we discuss special cases of 2DMBC where SKE is impossible despite the existence of noise in the system. These special cases include (1) one-way communication, (2) when one DMBC is completely noise free, and (3) when one DMBC is noisy but returns two identical outputs. We note that the possibility of SKE in the above cases has been already proved [8,12,9] with the assumption that initial randomness is available to the parties.

SKE Construction: We give a positive answer to Question 1 by considering an example scenario where each DMBC consists of two independent Binary Symmetric Channels (BSCs). We propose a two-round SKE construction that uses three simple primitives, a von Neumann randomness extractor, a binary error-correcting code, and a universal hash function. The protocol works as follows. In round 1, Alice sends a constant (all-zero) sequence to Bob; Bob receives a noisy string and uses the von Neumann extractor to derive a uniformly random binary sequence from it. In round 2, Bob splits the uniform sequence into two

sub-sequences, encodes them separately, and sends the codewords to Alice. Alice decodes her received sequence to find the two sub-sequences. Finally, Alice and Bob apply universal hashing to the sub-sequences to derive a secure secret key.

Bounds on the SK capacity: We formalize the 2DMBC model and focus on the general description of a SKE protocol over a 2DMBC. We define the *Secret Key (SK) capacity* of a 2DMBC as the highest SK rate that all possible SKE protocols can achieve. This leads to the following question:

Question 2. *What is the SK capacity of a given 2DMBC?*

Towards answering Question 2, we provide lower and upper bounds on the SK capacity of a 2DMBC. We prove the lower bound by showing that there exists a SKE construction to achieve it. We describe a multi-round SKE protocol, referred to as the *main protocol*, that consists of an *initialization round*, followed by repeated use of a two-round protocol, which we call the *basic protocol*.

The initialization round bootstraps the main protocol by providing Alice and Bob with some pieces of "independent randomness". By independent randomness, we mean a random variable that is independent of all variables collected by other parties. The randomness is derived from channel noise and is required for executing one iteration of the basic protocol. Each iteration of the basic protocol uses the fresh randomness derived in the previous iteration, and simultaneously serves two purposes: it (1) derives new pieces of independent randomness for Alice and Bob (for the next iteration), and (2) derives a part of the secret key. To accomplish these two purposes, the basic protocol uses two new deterministic primitives, which we refer to as *secure block code* and *secure equipartition*, respectively. Each iteration of the basic protocol achieves a fixed key rate. During the initialization round however, no secret key bit is derived. Since the channel uses in the initialization round can be amortized over the number of the consecutive invocations of the basic protocol, the SK rate tends towards that of a single basic protocol execution. Compared to other possible ways of key establishment (see Section 1.2 for an example), the protocol described in this paper achieves the highest rate, hence resulting in a tighter lower bound on the SK capacity.

The lower bound shows that positive SK rates are achievable when both DMBCs are in favor of the legitimate parties. More interestingly, it shows that this condition, although sufficient, is not necessary and *there are cases where both DMBCs are in favor of Eve, yet it is possible to establish secure shared key.*

We also provide an upper bound on the SK capacity and show that the lower and the upper bounds coincide in the case that the channels do not leak any information to the adversary. This corresponds to the problem of common randomness generation over independent noisy channels, studied in [15], where the common randomness capacity was derived.

Discussion: The communication scenario considered in this paper naturally occurs in real life. All physical channels are noisy and in most cases, esp., in wireless communication, they are easy to eavesdrop. Assuming no initial randomness is also natural when communicating nodes, e.g., mobile devices, do not have access to specialized hardware and complex processes. Our results show that, in

the absence of initial randomness, nodes can start with constant strings such as their pre-stored IDs and "distill" randomness from channel noise.

Our work initiates a new direction of research: existence and construction of cryptographic primitives when the only resource for randomness is channel noise. We note that converting a cryptographic primitive that uses noisy channel as a resource and allows Alice and Bob to have sources of initial randomness, to the case that they do not have such a source is not straightforward.

The lower bound proof given in this paper uses an existential argument. However, attempts to design efficient while optimal primitives for secure equipartition and secure block code can be directly applied to the main SKE protocol design to achieve SK rates close to the lower bound. This is an interesting direction for future research similar to the work in [5] that attempts to apply theoretical SKE results in [18,12] in practice.

It is remarkable that the SKE construction given for binary symmetric channels can be viewed as a relaxed version of the main protocol where a simplified one-round basic protocol is used only once. The von Neumann extractor plays the role of (secure) equipartition in deriving independent randomness while the combination of coding and universal hashing is to replace the secure block code. Of course, using these efficient but non-optimal primitives does not let SK rates reach close enough to the lower bound. We discuss this more clearly by comparing the construction SK rates with the lower bound results.

1.2 Related Work

The problem considered in this paper has relations to a part of prior work, in particular, secure message transmission and key agreement over noisy channels, key agreement using correlated randomness, and common randomness generation over noisy channels. In the following, we briefly clarify these relations.

Exploiting channel noise to provide security functionalities is pioneered by Wyner [18] who proposed an alternative to Shannon's model of secure communication [14]. Wyner's work initiated a long line of research on utilizing channel noise to construct information theoretically secure cryptographic primitives including SKE [1,8,11,12,13], Oblivious Transfer (OT) [7], and Bit Commitment (BC) schemes [4]. In all these works however, access to initial randomness is assumed and removing this assumption will require revisiting the results and examining the existence of the primitives.

Maurer [12], concurrently with Ahlswede and Csiszár [1], studied the problem of key agreement over a public discussion channel when Alice and Bob have initial correlated randomness, where they derived lower and upper bounds on the SK capacity. Key agreement using correlated randomness and a one-way noisy channel has been discussed in [11,13].

The following two works are closely related to the setting in this paper, while neither can provide a solution to the problem. Venkatesan and Anantharam [15] considered shared randomness generation over a pair of independent channels and acquired the common randomness capacity. The authors noted that their results could not be applied to the case where the channels are eavesdropped

by Eve – the setting that is considered in this paper. In [2], we considered SKE in the 2DMBC setup and provided bounds on the SK capacity. That work, however, assumed the availability of free independent randomness without which the proofs will not be valid. Assuming no initial randomness, one may of course use the results in [2] to design a protocol as follows. Alice and Bob first execute an initialization round to derive the required amount of independent randomness. Next, they run the protocol in [2] to establish a secret key. Compared to this, our main protocol potentially increases the SK rate up to two times, through iteration. The particular novelty of the basic protocol is that it combines the dual tasks of secure key derivation and randomness generation.

1.3 Notation

We use calligraphic letters ($\mathcal{X}$), uppercase letters (X), and lowercase letters (x) to denote finite alphabets, Random variables (RVs), and their realizations over sets, respectively. $\mathcal{X}^n$ is the set of all sequences of length n (so called n-sequences) with elements from $\mathcal{X}$. $X^n = (X_1, X_2, \ldots, X_n) \in \mathcal{X}^n$ denotes a random n-sequence in $\mathcal{X}^n$. In case there is no confusion about the length, we use $\mathbf{X}$ to denote a random sequence and $\mathbf{x}$ to denote a realization in $\mathcal{X}^n$. While describing a multiple round protocol, we may use $X^{n:r}$ (or $\mathbf{X}^{:r}$) to indicate a random n-sequence that is sent, received, or obtained in round r. '||' denotes the concatenation of two sequences. For a value x, we use $(x)_+$ to show $\max\{0, x\}$ and, for an integer N, we use $[N]$ to show the set of integers $\{1, 2, \ldots, N\}$. All logarithms are in base 2 and, for $0 \leq p \leq 1$, $h(x) = -p \log p - (1-p)\log(1-p)$ denotes the binary entropy function.

1.4 Paper Organization

Section 2 describes SKE over 2DMBCs and delivers the security definitions. In Section 3, we provide the impossibility results and the simple SKE construction over BSCs. Section 4 summarizes our main results on the SK capacity. In Section 5, we describe the main protocol that achieves the lower bound. Section 6 studies the SKE results for the case of BSCs and Section 7 concludes the paper.

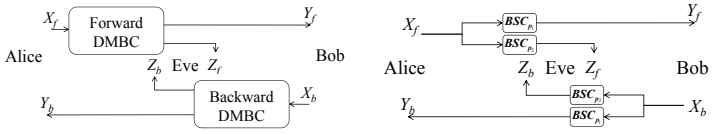
2 Problem Statement

The 2DMBC setup is shown in Fig. 1(a). There is a forward DMBC from Alice to Bob and Eve, denoted by $(\mathcal{X}_f, \mathcal{Y}_f, \mathcal{Z}_f, P_{Y_f,Z_f|X_f})$, and a backward DMBC from Bob to Alice and Eve, denoted by $(\mathcal{X}_b, \mathcal{Y}_b, \mathcal{Z}_b, P_{Y_bZ_b|X_b})$. The parties have deterministic computation systems.

To establish a secret key, Alice and Bob follow a SKE protocol with t communication rounds where, in round r, each channel is used n_r times. The protocol is defined by a sequence of deterministic function pairs, $(f_r, g_r)_{r=1}^{t-1}$, and a pair of (deterministic) key derivation functions (ϕ_A, ϕ_B) such that

$$f_r : \mathcal{Y}_f^{\sigma_{r-1}} \to \mathcal{X}_f^{n_r}, \qquad \phi_A : \mathcal{Y}_f^n \to \mathcal{S} \cup \{\perp\}, \tag{1}$$

$$g_r : \mathcal{Y}_b^{\sigma_{r-1}} \to \mathcal{X}_b^{n_r}, \qquad \phi_B : \mathcal{Y}_b^n \to \mathcal{S} \cup \{\perp\}, \tag{2}$$



(a) General 2DMBC (b) Independent BSCs

Fig. 1. The 2DMBC setup (a) in general and (b) in the case of independent BSCs

where $\sigma_j = \sum_{i=0}^{j} n_i$, $\perp$ indicates the error symbol, and $n = \sigma_{t-1}$ is the total number of channel uses. The protocol takes as input a pair, $(\mathbf{a}, \mathbf{b}) \in \mathcal{X}_f^{n_0} \times \mathcal{X}_b^{n_0}$, of constant and publicly known sequences. In a communication round r, Alice and Bob send the n_r-sequences $\mathbf{X}_f^{:r}$ and $\mathbf{X}_b^{:r}$ and receive $\mathbf{Y}_b^{:r}$ and $\mathbf{Y}_f^{:r}$, respectively. Eve receives $(\mathbf{Z}_f^{:r}, \mathbf{Z}_b^{:r})$. The input sequences are calculated as

$$\mathbf{X}_f^{:r} = \begin{cases} \mathbf{a}, & r = 0 \\ f_r(V_A^{:r-1}) & 1 \le r \le t-1 \end{cases}, \quad \mathbf{X}_b^{:r} = \begin{cases} \mathbf{b}, & r = 0 \\ g_r(V_B^{:r-1}) & 1 \le r \le t-1 \end{cases}. \tag{3}$$

$V_A^{:r-1}$, $V_B^{:r-1}$, and $V_E^{:r-1}$ are, respectively, the views of Alice, Bob and Eve, at the end of round $r-1$, i.e.,

$$V_A^{:r-1} = (\mathbf{Y}_b^{:i})_{i=1}^{r-1}, \quad V_B^{:r-1} = (\mathbf{Y}_f^{:i})_{i=1}^{r-1}, \quad \text{and} \quad V_E^{:r-1} = (\mathbf{Z}_f^{:i}, \mathbf{Z}_b^{:i})_{i=1}^{r-1}. \tag{4}$$

We have not included constants and deterministic functions that are applied to the variables in the views, since they do not contain any information (randomness). When the t rounds of communication are completed, Alice and Bob calculate their secret keys respectively as

$$S_A = \phi_A(V_A^{:t-1}), \quad \text{and} \quad S_B = \phi_B(V_B^{:t-1}). \tag{5}$$

Let $View_E = V_E^{:t-1}$ be Eve's view at the end of the protocol.

Definition 1. *For $R_{sk} \ge 0$ and $0 \le \delta \le 1$, the SKE protocol Π is* (R_{sk}, δ)-secure *if there exists a random variable $S \in \mathcal{S}$ such that the following requirements are satisfied:*

$$\textit{Randomness:} \quad \frac{H(S)}{n} \ge R_{sk} - \delta, \tag{6a}$$

$$\textit{Reliability:} \quad \Pr(S_A = S_B = S) \ge 1 - \delta, \tag{6b}$$

$$\textit{Secrecy:} \quad \frac{H(S|View_E)}{H(S)} \ge 1 - \delta. \tag{6c}$$

Definition 2. *The* Secret-Key (SK) capacity C_{sk} *is defined as the largest $R_{sk} \ge 0$ such that, for any arbitrarily small $\delta > 0$, there exists an (R_{sk}, δ)-secure SKE protocol.*

3 SKE in Special Cases of 2DMBC

3.1 Impossibility Results for Special Cases

We revisit a number of well-studied SKE scenarios that can be viewed as special cases of 2DMBC. We argue that, without initial randomness available to parties, SKE is impossible in these cases irrespective of the channel specification.

One-way communication: Consider a case that one of the DMBCs, say the backward DMBC, always returns constant values at its outputs. This implies one-way communication over the forward channel. Irrespective of the protocol, Alice will never have a single bit of randomness in her view and, without randomness, she cannot have a secret key. Note that this special case is essentially the one-way DMBC setting of Csiszár and Körner [8], with the difference that no initial randomness is provided to the parties.

One channel is noiseless and public: Without loss of generality, assume that the backward DMBC has this property. For any SKE protocol as described in Section 2, we have $\mathbf{X}_b^{ir} = \mathbf{Y}_b^{ir} = \mathbf{Z}_b^{ir}$ for each round r. This suggests that, overall, Eve's view includes Alice's view (see (4)). Eve can simply use Alice's key derivation function ϕ_A on her view to calculate S_A. This setting is proved to allow positive SK rates when parties have access to initial randomness [12].

One channel is noisy but returns two identical outputs: Assume that this property holds for the backward DMBC. In this case, $\mathbf{X}_b^{ir}$ may be different from the outputs and we only have $\mathbf{Y}_b^{ir} = \mathbf{Z}_b^{ir}$. This is sufficient to argue that Eve's view includes Alice's view; hence, the impossibility of SKE.

3.2 An SKE Protocol for Binary Symmetric Channels

Assume that the 2DMBC consists of four independent binary symmetric channels (BSCs) as illustrated in Fig. 1(b). The main channels have bit error probability p_1, while both Eve's channels have bit error probability p_2. We describe a two-round SKE construction that uses the primitives described below.

The von Neumann randomness extractor [16]: This extractor takes a binary sequence of even length and output a variable length sequence that has uniform distribution. For an input Bernoulli sequence $\mathbf{Y} = (Y_1Y_2, Y_3Y_4, \ldots, Y_{m-1}Y_m)$ of even length m, where $P(Y_i = 1) = p$, the von Neumann extractor divides the sequence into $m/2$ pairs of bits and uses the following mapping on each pair

$$00 \to \Lambda, \qquad 01 \to 0, \qquad 10 \to 1, \qquad 11 \to \Lambda,$$

where Λ represents no output. The output sequence is the concatenation of the mapped bits. It is easy to observe that the extractor is computationally efficient and the output bits are independently and uniformly distributed.

While the von Neumann extractor does not return a fixed-length output, it can be used to design a function $Ext : \{0,1\}^m \to \{0,1\}^l \cup \{\perp\}$ that derives a

l-bit uniform string from an m-bit Bernoulli sequence. The Ext function runs the von Neumann extractor on the m-bit sequence $\mathbf{Y}$. If the output length is less l, it returns $\perp$; otherwise, it returns the first l bits of the output. The probability that, for an m-bit Bernoulli sequence with $P(Y_i) = p$, Ext returns $\perp$ equals

$$\Pr(\mathcal{E}rr_{ext}) = \sum_{i=0}^{l-1} \binom{\frac{m}{2}}{i} (2p(1-p))^i \, (1-2p(1-p))^{\frac{m}{2}-i}. \tag{7}$$

An (n, k) binary error correcting channel code: We denote the encoding and the decoding functions by $Enc : \{0,1\}^k \rightarrow \{0,1\}^n$ and $Dec : \{0,1\}^n \rightarrow \{0,1\}^k$, respectively. There are efficient (n, k) error correcting codes that can correct nearly up to $t = (n-k)/2$ bits of error. When used over a BSC with error probability p, the decoding error probability of such codes equals

$$\Pr(n_{err} > t) = \sum_{i=t+1}^{n} \binom{n}{i} p^i \, (1-p)^{n-i}. \tag{8}$$

Universal class of hash functions: A class $\mathcal{H}$ of (hash) functions $h : \mathcal{A} \rightarrow \mathcal{B}$ is universal [6], if for any distinct $x_1, x_2 \in \mathcal{A}$, the equality $h(x_1) = h(x_2)$ happens with probability at most $1/|\mathcal{B}|$, provided that h is uniformly at random selected from $\mathcal{H}$. For the purpose of our SKE construction design, we use the following universal class of hash function proposed in [17].

$$\mathcal{H} = \{h_c : GF(2^k) \rightarrow \{0,1\}^s, \quad c \in GF(2^k)\},$$

where $h_c(x)$ returns the first s bits of $c.x$, and the product is in the polynomial representation of $GF(2^k)$. The hashing function is efficient in time and in memory.

Protocol description: Using the above primitives, the SKE protocol proceeds as follows. Alice sends her constant sequence $\mathbf{X}_f = \mathbf{a} = (\underline{0})^m$ over the forward DMBC. Bob and Eve receive the m-sequences $\mathbf{Y}_f$ and $\mathbf{Z}_f$ (m is even). Bob views this as an m-bit Bernoulli sequence, $\mathbf{Y}_f = (Y_{f,1}, \ldots, Y_{f,m})$, with $P(Y_{f,i} = 1) = p_1$ and finds $\mathbf{U} = Ext(\mathbf{Y}_f)$. If $\mathbf{U} = \perp$, the error $\mathcal{E}rr_{ext}$ occurs; otherwise, Bob splits the l-bit $\mathbf{U}$ into two independent and uniform k-bit sequences $\mathbf{U}_1$ and $\mathbf{U}_2$, where $k = l/2$. He calculates the n-bit codewords $\mathbf{X}_{1b} = Enc(\mathbf{U}_1)$ and $\mathbf{X}_{2b} = Enc(\mathbf{U}_2)$ and sends them over the backward DMBC where Alice and Eve receive $(\mathbf{Y}_{1b}, \mathbf{Y}_{2b})$ and $(\mathbf{Z}_{1b}, \mathbf{Z}_{2b})$, respectively. Alice calculates the k-sequences $\hat{\mathbf{U}}_1 = Dec(\mathbf{Y}_{1b})$ and $\hat{\mathbf{U}}_2 = Dec(\mathbf{Y}_{2b})$. The error event $\mathcal{E}rr_{enc1}$ (resp. $\mathcal{E}rr_{enc2}$) occurs when $\hat{\mathbf{U}}_1 \neq \mathbf{U}_1$ (resp. $\hat{\mathbf{U}}_2 \neq \mathbf{U}_2$). Next, Alice and Bob use universal hashing for privacy amplification, i.e., to derive keys that are secure against Eve. The secret key is $S = h_C(\mathbf{U}_1)$ where $C = \mathbf{U}_2$. Bob calculates $S_B = S$ and Alice calculates $S_A = h_{\hat{C}}(\hat{\mathbf{U}}_1)$ where $\hat{C} = \hat{\mathbf{U}}_2$.

The above protocol provides Alice and Bob with s uniformly random bits of key. The rate of key establishment is calculated as the number of the key

bits divided by the number of channel uses, i.e., $R_{sk} = \frac{s}{m+2n}$. Due to lack of space, we omit the argument on reliability and secrecy of the construction. For a detailed analysis, we refer to the full version in [3].

Table 1 shows the construction parameters for SKE over BSCs with $p_1 = 0.1$ and $p_2 = 0.2$ when the secret key length is $s = 100$ and the security parameter δ has different values. According to this table, the achievable SK rate by this construction is about $R_{sk} = 0.015$ bits per channel use.

Table 1. The SKE parameters with respect to δ values for $s = 100$

δ	n	k	l	m	R_{sk}
10^{-1}	404	300	600	5230	0.0166
10^{-2}	458	330	660	5430	0.0158
10^{-3}	508	358	716	5590	0.0151
10^{-4}	560	388	776	5730	0.0146

Remark 1. Assuming the full-duplex communication model allows Alice and Bob to run, in parallel, another execution of the protocol in the reverse direction. This will double the SK rate achieved by this construction, i.e., $R_{sk} = 0.03$.

Remark 2. This construction is given to show the feasibility of efficient SKE with no initial randomness. Using more optimal primitives, one may achieve higher secret key rates.

4 Results on the SK Capacity

We provide lower and upper bounds on the SK capacity as defined in Section 2. Let the RVs X_f, Y_f, Z_f and X_b, Y_b, Z_b denote the channel probability distributions $P_{Y_f,Z_f|X_f}$ and $P_{Y_b,Z_b|X_b}$, respectively.

Theorem 1. *The SK capacity is lower bounded as*

$$C_{sk}^{2DMBC} \geq \max_{\mu \geq 0, P_{X_f}, P_{X_b}} \{Lbound_A + Lbound_B\}, \tag{9}$$

where

$$Lbound_A = \frac{1}{1+\mu}\left(\mu(I(Y_b;X_b) - I(Y_b;Z_b)) + \gamma_1(I(X_f;Y_f) - I(X_f;Z_f))_+\right), \tag{10}$$

$$Lbound_B = \frac{1}{1+\mu}\left(\mu(I(Y_f;X_f) - I(Y_f;Z_f)) + \gamma_2(I(X_b;Y_b) - I(X_b;Z_b))_+\right), \tag{11}$$

$$\gamma_1 = \min\{1, \frac{H(Y_b|X_b,Z_b) + \mu(H(Y_b|X_b) - H(X_f))}{I(X_f;Y_f)}\}, \tag{12}$$

$$\gamma_2 = \min\{1, \frac{H(Y_f|X_f,Z_f) + \mu(H(Y_f|X_f) - H(X_b))}{I(X_b;Y_b)}\}, \tag{13}$$

such that

$$H(Y_b|X_b,Z_b) > \mu H(X_f), \qquad I(X_f;Y_f) > \mu H(Y_b|X_b), \tag{14}$$

$$H(Y_f|X_f,Z_f) > \mu H(X_b), \qquad I(X_b;Y_b) > \mu H(Y_f|X_f). \tag{15}$$

Proof. See Section 5 and [3, Appendix A].

The lower bound (9) is achieved by the so-called main protocol, which consists of an initialization round followed by iteration of the so-called basic protocol. The full duplex channel allows Alice and Bob to run two instances of the basic protocol in parallel. These two instances achieve the key rates $Lbound_A$ and $Lbound_B$, respectively. The key rate achieved in the second round of the basic protocol depends on the DMBC parameters (i.e., $I(X_f;Y_f) - I(X_f;Z_f)$ and $I(X_b;Y_b) - I(X_b;Z_b)$), while that of the first round depends on the "inverse" DMBC parameters (i.e., $I(Y_f;X_f) - I(Y_f;Z_f)$ and $I(Y_b;X_b) - I(Y_b;Z_b)$). The real value μ is the ratio between the number of channel uses in the first and the second rounds. The real values γ_1 and γ_2 are to restrict the amount of achievable key rate as a function of the randomness obtained from channel noise.

When both DMBCs are in favor of Alice and Bob, i.e., $I(X_f;Y_f) - I(X_f;Z_f)$ and $I(X_b;Y_b) - I(X_b;Z_b)$ are positive, $Lbound_A$ and $Lbound_B$ will be positive by simply choosing $\mu = 0$. This implies a positive SK capacity. When the channels are in favor of Eve, the lower bound may remain positive if the inverse DMBCs are in favor of Alice and Bob. The study of the lower bound for BSCs in Section 6 shows clearly the existence positive SK rates in the latter case (see Fig. 2).

Theorem 2. *The SK capacity is upper bounded as*

$$C_{sk}^{2DMBC} \leq \max_{P_{X_f}, P_{X_b}} \{Ubound_A + Ubound_B\}, \text{ where} \tag{16}$$

$$Ubound_A = \min\{H(Y_b|X_b, Z_b), I(X_f;Y_f|Z_f)\}, \text{ and} \tag{17}$$

$$Ubound_B = \min\{H(Y_f|X_f, Z_f), I(X_b;Y_b|Z_b)\}. \tag{18}$$

Proof. See [3, Appendix B].

Theorem 3 shows that the two bounds coincide when the two DMBCs do not leak information. The resulting value matches the common randomness capacity of a pair of independent Discrete Memoryless Channels (DMCs), given in [15].

Theorem 3. *When the DMBCs do not leak information to Eve, the bounds coincide and the SK capacity equals*

$$C_{sk}^{2DMBC} = \max_{P_{X_f}, P_{X_b}} \{\min\{H(Y_b|X_b), I(X_f;Y_f)\} + \min\{H(Y_f|X_f), I(X_b;Y_b)\}\}. \tag{19}$$

Proof. See [3, Appendix C].

5 The Main SKE Protocol: Achieving the Lower Bound

We noted that the bound in Theorem 1 is achieved by the *main protocol*. The main protocol contains $2t + 1$ rounds and does not assume any initial randomness. The protocol starts with an initialization round (round 0) that provides Alice and Bob with some amount of independent randomness. This round is followed by t iterations of a two-round sub-protocol, called the *basic protocol*. The basic protocol takes some independent randomness from Alice and Bob and

returns to them a secret key part and some new independent randomness. The independent randomness that is produced in iteration $1 \leq r \leq t-1$ (resp. round 0) will be used in iteration $r+1$ (resp. iteration 1). The secret key parts are finally concatenated to give the final secret key. The main protocol relies on the existence of two primitives, referred to as *secure equipartition* and *secure block code*. In the following, we provide definitions and theorems to support the existence of these primitives, and then we describe the main protocol.

5.1 Preliminaries

Definition 3. *For a probability distribution P_X over the set $\mathcal{X}$, a sequence $x^n \in \mathcal{X}^n$ is called ϵ-typical if $|-\frac{1}{n}\log P(x^n) - H(X)| < \epsilon$, where $P(x^n) = \prod_{i=1}^{n} P(x_i)$.*

Definition 4. *An (n, M, ϵ)-*block code *for the DMC $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ is a set $\{(c_i, \mathcal{C}_i) : i \in [M]\}$ such that $c_i \in \mathcal{X}^n$, $(\mathcal{C}_i)_{i=1}^{M}$ partitions $\mathcal{Y}^n$, and $P_{Y|X}^n(Y^n = \mathcal{C}_i | X^n = c_i) \geq 1 - \epsilon$.*

We define a *secure block code* for a DMBC as a composition of a block code and a function that we refer to as a *key derivation function*, and is used to achieve secure shared key between two parties in the presence of an adversary.

Definition 5. *An (n, M, K, ϵ)-*secure block code *for the DMBC $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, P_{YZ|X})$ consists of an (n, M, ϵ)-block code for the DMC $(\mathcal{X}, \mathcal{Y}, P_{Y|X})$ as above, a partition of $(c_i)_{i=1}^{M}$ into $(\mathcal{K}_j)_{j=1}^{K}$, and a key derivation function $\phi_s : (c_i)_{i=1}^{M} \to [K]$ defined as $\phi_s(c_i) = j$ iff $c_i \in \mathcal{K}_j$, such that if X^n is uniformly selected from $(c_i)_{i=1}^{M}$ and $S = \phi_s(X^n)$ then $H(S|Z^n)/\log K \geq 1 - \epsilon$.*

Although the above definition of a secure block code as a primitive is new to the literature, the work on secure message transmission or key agreement over one-way DMBCs [18,8] implicitly studies the existence of such a primitive. The results in [18,8] let us conclude the following.

Lemma 1. *For any P_X, $R_c < I(X;Y)$, $R_{sc} < R_c - I(X;Z)$, and large enough n, there exists an (n, M, K, ϵ)-secure block code for the DMBC $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, P_{YZ|X})$ with ϵ-typical codewords c_i such that $M = \lfloor 2^{nR_c} \rfloor$, $K = \lfloor 2^{nR_{sc}} \rfloor$, and $\epsilon = \max\{2^{n(R_c - I(X;Y))}, 2^{n(R_{sc} - (R_c - I(X;Z)))}\} \to 0$.*

Proof. See [18, Theorem 2] and [8, Corrollary 1].

Lemma 1 indicates that, for the above DMBC, there exists a secure block code that achieves key rates up to $I(X;Y) - I(X;Z)$. In the following, we extend this result by showing that the number of such secure block codes is such that any X^n as input to the channel belongs to at least one of them.

Lemma 2. *For any P_X, $R_c < I(X;Y)$, $R_{sc} < R_c - I(X;Z)$, large enough $R' > H(X) - R_c$, and large enough n, there exist N (not necessarily distinct) (n, M, K, ϵ)-secure block codes for the DMBC $(\mathcal{X}, \mathcal{Y}, \mathcal{Z}, P_{YZ|X})$ with ϵ-typical codewords, such that $M = \lfloor 2^{nR_c} \rfloor$, $K = \lfloor 2^{nR_{sc}} \rfloor$, $N = \lfloor 2^{nR'} \rfloor$, and $\epsilon = \max\{2^{n(R_c - I(X;Y))}, 2^{n(R_{sc} - (R_c - I(X;Z)))}\} \to 0$; furthermore, the probability that a randomly selected ϵ-typical sequence $X^n \in \mathcal{X}^n$ belongs to at least one of the codes is at least $1 - e^{-\gamma}$, where $\gamma = 2^{n(R' + R_c - H(X) - \epsilon)} \to \infty$.*

Proof. See [3, Appendix D]

For a DMBC, a secure equipartition is a primitive to derive uniform randomness that is independent of both input and Eve's received sequence.

Definition 6. *An* (M,ϵ)-secure equipartition *of* $\mathcal{C}\subseteq\mathcal{Y}^n$ *w.r.t.* $c\in\mathcal{X}^n$ *over the DMBC* $(\mathcal{X},\mathcal{Y},\mathcal{Z},P_{YZ|X})$ *is an* (M,ϵ)*-equipartition of* $\mathcal{C}$ *over the DMC* $(\mathcal{X},\mathcal{Y},P_{Y|X})$ *and a randomness derivation function* $\psi_t:\mathcal{C}\to[M]\cup\perp$ *defined as* $\psi_t(y^n)=j$ *if* $y^n\in\mathcal{C}(j)$ *and* $\psi_t(y^n)=\perp$ *if* $y^n\in\mathcal{C}(e)$*, such that if* $X^n=c$ *and* $T=\psi_t(Y^n)$*, then* $H(T|X^n=c,Z^n)/\log M\geq 1-\epsilon$.

The following lemma shows the existence of a secure equipartition over the DMBC that achieves randomness rates up to $H(Y|XZ)$ bits per channel use.

Lemma 3. *For any* P_X*, typical* $c\in\mathcal{X}^n$*,* $\mathcal{C}\subseteq\mathcal{Y}^n$ *of size less than* $2^{nH(Y)}$*,* $R_{se}<H(Y|XZ)$*, and large enough* n*, there exists an* (M,ϵ)*-secure equipartition over the DMBC* $(\mathcal{X},\mathcal{Y},\mathcal{Z},P_{YZ|X})$ *such that* $M=\lfloor 2^{nR_{se}}\rfloor$ *and*

$$\epsilon=\frac{3I(Y;X,Z)h(\epsilon')}{H(Y|XZ)-\epsilon'}\to 0,\quad \textit{where}\quad \epsilon'=2^{n(R_{se}-H(Y|XZ))}.$$

Proof. See [3, Appendix E].

To describe of the main protocol, we shall use the notion of an inverse DMBC that implies a virtual channel defined as follows.

Definition 7. *Given a distribution* P_X*, for a DMBC* $(\mathcal{X},\mathcal{Y},\mathcal{Z},P_{YZ|X})$*, we define its corresponding inverse DMBC as* $(\mathcal{Y},\mathcal{X},\mathcal{Z},P_{XZ|Y})$ *where* $P_{XZ|Y}$ *is calculated from the joint distribution* P_{XYZ}.

5.2 Description of the Main Protocol

Let P_{X_f}, P_{X_b}, and μ be chosen such that the conditions (14) and (15) are satisfied. The conditions can be rephrased as

$$n_2H(Y_b|X_b,Z_b)\geq n_1(H(X_f)+\alpha),\quad n_2I(X_f;Y_f)\geq n_1(H(Y_b|X_b)+\alpha),\tag{20}$$
$$n_2H(Y_f|X_f,Z_f)\geq n_1(H(X_b)+\alpha),\quad n_2I(X_b;Y_b)\geq n_1(H(Y_f|X_f)+\alpha),\tag{21}$$

where $\alpha>0$ is a sufficiently small real constant, to be determined from δ, and n_1 and n_2 are sufficiently large positive integers such that $n_1=\mu n_2$, and $1/\alpha=o(\min\{n_1,n_2\})$; in other words, $2^{-\alpha\min\{n_1,n_2\}}$ approaches zero. Define

$$\begin{array}{lll}R_{1f}=H(X_f)-\alpha, & R_{cf}=I(X_f;Y_f)-\alpha, & R_{scf}=I(X_f;Y_f)-I(X_f;Z_f)-2\alpha,\\ R_{ef}=H(Y_f|X_f), & R_{ef}^{+}=H(Y_f|X_f)+2\alpha, & R_{sef}=H(Y_f|X_f,Z_f)-\alpha,\\ R_{scf^{-1}}=I(Y_f;X_f)-I(Y_f;Z_f)-2\alpha. & & \end{array}\tag{22}$$

$$\begin{array}{lll}R_{1b}=H(X_b)-\alpha, & R_{cb}=I(X_b;Y_b)-\alpha, & R_{scb}=I(X_b;Y_b)-I(X_b;Z_b)-2\alpha,\\ R_{eb}=H(Y_b|X_b), & R_{eb}^{+}=H(Y_b|X_b)+2\alpha, & R_{seb}=H(Y_b|X_b,Z_b)-\alpha,\\ R_{scb^{-1}}=I(Y_b;X_b)-I(Y_b;Z_b)-2\alpha. & & \end{array}\tag{23}$$

Each iteration of the two-round basic protocol uses the 2DMBC channel n_1 times in the first round and n_2 times in the second round; i.e. in total $n_1 + n_2$. In the second round, Alice (resp. Bob) sends two sequences of lengths n_{21A} and n_{22A} (resp. n_{21B} and n_{22B}), where $n_{21A} + n_{22A} (= n_{21B} + n_{22B}) = n_2$ and,

$$n_{21A} = \frac{1}{R_{cf}} \min\{n_2 R_{cf}, n_2 R_{seb} + n_1 R_{eb} - n_1 R_{1f}\}, \tag{24}$$

$$n_{21B} = \frac{1}{R_{cb}} \min\{n_2 R_{cb}, n_2 R_{sef} + n_1 R_{ef} - n_1 R_{1b}\}. \tag{25}$$

Using the above quantities, we define,

$$\begin{array}{lll} M_{1A} = \lfloor 2^{n_1 R_{cb}} \rfloor, & M_{21A} = \lfloor 2^{n_{21A} R_{cf}} \rfloor, & \\ K_{1A} = \lfloor 2^{n_1 R_{scb^{-1}}} \rfloor, & K_{21A} = \lfloor 2^{n_{21A} R_{scf}} \rfloor, & \\ N_A = \lfloor 2^{n_1 R^+_{eb}} \rfloor, & & \\ L_{1A} = \lfloor 2^{n_1 R_{1f}} \rfloor, & L_{2A} = \lfloor 2^{n_{21A} R_{cf} - n_1 R_{eb}} \rfloor, & L_A = L_{1A}.L_{2A}, \\ \Gamma_{21A} = \min\{L_A, \lfloor 2^{n_{21B} R_{seb}} \rfloor\}, & \Gamma_{22A} = \lfloor 2^{n_{22B} R_{seb}} \rfloor, & \Gamma_A = \Gamma_{21A}.\Gamma_{22A}. \end{array} \tag{26}$$

$$\begin{array}{lll} M_{1B} = \lfloor 2^{n_1 R_{cf}} \rfloor, & M_{21B} = \lfloor 2^{n_{21B} R_{cb}} \rfloor, & \\ K_{1B} = \lfloor 2^{n_1 R_{scf^{-1}}} \rfloor, & K_{21B} = \lfloor 2^{n_{21B} R_{scb}} \rfloor, & \\ N_B = \lfloor 2^{n_1 R^+_{ef}} \rfloor, & & \\ L_{1B} = \lfloor 2^{n_1 R_{1b}} \rfloor, & L_{2B} = \lfloor 2^{n_{21B} R_{cb} - n_1 R_{ef}} \rfloor, & L_B = L_{1B}.L_{2B}, \\ \Gamma_{21B} = \min\{L_B, \lfloor 2^{n_{21A} R_{sef}} \rfloor\}, & \Gamma_{22B} = \lfloor 2^{n_{22A} R_{sef}} \rfloor, & \Gamma_B = \Gamma_{21B}.\Gamma_{22B}. \end{array} \tag{27}$$

Using (22)-(26), one can observe that $L_A = \Gamma_A$ and $L_B = \Gamma_B$ in the above. Let the set $\mathcal{X}^{n_1}_{f,\epsilon} = \{\mathbf{x}_{f,1}, \ldots, \mathbf{x}_{f,L_{1A}}\}$ be obtained by independently selecting L_{1A} sequences in $\mathcal{X}^{n_1}_f$. Similarly define $\mathcal{X}^{n_1}_{b,\epsilon} = \{\mathbf{x}_{b,1}, \ldots, \mathbf{x}_{b,L_{1B}}\} \subseteq \mathcal{X}^{n_1}_b$. Let Alice and Bob have two fixed public integers $u_a \in [\Gamma_{21A}]$ and $u_b \in [\Gamma_{21B}]$ as well as two fixed public sequences $\mathbf{a} \in \mathcal{X}^{n_{22A}}_f$ and $\mathbf{b} \in \mathcal{X}^{n_{22B}}_b$, respectively. Let $u_{A,split} : [\Gamma_{21A}] \times [\Gamma_{22A}] \to [L_{1A}] \times [L_{2A}]$ and $u_{B,split} : [\Gamma_{21B}] \times [\Gamma_{22B}] \to [L_{1B}] \times [L_{2B}]$ be arbitrary bijective mappings.

Define the inverse forward DMBC $(\mathcal{Y}_f, \mathcal{X}_f, \mathcal{Z}_f, P_{X_f, Z_f|Y_f})$ and the inverse backward DMBC $(\mathcal{Y}_b, \mathcal{X}_b, \mathcal{Z}_b, P_{X_b, Z_b|Y_b})$ according to Definition 7. Letting $\epsilon = 2^{-\min(n_1, n_{21A}, n_{21B})\alpha} \to 0$ and $\gamma = 2^{n_1(\alpha-\epsilon)} \to \infty$, and using Lemmas 1, 2, and 3 we arrive at the following.

- For the inverse forward DMBC, there exist N_B $(n_1, M_{1B}, K_{1B}, \epsilon)$-secure block codes $\{d^j_{f,i}, \mathcal{D}^j_{f,i} : 1 \le i \le M_{1B},\ 1 \le j \le N_B\}$ with the key derivation functions $\phi^j_{s,B}$, such that a randomly selected ϵ-typical sequence in $\mathcal{Y}^n_f$ is in at least one of the codes with probability at least $1 - e^{-\gamma}$.
- For the inverse backward DMBC, there exist N_A $(n_1, M_{1A}, K_{1A}, \epsilon)$-secure block codes $\{d^j_{b,i}, \mathcal{D}^j_{b,i} : 1 \le i \le M_{1A},\ 1 \le j \le N_A\}$ with the key derivation functions $\phi^j_{s,A}$, such that a randomly selected ϵ-typical sequence in $\mathcal{Y}^n_b$ is in at least one of the codes with probability at least $1 - e^{-\gamma}$.

- For the forward DMBC, there exists an $(n_{21A}, M_{21A}, K_{21A}, \epsilon)$-secure block code $\{c_{f,i}, \mathcal{C}_{f,i} : 1 \leq i \leq M_{21A}\}$ with the key derivation function $\phi_{s,A}$; furthermore, for each $(c_{f,i}, \mathcal{C}_{f,i})$ there exists a (Γ_{21B}, ϵ)-secure equipartition $\{\mathcal{C}_{f,i}(e), \mathcal{C}_{f,i}(1), \ldots, \mathcal{C}_{f,i}(\Gamma_{21B})\}$ with the randomness derivation function ψ_B^i.
- For the backward DMBC, there exists an $(n_{21B}, M_{21B}, K_{21B}, \epsilon)$-secure block code $\{c_{b,i}, \mathcal{C}_{b,i} : 1 \leq i \leq M_{2B}\}$ with the key derivation function $\phi_{s,B}$; furthermore, for each $(c_{b,i}, \mathcal{C}_{b,i})$ there exists a (Γ_{21A}, ϵ)-secure equipartition $\{\mathcal{C}_{b,i}(e), \mathcal{C}_{b,i}(1), \ldots, \mathcal{C}_{b,i}(\Gamma_{21A})\}$ with the randomness derivation function ψ_A^i.
- For the forward DMBC, for $(\mathbf{a}, \mathcal{Y}_f)$, there exists a (Γ_{22B}, ϵ)-secure equipartition $\{\mathcal{Y}_f(e), \mathcal{Y}_f(1), \ldots, \mathcal{Y}_f(\Gamma_{22B})\}$ with the randomness derivation function ψ_B.
- For the backward DMBC, for $(\mathbf{b}, \mathcal{Y}_b)$, there exists a (Γ_{22A}, ϵ)-secure equipartition $\{\mathcal{Y}_b(e), \mathcal{Y}_b(1), \ldots, \mathcal{Y}_b(\Gamma_{22A})\}$ with the randomness derivation function ψ_A.

The initialization round (round 0): Alice and Bob send the constant n_2-sequences $\mathbf{X}_f^{:0} = (c_{f,u_a}||a)$ and $\mathbf{X}_b^{:0} = (c_{b,u_b}||b)$ over their channels and receive the noisy versions $\mathbf{Y}_b^{:0} = (\mathbf{Y}_{1b}||\mathbf{Y}_{2b})$ and $\mathbf{Y}_f^{:0} = (\mathbf{Y}_{1f}||\mathbf{Y}_{2f})$, respectively. Eve also receives $\mathbf{Z}_f^{:0}$ and $\mathbf{Z}_b^{:0}$. In this round, no secret key is established; however, to derive independent randomness, Alice and Bob calculate $U_A^{:0} = (\psi_A^{u_b}(\mathbf{Y}_{1b})||\psi_A(\mathbf{Y}_{2b}))$ and $U_B^{:0} = (\psi_B^{u_a}(\mathbf{Y}_{1f})||\psi_B(\mathbf{Y}_{2f}))$, respectively. They next calculate $(U_{1A}^{:0}, U_{2A}^{:0}) = u_{A,split}(U_A^{:0})$ and $(U_{1B}^{:0}, U_{2B}^{:0}) = u_{B,split}(U_B^{:0})$, where the first and the second parts are respectively used in the first and the second rounds of iteration 1.

The basic protocol (iteration $1 \leq r \leq t$): There are two rounds, $2r-1$ and $2r$, where the protocol uses the 2DMBC n_1 and n_2 times, respectively. In round $2r-1$, Alice and Bob send $\mathbf{X}_f^{:2r-1} = x_{f,U_{1A}^{:2r-2}}$ and $\mathbf{X}_b^{:2r-1} = x_{b,U_{1B}^{:2r-2}}$, and receive $\mathbf{Y}_b^{:2r-1}$ and $\mathbf{Y}_f^{:2r-1}$, respectively. Eve also receives $\mathbf{Z}_f^{:2r-1}$ and $\mathbf{Z}_b^{:2r-1}$.

Alice finds (I_A, J_A) such that $\mathbf{Y}_b^{:2r-1} = d_{b,I_A}^{J_A}$, i.e., the I_A-th codeword in the J_A-th secure block code over the inverse backward DMBC; similarly, Bob obtains (I_B, J_B) such that $\mathbf{Y}_f^{:2r-1} = d_{f,I_B}^{J_B}$. Round $2r-1$ may also be interpreted as follows. Alice and Bob have encoded $I_A \in [M_{1A}]$ and $I_B \in [M_{1B}]$ to the codewords $d_{b,I_A}^{J_A}$ and $d_{b,I_B}^{J_B}$; they have sent them over the inverse DMBCs but have not mentioned which block code they belong to. Thus, round $2r$ is primarily used for sending the block code labels, i.e., $J_A \in [N_A]$ and $J_B \in [N_B]$. That round is also used to send the pieces of randomness, $U_{2A}^{:2r-2} \in [L_{2A}]$ and $U_{2B}^{:2r-2} \in [L_{2B}]$, as well as the deterministic sequences, $\mathbf{a}$ and $\mathbf{b}$.

In the beginning of round $2r$, Alice and Bob respectively calculate $Q_A \in [M_{21A}]$ and $Q_B \in [M_{21B}]$ as (note that $M_{21A} = N_A.L_{2A}$ and $M_{21B} = N_B.L_{2B}$)

$$Q_A = L_{2A}J_A + U_{2A}^{:2r-2}, \quad \text{and} \quad Q_B = L_{2B}J_B + U_{2B}^{:2r-2}. \tag{28}$$

They next use the key derivation functions (in the secure block code) to calculate key parts $S_A^{:2r} = \phi_{s,A}(Q_A)$ and $S_B^{:2r} = \phi_{s,B}(Q_B)$. In this round, Alice and Bob send the n_2-sequences $\mathbf{X}_f^{:2r} = (c_{f,Q_A}||a)$ and $\mathbf{X}_b^{:2r} = (c_{b,Q_B}||b)$ and receive $\mathbf{Y}_b^{:2r} = (\mathbf{Y}_{1b}||\mathbf{Y}_{2b})$ and $\mathbf{Y}_f^{:2r} = (\mathbf{Y}_{1f}||\mathbf{Y}_{2f})$, respectively. Eve also receives $\mathbf{Z}_f^{:2r}$

and $\mathbf{Z}_b^{:2r}$. Using the secure block code for the forward DMBC, Bob obtains $\hat{Q}_A$ such that $\mathbf{Y}_{1f} \in \mathcal{C}_{f,\hat{Q}_A}$ and calculates $\hat{S}_A^{:2r} = \phi_{s,A}(\hat{Q}_A)$; similarly, Alice obtains $\hat{Q}_B$ such that $\mathbf{Y}_{1b} \in \mathcal{C}_{b,\hat{Q}_B}$ and calculates $\hat{S}_B^{:2r} = \phi_{s,B}(\hat{Q}_B)$. To produce randomness for the next iteration, Alice and Bob use their secure equipartitions to calculate $U_A^{:2r} = (\psi_A^{\hat{Q}_B}(\mathbf{Y}_{1b})||\psi_A(\mathbf{Y}_{2b}))$ and $U_B^{:2r} = (\psi_B^{\hat{Q}_A}(\mathbf{Y}_{1f})||\psi_B(\mathbf{Y}_{2f}))$, respectively. The randomness pieces are then split into $(U_{1A}^{:2r}, U_{2B}^{:2r}) = u_{A,split}(U_A^{:2r})$ and $(U_{1B}^{:2r}, U_{2B}^{:2r}) = u_{B,split}(U_B^{:2r})$. The above calculations are to derive independent randomness and secret key parts from round $2r$. The following is for deriving a key part out of round $2r-1$. Firstly, the parties calculate

$$\hat{U}_{2A}^{:2r-2} = \hat{Q}_A \mod (L_{2A}),\ \hat{J}_A = (\hat{Q}_A - \hat{U}_{2A}^{:2r-2})/L_{2A}, \tag{29}$$

$$\hat{U}_{2B}^{:2r-2} = \hat{Q}_B \mod (L_{2B}),\ \hat{J}_B = (\hat{Q}_B - \hat{U}_{2B}^{:2r-2})/L_{2B}. \tag{30}$$

The quantities $\hat{J}_A \in [N_A]$ and $\hat{J}_B \in [N_B]$ are used to find which secure block codes need to be considered over the inverse DMBCs in round $2r-1$. More precisely, Alice finds $\hat{I}_B$ such that $\mathbf{X}_f^{:2r-1} \in \mathcal{D}_{f,\hat{I}_B}^{\hat{J}_B}$ and Bob finds $\hat{I}_A$ such that $\mathbf{X}_b^{:2r-1} \in \mathcal{D}_{b,\hat{I}_A}^{\hat{J}_A}$. As for the establishment of the secret key part, Alice calculates $S_A^{:2r-1} = \phi_{s,A}^{J_A}(d_{b,I_A}^{J_A})$ and $\hat{S}_B^{:2r-1} = \phi_{s,B}^{\hat{J}_B}(d_{f,\hat{I}_B}^{\hat{J}_B})$, and Bob calculates $\hat{S}_A^{:2r-1} = \phi_{s,A}^{\hat{J}_A}(d_{b,\hat{I}_A}^{\hat{J}_A})$ and $S_B^{:2r-1} = \phi_{s,B}^{J_B}(d_{f,I_B}^{J_B})$. The total secret key part in iteration r is $\left(S_A^{:2r-1}, S_A^{:2r}, S_B^{:2r-1}, S_B^{:2r}\right)$. Overall, the main protocol uses the 2DMBC $n = (2t+1)(n_1+n_2)$ times to establish $S = (S_A^{:r}, S_B^{:r})_{r=1}^{2t}$. By following this protocol, Alice calculates $S_A = (S_A^{:r}, \hat{S}_B^{:r})_{r=1}^{2t}$ and Bob calculates $S_B = (\hat{S}_A^{:r}, S_B^{:r})_{r=1}^{2t}$. In [3, Appendix A], we show that the main algorithm satisfies the three requirements given in Definition 1 and achieves the lower bound in Theorem 1.

6 The SK Capacity for Binary Symmetric Channels

Consider the case that each DMBC consists of independent BSCs with error probabilities p_1 and p_2, i.e., the special case discussed in Section 3.2 (see Fig. 1(b)). Following the lower bound expression (9) in Theorem 1, and letting X_f and X_b to be uniform binary RVs, we conclude the following lower bound on the SK capacity in the case of BSCs, C_{sk}^{BSC}.

$$C_{sk}^{BSC} \geq 2 \max_{\mu \geq 0}\{Lbound\}, \text{ such that} \tag{31}$$

$$Lbound = \tfrac{1}{1+\mu}\left(\mu(h(p_1+p_2-2p_1p_2) - h(p_1)) + \gamma(h(p_2) - h(p_1))_+\right), \tag{32}$$

$$\gamma = \min\{1, \tfrac{h(p_1)}{1-h(p_1)} - \mu\}, \qquad \mu \leq \min\{h(p_1), \tfrac{1-h(p_1)}{h(p_1)}\}. \tag{33}$$

In general, $\mu \geq 0$ is a non-negative real number. However, we show in [3] that only three selections of μ, that is $\mu \in \{0, M_1, M_2\}$ (with M_1 and M_2 defined in (34)) can lead to the lower bound (31).

$$M_1 = \frac{h(p_1)}{1-h(p_1)} - 1 \quad \text{and} \quad M_2 = \min\{h(p_1), \frac{1-h(p_1)}{h(p_1)}\}, \tag{34}$$

In other words, the lower bound in (31) is simplified to

$$C_{sk}^{BSC} \geq 2 \max_{\mu \in \{0, M_1, M_2\}} \{Lbound\}. \tag{35}$$

This makes it easy to calculate the lower bound. Following the upper bound (16) in Theorem 2 for the above setting, we arrive at

$$C_{sk}^{BSC} \leq 2 \max_{P_{X_f}, P_{X_b}} \{Ubound_A, Ubound_B\}, \text{ where} \tag{36}$$

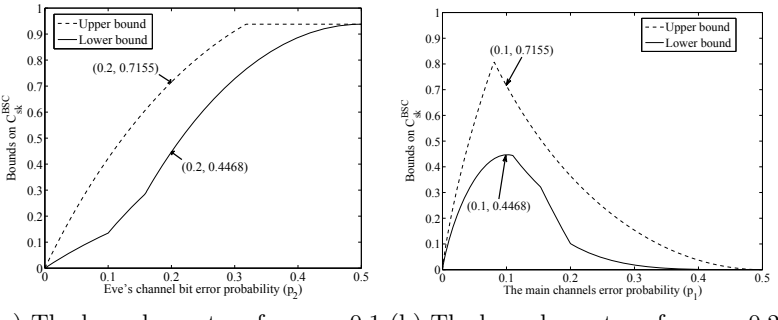
$$Ubound_A = \min\{h(p_1), H(Y_f|Z_f) - h(p_1)\}, \text{ and} \tag{37}$$

$$Ubound_B = \min\{h(p_1), H(Y_b|Z_b) - h(p_1)\}. \tag{38}$$

One can easily observe that, for uniform X_f and X_b, $Ubound_A$ and $Ubound_B$ reach their highest values, respectively. The upper bound is simplified as

$$C_{sk}^{BSC} \leq 2 \min\{h(p_1), h(p_1 + p_2 - 2p_1p_2) - h(p_1)\}. \tag{39}$$

Fig. 2 graphs the two bounds, (35) and (39), for different probability values p_1 and p_2. Fig. 2(a) illustrates the changes in the two bounds with respect to $0 \leq p_2 \leq 0.5$ when $p_1 = 0.1$. The bounds coincide when $p_2 = 0$ or when $p_2 = 0.5$. When $p_2 = 0$ all information sent over the 2DMBC is seen by Eve and SKE is impossible; so, both bounds equal zero. When $p_2 = .5$, the setup does not leak any information to Eve and, from Theorem 3, the two bounds are expected to coincide. Fig. 2(b) graphs the changes of the two bounds when $0 \leq p_1 \leq 0.5$ and $p_2 = 0.2$. When the main channels are noiseless ($p_1 = 0$) or completely noisy ($p_1 = 0.5$), the two bounds coincide at zero and so SKE is impossible. In the former case, no randomness exists in the system and, in the latter, there is no chance of reliable communication. The graphs also show the possibility of SKE even when both DMBCs are in favor of Eve. This can be observed in Fig. 2(a) for values of $0 < p_2 < 0.1$ and in Fig. 2(b) for values of $0.2) < p_1 < 0.5$.



(a) The bounds w.r.t p_2 for $p_1 = 0.1$ (b) The bounds w.r.t. p_1 for $p_2 = 0.2$

Fig. 2. The relationship between the two bounds with respect to p_1 and p_2

In Section 3.2, we have provided a simple SKE construction. For the values $p_1 = 0.1$ and $p_2 = 0.2$, the construction achieves the SK rate 3%. As depicted in Fig. 2, the two bounds on the SK capacity for these probability values are about 45% and 72%, respectively. This reveals how the example construction of Section 3.2 works far from optimal achievable rates. As noted earlier, one can improve the performance of the protocol by using more suitable primitives.

7 Conclusion

This paper has raised the question of building cryptographic functionalities over noisy channels when there is no initial randomness available to the parties of a system. We focused on two-party secret key establishment (SKE) where the communicants are connected by independent noisy broadcast channels that leak information to an adversary. We formalized the problem and defined the secret key capacity. We discussed some special cases where SKE is impossible, and then provided a concrete construction for binary symmetric channels. We obtained lower and upper bounds on the secret key capacity and showed that they coincide when the channels do not leak information to Eve. For the case of binary symmetric channels, we simplified the bounds and showed the gap between the rate achieved by the concerted construction and the rate proved to be achievable by optimal primitives. It would be interesting to design constructions with higher SK rates. Our work also suggests the question of the existence of other cryptographic primitives when channel noise is the only resource for randomness.

References

1. Ahlswede, R., Csiszár, I.: Common randomness in information theory and cryptography. Part I: secret sharing. IEEE Transaction Information Theory 39, 1121–1132 (1993)
2. Ahmadi, H., Safavi-Naini, R.: Secret key establishment over a pair of independent broadcast channels. In: International Symposium Information Theory and its Application (2010); Full version on the arXiv preprint server, arXiv:1001.3908
3. Ahmadi, H., Safavi-Naini, R.: Secret keys from channel noise. Technical Reports 2011/056, Cryptology ePrint archive, http://eprint.iacr.org/2011/056
4. Barros, J., Imai, H., Nascimento, A.C.A., Skludarek, S.: Bit commitment over Gaussian channels. In: IEEE International Symposium Information Theory, pp. 1437–1441 (2006)
5. Bloch, M., Barros, J., Rodrigues, M.R.D., McLaughlin, S.W.: Wireless information theoretic security. IEEE Transaction Information Theory 54, 2515–2534 (2008)
6. Carter, J.L., Wegman, M.N.: Universal Classes of Hash Functions. Journal of Computer and System Sciences 18, 143–154 (1979)
7. Crépeau, C., Kilian, J.: Weakening security assumptions and oblivious transfer. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 2–7. Springer, Heidelberg (1990)
8. Csiszár, I., Körner, J.: Broadcast channels with confidential messages. IEEE Transaction Information Theory 24, 339–348 (1978)

9. Csiszár, I., Narayan, P.: Common randomness and secret key generation with a helper. IEEE Transaction Information Theory 46, 344–366 (2000)
10. Dodis, Y., Spencer, J.: On the (non)universality of the one-time pad. In: IEEE Annual Symposium FOCS, pp. 376–388 (2002)
11. Khisti, A., Diggavi, S., Wornell, G.: Secret key generation with correlated sources and noisy channels. In: IEEE International Symposium Information Theory, pp. 1005–1009 (2008)
12. Maurer, U.: Secret key agreement by public discussion from common information. IEEE Transaction Information Theory 39, 733–742 (1993)
13. Prabhakaran, V., Eswaran, K., Ramchandran, K.: Secrecy via sources and channels - a secret key - secret message rate trade-off region. In: IEEE International Symposium Information Theory, pp. 1010–1014 (2008)
14. Shannon, C.E.: Communication theory of secrecy systems. Bell System Technical Journal 28, 656–715 (1948)
15. Venkatesan, S., Anantharam, V.: The common randomness capacity of a pair of independent discrete memoryless channels. IEEE Transaction Information Theory 44, 215–224 (1998)
16. von Neumannm, J.: Various techniques used in connection with random digits. National Bureau of Standards Applied Math Series 12, 36–38 (1951)
17. Wegman, M.N., Carter, J.L.: New hash functions and their use in authentication and set equality. Journal of Computer and System Sciences 22, 265–279 (1981)
18. Wyner, A.D.: The wire-tap channel. Bell System Technical Journal 54, 1355–1367 (1975)

Almost Optimum t-Cheater Identifiable Secret Sharing Schemes

Satoshi Obana

NEC
obana@bx.jp.nec.com

Abstract. In Crypto'95, Kurosawa, Obana and Ogata proposed a k-out-of-n secret sharing scheme capable of identifying up to t cheaters with probability $1-\epsilon$ under the condition $t \leq \lfloor (k-1)/3 \rfloor$. The size of share $|\mathcal{V}_i|$ of the scheme satisfies $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon^{t+2}$, which was the most efficient scheme known so far. In this paper, we propose new k-out-of-n secret sharing schemes capable of identifying cheaters. The proposed scheme possesses the same security parameters t, ϵ as those of Kurosawa *et al.*. The scheme is surprisingly simple and its size of share is $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon$, which is much smaller than that of Kurosawa *et al.* and is almost optimum with respect to the size of share; that is, the size of share is only one bit longer than the existing bound. Further, this is the first scheme which can identify cheaters, and whose size of share is independent of any of n, k and t. We also present schemes which can identify up to $\lfloor (k-2)/2 \rfloor$, and $\lfloor (k-1)/2 \rfloor$ cheaters whose sizes of share can be approximately written by $|\mathcal{V}_i| \approx (n \cdot (t+1) \cdot 2^{3t-1} \cdot |\mathcal{S}|)/\epsilon$ and $|\mathcal{V}_i| \approx ((n \cdot t \cdot 2^{3t})^2 \cdot |\mathcal{S}|)/\epsilon^2$, respectively. The number of cheaters that the latter two schemes can identify meet the theoretical upper bound.

Keywords: Secret Sharing, Cheater Identification, Reed-Solomon Code, Universal Hash.

1 Introduction

Secret sharing scheme is a cryptographic primitive in which a secret is divided into shares and distributed among participants in such a way that only a qualified set of participants can recover the secret. It is a fundamental building block for many cryptographic protocols and is often used in the general composition of secure multiparty computations. Because of their importance in cryptography it has been studied actively for more than three decades since the seminal papers by Shamir [23] and Blakley [3].

Cheating prevention is one of the main topics in secret sharing schemes. Tompa and Woll first considered a secret sharing scheme capable of detecting the presence of cheating when invalid shares are submitted in the secret reconstruction phase [25]. For the problem of detecting cheating, the upper bound of the size of share and efficient constructions have been actively studied so far [1,2,6,9,16,19].

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 284–302, 2011.

Secret sharing schemes that not only detect the presence of cheating but also identify cheaters who submit invalid shares are also a hot topic in this area. Rabin and Ben-Or proposed a k-out-of-n secret sharing scheme capable of identifying cheaters [21]. The size of share $|\mathcal{V}_i|$ of their scheme is $|\mathcal{V}_i| = |\mathcal{S}|^{3n-2}$ where $|\mathcal{S}|$ denotes the size of secret[1]. In [12], Kurosawa, Obana and Ogata showed that when the number of cheater t satisfies $t \leq \lfloor (k-1)/3 \rfloor$ the share size is greatly reduced compared to that of [21]. The size of share of their scheme is $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon^{t+2}$, which until now has been the most efficient scheme, despite the fact that the bit length of their scheme is still linear to the number of cheaters. The lower bound of share size is given by Kurosawa *et al.* as follows [12]:

$$|\mathcal{V}_i| \geq \frac{|\mathcal{S}| - 1}{\epsilon} + 1 \tag{1}$$

where ϵ denotes the successful cheating probability of cheaters. Though, the sizes of shares of all the existing schemes are far from the above bound.

In this paper, we first present efficient k-out-of-n threshold secret sharing schemes capable of identifying up to t cheaters under the condition $t \leq \lfloor (k-1)/3 \rfloor$. While this condition is the same as that of Kurosawa *et al.* [12], the share size is dramatically reduced compared to [12]. Namely, the share size of the first scheme satisfies $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon$ and is only one bit longer than the bound of eq. (1). We also present a scheme with the desired property that the successful cheating probability of cheaters can be determined without regard to the size of the secret, which is not the case in the first scheme. Further, we present k-out-of-n threshold schemes capable of detecting up to t cheaters such that $t \leq \lfloor (k-2)/2 \rfloor$ and $t \leq \lfloor (k-1)/2 \rfloor$, respectively. The numbers of cheaters these two schemes can identify reach the theoretical limit when k is even and for any k, respectively. The sizes of share of the schemes can be approximately written by $|\mathcal{V}_i| \approx (n \cdot t \cdot 2^{3t-1} \cdot |\mathcal{S}|)/\epsilon$ and $|\mathcal{V}_i| \approx ((n \cdot t \cdot 2^{3t})^2 \cdot |\mathcal{S}|)/\epsilon^2$, respectively, which are also much smaller than that of Kurosawa *et al.* despite the difference of their cheater identifiabilities.

We note that secret sharing schemes against cheating are strongly related to secure message transmission schemes as mentioned in [11,13]. Therefore, we believe that ideas used to construct proposed schemes will help to construct secure message transmission schemes.

The rest of the paper is organized as follows. In Section 2, we briefly review models of secret sharing schemes capable of identifying cheaters, and we discuss related work. In Section 3, we present almost optimum schemes which can identify up to $\lfloor (k-1)/3 \rfloor$ cheaters. In Sections 4 and 5, we give efficient schemes which can identify up to $\lfloor (k-2)/2 \rfloor$ cheaters and $\lfloor (k-1)/2 \rfloor$ cheaters, respectively. In Section 6, we summarize our work.

[1] Throughout the paper, we use notations $|\mathcal{X}|$ and $\mathtt{X}$ to denote the cardinality of a set $\mathcal{X}$ and a random variable over $\mathcal{X}$, respectively.

2 Preliminaries

2.1 Secret Sharing Schemes

In the model of secret sharing schemes, there are n participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ and a dealer D. The model consists of two algorithms: $\mathsf{ShareGen}$ and $\mathsf{Reconst}$. The share generation algorithm $\mathsf{ShareGen}$ takes a secret $s \in \mathcal{S}$ as input and outputs a list $(v_1, v_2, \ldots, v_n)$. Each $v_i \in \mathcal{V}_i$ is called a *share* and is given to a participant P_i. In a usual setting, $\mathsf{ShareGen}$ is invoked by the dealer. The secret reconstruction algorithm $\mathsf{Reconst}$ takes a list of shares and outputs a secret $s \in \mathcal{S}$.

The set of participants who are allowed to reconstruct the secret is characterized by an *access structure* $\Gamma \subseteq 2^{\mathcal{P}}$; that is, participants $P_{i_1}, \ldots, P_{i_k}$ are allowed to reconstruct the secret if and only if $\{P_{i_1}, \ldots, P_{i_k}\} \in \Gamma$ (for instance, the access structure of a k-out-of-n threshold secret sharing scheme is defined by $\Gamma = \{\mathcal{A} \mid \mathcal{A} \subseteq 2^{\mathcal{P}}, |\mathcal{A}| \geq k\}$.) A secret sharing scheme is called *perfect* if the following two conditions are satisfied for the output $(v_1, \ldots, v_n)$ of $\mathsf{ShareGen}(\hat{s})$ where the probabilities are taken over the random tape of $\mathsf{ShareGen}$.

1. if $\{P_{i_1}, \ldots, P_{i_k}\} \in \Gamma$ then $\Pr[\mathsf{Reconst}(v_{i_1}, \ldots, v_{i_k}) = \hat{s}] = 1$,
2. if $\{P_{i_1}, \ldots, P_{i_k}\} \notin \Gamma$ then $\Pr[\mathtt{S} = s \mid \mathtt{V}_{i_1} = v_{i_1}, \ldots, \mathtt{V}_{i_k} = v_{i_k}] = \Pr[\mathtt{S} = s]$ for any $s \in \mathcal{S}$.

We note that only perfect secret sharing schemes are dealt with in this paper.

2.2 t-Cheater Identifiable Secret Sharing Schemes

A secret sharing scheme capable of identifying cheaters was first presented by Rabin and Ben-Or [21]. They considered the scenario in which cheaters who do not belong to the access structure submit forged shares in the secret reconstruction phase. Such cheaters will succeed if they cannot be identified as cheaters in reconstructing the secret.

As with ordinary secret sharing schemes, this model consists of $\mathsf{ShareGen}$ and $\mathsf{Reconst}$. The share generation algorithm $\mathsf{ShareGen}$ is the same as that in the ordinary secret sharing schemes. Two types of secret reconstruction algorithms have been defined so far depending on whether identification of the cheater is done *privately* or *publicly*. We will use $\mathsf{Reconst}^{(\mathsf{pri})}$ and $\mathsf{Reconst}^{(\mathsf{pub})}$ to denote secret reconstruction algorithms which identify cheaters privately and publicly, respectively. A secret reconstruction algorithm $\mathsf{Reconst}^{(\mathsf{pri})}$ takes a share called a *base share* and a list of shares as input and outputs a pair of a secret and a set of cheaters; that is, if no cheater is identified $\mathsf{Reconst}^{(\mathsf{pri})}$ outputs a pair $(s, \emptyset)$ where s is a secret reconstructed. If $\mathsf{Reconst}^{(\mathsf{pri})}$ finds cheaters and the secret s can be reconstructed from valid shares submitted, it outputs (s, L) (where $s \in \mathcal{S}$ and $L \neq \emptyset$ is a set of cheaters submit invalid shares,) otherwise (i.e. if a secret cannot be reconstructed from valid shares,) it outputs $(\perp, L)$ where $\perp (\notin \mathcal{S})$ is a special symbol indicating that cheating was detected and, again, L is a set of cheaters. In $\mathsf{Reconst}^{(\mathsf{pri})}$, the base share becomes a basis for deciding whether a participant submitting a share to $\mathsf{Reconst}^{(\mathsf{pri})}$ is a cheater. On the other hand, $\mathsf{Reconst}^{(\mathsf{pub})}$ identifies cheaters without a trusted share: it takes a list of shares

as input and outputs a pair of a secret and a set of cheaters. We require that the algorithms $\mathsf{ShareGen}$ and $\mathsf{Reconst}$ satisfy the following *correctness* condition:

$$\Pr[(v_1, \ldots, v_n) \leftarrow \mathsf{ShareGen}(s);\ (\hat{s}, L) \leftarrow \mathsf{Reconst}(v_{i_1}, \ldots, v_{i_m}) : s = \hat{s} \wedge L = \emptyset] = 1$$

for any $s \in \mathcal{S}$, for any $i_1, \ldots, i_m$ such that $m \geq k$.

The security of the model can be formalized by the following simple game defined for any k-out-of-n threshold secret sharing scheme $\mathbf{SS} = (\mathsf{ShareGen}, \mathsf{Reconst})$ and for any (not necessarily polynomially bounded) Turing machine $\mathsf{A}^{(t)} = (\mathsf{A}_1^{(t)}, \mathsf{A}_2^{(t)})$, where $\mathsf{A}^{(t)}$ represents t cheaters $P_{i_1}, \ldots, P_{i_t}$ who try to cheat honest participants $P_{i_{t+1}}, \ldots, P_{i_m}$ where $m \geq k$.

$\mathsf{Game}(\mathbf{SS}, \mathsf{A}^{(t)})$
$s \leftarrow \mathcal{S}$; // according to the probability distribution over $\mathcal{S}$.
$(v_1, \ldots, v_n) \leftarrow \mathsf{ShareGen}(s)$;
$(i_1, \ldots, i_t) \leftarrow \mathsf{A}_1^{(t)}()$;
$(v'_{i_1}, \ldots, v'_{i_t}, i_{t+1}, \ldots, i_m) \leftarrow \mathsf{A}_2^{(t)}(v_{i_1}, \ldots, v_{i_t})$;

Cheaters P_{i_j} succeeds in cheating if $\mathsf{Reconst}$ fails to identify P_{i_j} as a cheater when a secret reconstructed is not identical to the original one. In the public model, we will denote successful cheating probability of P_{i_j} against $\mathbf{SS}^{(\mathsf{pub})}$ by $\epsilon(\mathbf{SS}^{(\mathsf{pub})}, \mathsf{A}^{(t)}, P_{i_j})$ where $\epsilon(\mathbf{SS}^{(\mathsf{pub})}, \mathsf{A}^{(t)}, P_{i_j})$ is define as follows:

$$\epsilon(\mathbf{SS}^{(\mathsf{pub})}, \mathsf{A}^{(t)}, P_{i_j}) = \Pr[(s', L) \leftarrow \mathsf{Reconst}^{(\mathsf{pub})}(v'_{i_1}, \ldots, v'_{i_t}, v_{i_{t+1}}, \ldots, v_{i_k}) : i_j \notin L].$$

On the other hand, in the private model, successful cheating probability of P_{i_j} is defined for each P_ℓ submitting base share. Therefore, we will define such probability $\epsilon(\mathbf{SS}^{(\mathsf{pri})}, \mathsf{A}^{(t)}, P_{i_j}, P_{i_\ell})$ by

$$\begin{aligned}&\epsilon(\mathbf{SS}^{(\mathsf{pri})}, \mathsf{A}^{(t)}, P_{i_j}, P_{i_\ell}) \\ &= \Pr[(s', L) \leftarrow \mathsf{Reconst}^{(\mathsf{pri})}(v_{i_\ell}, (v'_{i_1}, \ldots, v'_{i_t}, \overset{v_{i_\ell}}{\vee}, v_{i_{t+1}}, \ldots, v_{i_k})) : i_j \notin L]\end{aligned}$$

where the first argument v_{i_ℓ} of $\mathsf{Reconst}^{(\mathsf{pri})}$ denotes a base share. The probabilities are taken over the distribution of $\mathcal{S}$, and over the random tapes of $\mathsf{ShareGen}$ and $\mathsf{A}^{(t)}$. Note that the above game implicitly assumes *simultaneous secret reconstruction*; that is, all the participants submit their shares simultaneously to secret reconstruction algorithm in reconstructing the secret. Therefore, so-called "rushing adversary" who tries to forge its share *after* observing shares of honest participants is not allowed in this model.

Cheaters in this model can be classified into two classes: *non-critical cheaters* and *critical cheaters*. Non-critical cheaters only disclose their information to other cheaters or forge their shares in such a way that their forgeries do not cause the secret reconstruction algorithm to reconstruct a different secret from the original one. On the other hand, critical cheaters submit forged shares which cause the secret reconstruction algorithm to reconstruct a different secret from the original one. In this paper we focus on identifying only critical cheaters since

the goal of the cheaters in the models considered is to make other participants reconstruct an *invalid* secret. The formal definition of a critical cheater for public cheater identification models are given as follows:

Definition 1. *Let* $(v_1, \ldots, v_n)$ *be output of* $\mathsf{ShareGen}^{(\mathsf{pub})}(s)$. *A participants* P_j *who submit* v'_j *to* $\mathsf{Reconst}^{(\mathsf{pub})}$ *is called* a critical cheater *if and only if there exist* $i_1, i_2, \ldots, i_{k-1}$ *such that*

$$\Pr[(s', L) \leftarrow \mathsf{Reconst}^{(\mathsf{pub})}(v_{i_1}, \ldots, v_{i_{k-1}}, v'_j) : s' \neq s \wedge s' \in \mathcal{S}] \neq 0\,.$$

In the case of private cheater identification model, a critical cheater may vary according to a participant who submit base share.

Definition 2. *Let* $(v_1, \ldots, v_n)$ *be output of* $\mathsf{ShareGen}^{(\mathsf{pri})}(s)$. *A participants* P_j *who submit* v'_j *to* $\mathsf{Reconst}^{(\mathsf{pri})}$ *is called* a critical cheater against P_ℓ *if and only if there exist* $i_1, i_2, \ldots, i_{k-2}$ *such that*

$$\Pr[(s', L) \leftarrow \mathsf{Reconst}^{(\mathsf{pri})}(v_\ell, (v_{i_1}, \ldots, v_{i_{k-2}}, v'_j)) : s' \neq s \wedge s' \in \mathcal{S}] \neq 0\,.$$

Based on the above definition, we define the security of secret sharing schemes capable of identifying cheaters for both public and private models as follows:

Definition 3. *A* (k, n) *threshold secret sharing scheme* $\mathbf{SS}^{(\mathsf{pub})} = (\mathsf{ShareGen}^{(\mathsf{pub})}, \mathsf{Reconst}^{(\mathsf{pub})})$ *is called a* (t, ϵ) *cheater identifiable secret sharing scheme with public cheater identification if* $\epsilon(\mathbf{SS}^{(\mathsf{pub})}, \mathsf{A}^{(t)}, P_j) \leq \epsilon$ *for any* $\mathsf{A}^{(t)}$ *representing set of* t *or less cheaters* $\mathcal{P}$, *for any critical cheater* $P_j \in \mathcal{P}$.

Definition 4. *A* (k, n) *threshold secret sharing scheme* $\mathbf{SS}^{(\mathsf{pri})} = (\mathsf{ShareGen}^{(\mathsf{pri})}, \mathsf{Reconst}^{(\mathsf{pri})})$ *is called a* (t, ϵ) *cheater identifiable secret sharing scheme with private cheater identification if* $\epsilon(\mathbf{SS}^{(\mathsf{pri})}, \mathsf{A}^{(t)}, P_j, P_\ell) \leq \epsilon$ *for any* $\mathsf{A}^{(t)}$ *representing set of* t *or less cheaters* $\mathcal{P}$, *for any critical cheater* $P_i \in \mathcal{P}$ *and for any honest participant* P_ℓ.

We note that (t, ϵ) publicly cheater identifiable schemes for $\epsilon < 1$ exist only if $t \leq \lfloor (k-1)/2 \rfloor$ whereas $(k-1, \epsilon)$ cheater identifiable scheme with private cheater identification can be constructed. This is because cheaters can easily generate arbitrary number of consistent shares by invoking $\mathsf{ShareGen}$ with forged secret s' as input and distribute them among the cheaters in publicly cheater identifiable schemes. In this case, it is impossible to identify cheaters unless we can determine cheaters on a majority basis.

We also note that the model of (t, ϵ) cheater identifiable secret sharing scheme is different from that of the verifiable secret sharing (VSS for short) in the sense that the dealer is honest in the (t, ϵ) cheater identifiable secret sharing whereas the dealer may cheat in the VSS.

2.3 Related Work

In this subsection, we briefly review a known bound and constructions of (t, ϵ) cheater identifiable secret sharing schemes and related topics.

The capability to identify cheaters in secret sharing schemes was first pointed out by McEliece and Sarwate [15]. Namely, they observed that a list of shares of Shamir's (k, n) threshold secret sharing scheme constitutes a codeword of Reed-Solomon code. Therefore, if $k + 2t + 1$ shares containing up to t invalid shares are submitted in reconstructing a secret, the secret reconstruction algorithm can identify all cheaters with probability 1. However, this observation does not directly lead to constructing (t, ϵ) cheater identifiable secret sharing schemes since $k + 1$ or more shares are required to identify cheaters.

(t, ϵ) cheater identifiable (k, n) secret sharing scheme with private cheater identification are presented in various literature. Here, we will briefly review previous results. In [21,22], Rabin and Ben-Or presented a scheme on which they constructed a verifiable secret sharing scheme. The property of their scheme can be summarized by the following proposition:

Proposition 1 [21,22]. *There exists $(k-1, \epsilon)$ cheater identifiable (k, n) threshold secret sharing scheme with private cheater identification with parameter $|\mathcal{S}| = p$, $\epsilon = 1/p$, and $|\mathcal{V}_i| = p^{3n-2}$ where p is a prime power.*

Carpentieri proposed a scheme in which the size of share is reduced compared to [21,22]:

Proposition 2 [5]. *There exists $(k-1, \epsilon)$ cheater identifiable (k, n) threshold secret sharing scheme with private cheater identification with parameter $|\mathcal{S}| = p$, $\epsilon = 1/p$, and $|\mathcal{V}_i| = p^{k+2(n-1)}$ where p is a prime power.*

Ogata and Kurosawa proposed an elegant scheme in which the size of share is independent of n:

Proposition 3 [18]. *There exists $(k-1, \epsilon)$ cheater identifiable (k, n) threshold secret sharing scheme with private cheater identification with parameter $|\mathcal{S}| = p$, $\epsilon = (k-1)/(p-1)$, and $|\mathcal{V}_i| = p^{2k+1}$ where p is a prime power.*

We note that the schemes in [21,22] (Proposition 1) and [5] (Proposition 2) are secure even when cheater *knows* shares of $n-1$ participants whereas the scheme in [18] (Proposition 3) ensures security against cheaters who know at most $k-1$ shares.

With respect to a scheme with public cheater identification, Kurosawa, Obana and Ogata presented an efficient scheme whose share size only depends on the maximum number of cheaters [12]. The properties of their scheme can be summarized as follows:

Proposition 4 [12]. *If $t \leq \lfloor (k-1)/3 \rfloor$, there exists (t, ϵ) cheater identifiable (k, n) threshold secret sharing scheme with public cheater identification with parameter $|\mathcal{S}| = p, \epsilon = 1/q$ and $|\mathcal{V}_i| = p \cdot q^{t+2}$ where p, q are prime powers satisfying $q \geq n \cdot p - t$*[2].

In [12], Kurosawa *et al.* also showed a lower bound of share size for (t, ϵ) cheater identifiable secret sharing schemes with both publicly and privately cheater identification as follows:

[2] Though this limitation is not addressed in [12], it follows directly from Bush bound on orthogonal array of strength $t + 1$.

Proposition 5 [12]. *The size of share for* (t, ϵ) *cheater identifiable* (k, n) *threshold secret sharing schemes is lower bounded by* $|\mathcal{V}_i| \geq \dfrac{|\mathcal{S}| - 1}{\epsilon} + 1$.

However, share sizes of existing schemes are far from the above bound. Therefore, it was not clear whether the above bound is tight.

3 Publicly Cheater Identifiable Schemes for $t \leq \lfloor \frac{k-1}{3} \rfloor$

In this section, we present two efficient (t, ϵ) cheater identifiable (k, n) threshold secret sharing schemes with public cheater identification under the condition $t \leq \lfloor (k-1)/3 \rfloor$. The first scheme is almost optimum with respect to the share size; that is, the bit length of shares of the scheme is only one bit longer than the lower bound of Proposition 5. The second scheme, even though the share size is slightly larger than the first scheme, possesses a particular merit in that the successful cheating probability of cheaters can be chosen without regard to the size of the secret, which is the case neither in the first scheme nor in the scheme of [12].

As with the scheme in [12], the proposed scheme uses Reed-Solomon code to identify cheaters. The major difference between the scheme in [12] and the proposed scheme is as follows. In [12], a share of each participant consists of (1) a share of Shamir's (k, n) secret sharing for a secret, (2) a share of Shamir's (t, n) secret sharing scheme for a key of strongly universal hash functions of strength $t + 1$ (please refer to [24] for the definition,) and (3) a hash value of (1) under the key (2). Here, Reed-Solomon code is used in (2) to make cheaters impossible to alter the value of the key, which is used to examine the validity of shares (as pointed out in [15], (t, n) secret sharing scheme is equivalent to codeword of generalized Reed-Solomon code). Since the size of key of the strongly universal hash function of strength $t + 1$ is as large as $1/\epsilon^{t+1}$ the share size of the scheme in [12] grows linear with the number of cheaters. On the other hand, a share of the proposed scheme only consists of (1) a share of Shamir's (k, n) secret sharing for a secret, and (2) a hash value of (1) computed by a strongly universal hash function of strength $t+1$. Interestingly, the key used to compute hash values is not explicitly shared among the participants but is recovered from the hash values in the secret reconstruction phase by utilizing the error correction capability of Reed-Solomon code. This is made possible by choosing a strongly universal hash family based on polynomials over a finite field. Since the size of hash value is equal to $1/\epsilon$ in the proposed scheme, we see that the share size $|\mathcal{V}_i|$ of the proposed scheme satisfies $|\mathcal{V}_i| = |\mathcal{S}|/\epsilon$, which is independent of any of k, n and t. The detailed description of the first scheme is given in the next subsection.

3.1 An Almost Optimum Scheme

The share generation algorithm ShareGen and the secret reconstruction algorithm Reconst of the first scheme are described as follows where p and q are prime powers such that $q \geq n \cdot p$ and $\psi : GF(p) \times \{1, \ldots, n\} \to GF(q)$ is an injective function (e.g. $\psi(x, y) = (y - 1) \cdot p + x$ for prime numbers p, q.)

Share Generation: On input a secret $s \in GF(p)$, the share generation algorithm ShareGen outputs a list of shares $(v_1, \ldots, v_n)$ as follows:

1. Generate a random polynomial $f_s(x) \in GF(p)[X]$ of degree $k-1$ such that $f_s(0) = s$.
2. Generate a random polynomial $C(x) \in GF(q)[X]$ of degree t.
3. Compute $v_i = (f_s(i), C(\psi(f_s(i), i)))$ and output $(v_1, \ldots, v_n)$.

Secret Reconstruction and Cheater Identification: On input a list of m ($\geq k$) shares $((v_{s,i_1},\ v_{C,i_1}),\ \ldots,\ (v_{s,i_m},\ v_{C,i_m}))$, the secret reconstruction algorithm Reconst output a secret or a list of identities of cheaters as follows.

1. Reconstruct $\hat{C}(x)$ from $(v_{C,i_1}, \ldots, v_{C,i_m})$ using an error correction algorithm of generalized Reed-Solomon Code (e.g. Berlekamp algorithm.)
2. Check if $v_{C,i_j} = \hat{C}(\psi(v_{s,i_j}, i_j))$ holds (for $1 \leq j \leq m$.) If $v_{C,i_j} \neq \hat{C}(\psi(v_{s,i_j}, i_j))$ then i_j is added to the list of cheaters L.
3. If $|L| \leq m - k$ then reconstruct $f_s(x)$ from (k or more) shares v_{i_j} such that $i_j \notin L$ using Lagrange interpolation, and output $(f_s(0), L)$ if $\deg(f_s) \leq k-1$, otherwise Reconst output $(\perp, L)$. Reconst also output $(\perp, L)$ if $|L| > m - k$ holds.

Security of the proposed scheme can be summarized by the following theorem.

Theorem 1. *If $t \leq \lfloor (k-1)/3 \rfloor$ then the proposed scheme is a (t, ϵ) cheater identifiable secret sharing scheme with public cheater identification such that*

$$|\mathcal{S}| = p, \quad \epsilon = 1/q, \quad q \geq n \cdot p, \quad |\mathcal{V}_i| = p \cdot q \ (= |\mathcal{S}|/\epsilon)\,.$$

Proof. First, we show that the scheme is perfect. It is well known that $v_{s,i_1}, \ldots, v_{s,i_k}$ do not reveal any information about the secret since each $v_{s,i}$ is a share of Shamir's k-out-of-n secret sharing scheme. Further, it is easy to see that the knowledge about $v_{C,i}$ does not leak any information about the secret since the polynomial $C(x)$ is completely independent of the secret s.

Next we show that the scheme is (t, ϵ) cheater identifiable. The following two facts are key to prove (t, ϵ) cheater identifiability of the scheme:

1. $(C(x_1), C(x_2), \ldots, C(x_k))$ is a codeword of the Reed-Solomon Code with minimum distance $k - t$. Therefore, if $k - t > 2t$ (i.e. $t \leq \lfloor (k-1)/3 \rfloor$) then $C(x)$ can be reconstructed even when t points are forged.
2. A family of functions $\{C(x) \mid C(x) \in GF(q)[X],\ \deg(C(x)) \leq t\}$ is a strong class of universal hash functions $GF(q) \to GF(q)$ with strength $t+1$; that is, the following equality holds for any distinct $x_1, \ldots, x_t, x_{t+1} \in GF(q)$ and for any $y_1, \ldots, y_t, y_{t+1} \in GF(q)$.

$$\Pr[C(x_{t+1}) = y_{t+1} \mid C(x_1) = y_1, \ldots, C(x_t) = y_t] = 1/q\,. \tag{2}$$

Without loss of generality, we can assume $P_1, \ldots, P_t$ are cheaters who cooperatively try to fool the other participants by forging (part of) their shares. Suppose that P_1 is a critical cheater who is told the values $v_2, \ldots, v_t$ (i.e. the shares of

$P_2, \ldots, P_t$) and submits invalid share $v'_1 = (v'_{s,1}, v'_{C,1})$ such that $v'_{s,1} \neq v_{s,1}$. P_1 is not identified as a cheater only if he submits $v'_{C,1}$ such that $v'_{C,1} = C(\psi(v'_{s,1}, 1))$ since Reconst can recover the original $C(x)$ even when t shares are forged. Further, since $\{C(x) \mid C(x) \in GF(q)[X], \deg(C(x)) \leq t\}$ is a strong class of universal hash functions and $\psi(v'_{s,1}, 1)$ is different from any of $\psi(v_{s,i}, i)$ $(1 \leq i \leq t,)$ the following equation holds:

$$\Pr[C(\psi(v'_{s,1}, 1)) = v'_{C,1} \mid C(\psi(v_{s,i}, i)) = v_{C,i} \ (\text{for } 1 \leq i \leq t)] = 1/q$$

where the probability is taken over the random choice of $C(x)$. Since the above discussion holds for any critical cheater P_i $(1 \leq i \leq t,)$ we see that no critical cheater can succeed in cheating without being identified with probability better than $1/q$. □

It should be noted that the size of share of the proposed scheme is independent of any of n, k and t, though, there is an implicit limitation on the parameter that $\epsilon < 1/(n \cdot |\mathcal{S}|)$ must holds. This is similar limitation of Shamir's secret sharing scheme which implicitly requires $|\mathcal{S}| > n$.

3.2 A Scheme with Flexible Parameter Choice

As we noted in the previous section, there is such a limitation in the first scheme that the successful cheating probability of cheaters must be smaller than $\frac{1}{n \cdot |\mathcal{S}|}$. This limitation is not desirable, especially when we want to share a secret with large size. Consider the situation in which we want to share a 1M bit secret (i.e. $|\mathcal{S}| = 2^{2^{20}}$,) with the first scheme. In this case, the share size becomes as large as 2M bit with a security level of $\epsilon < 1/2^{2^{20}}$ whereas $\epsilon = 1/2^{128}$ will be sufficient in real life. The second scheme is useful in such a situation since the successful cheating probability of cheaters can be chosen without regard to the size of the secret and the share size can be made reasonable in the second scheme. For example, when we share a 1M bit secret with the second scheme with $\epsilon = 1/2^{128}$, the share size is only (1M+282) bit.

The basic idea of the second scheme is same as the first scheme. We introduce the following trick to the first scheme so that we can determine $|\mathcal{S}|$ and ϵ flexibly. In the first scheme, the random polynomial $C(x)$ must be chosen from $GF(q)[X]$ such that $q \geq n \cdot |\mathcal{S}|$ in order to ensure $\psi(v_i, i) \neq \psi(v_j, j)$ for any distinct (i, v_i) and (j, v_j), which causes $\epsilon \leq \frac{1}{n \cdot |\mathcal{S}|}$. In the second scheme, we introduce *almost universal hash function* (e.g. [24]) $\phi_e : \mathcal{S} \to GF(p)$ (where $\mathcal{S} = GF(p^N)$) and modify the input of $C(x)$ ($C_s(x)$ in the second scheme) to $\psi(\phi_e(v_i), i)$ where $\psi : GF(p) \times \{1 \ldots, n\}$ is an injective function. The use of ϕ_e allows $\psi(\phi_e(v_i), i) = \psi(\phi_e(v'_i), i)$ with small probability, though, the limitation of $\epsilon < \frac{1}{n \cdot |\mathcal{S}|}$ can be eliminated since the range of ϕ_e is chosen flexibly by choosing the parameter p, N and introduce a universal hash family $\phi_e(x_0, \ldots, x_{N-1}) = \sum_{i=0}^{N-1} x_i \cdot e^i$ defined over $GF(p)$. The share generation algorithm and the secret reconstruction algorithm of the second scheme are described as follows:

Share Generation: On input a secret $(s_0, \ldots, s_{N-1}) \in GF(p^N)$, the share generation algorithm ShareGen outputs a list of shares $(v_1, \ldots, v_n)$ as follows:

1. Generate a random polynomial $f_s(x) \in GF(p^N)[X]$ of degree $k-1$ such that $f_s(0) = s$.
2. Generate $e \in GF(p)$ randomly and construct a random polynomial $C_e(x) \in GF(p)[X]$ of degree t such that $C_e(0) = e$.
3. Generate random polynomials $C_s(x) \in GF(q)[X]$ of degree t such that $q \geq n \cdot p$.
4. Compute $v_{s,i} = (v_{s,i,0}, \ldots, v_{s,i,N-1}) = f_s(i)$ where $v_{s,i,j} \in GF(p)$ (for $0 \leq j \leq N-1$), $v_{C_e,i} = C_e(i)$ and $v_{C_s,i} = C_s(\psi(\sum_{j=0}^{N-1} v_{s,i,j} \cdot e^j, i))$.
5. Compute $v_i = (v_{s,i}, v_{C_e,i}, v_{C_s,i})$ and output $(v_1, \ldots, v_n)$.

Secret Reconstruction and Cheater Identification: On input a list of m $(m \geq k)$ shares $((v_{s,i_1}, v_{C_e,i_1}, v_{C_s,i_1}), \ldots, (v_{s,i_m}, v_{C_e,i_m}, v_{C_s,i_m}))$ the secret reconstruction algorithm Reconst outputs a secret or a list of identities of cheaters as follows.

1. Reconstruct $\hat{C}_s(x)$ and $\hat{C}_e(x)$ from $(v_{C_s,i_1}, \ldots, v_{C_s,i_m})$ and $(v_{C_e,i_1}, \ldots, v_{C_e,i_m})$, respectively using an error correction algorithm of Reed-Solomon code.
2. Check if $v_{C_e,i_j} = \hat{C}_e(i_j)$ (for $1 \leq j \leq m$.) If $v_{C_e,i_j} \neq \hat{C}_e(i_j)$ then i_j is added to the list of cheaters L.
3. Compute $\hat{e} = \hat{C}_e(0)$.
4. Check if $v_{C_s,i_j} = \hat{C}_s(\psi(\sum_{\ell=0}^{N-1} v_{s,i_j,\ell} \cdot \hat{e}^\ell, i_j))$ holds (for $1 \leq j \leq m$.) i_j is added to the list of cheaters L if this is not the case.
5. If $|L| \leq m-k$ then reconstruct $f_s(x)$ from (k or more) shares v_{i_j} such that $i_j \notin L$ using Lagrange interpolation, and output $(f_s(0), L)$ if $\deg(f_s) \leq k-1$, otherwise Reconst output $(\perp, L)$. Reconst also output $(\perp, L)$ if $|L| > m-k$ holds.

Security of the proposed scheme can be summarized by the following theorem. Note that the successful cheating probability ϵ can be chosen without regard to $|\mathcal{S}|$ by selecting the value of p appropriately.

Theorem 2. *If $t \leq \lfloor (k-1)/3 \rfloor$ then the proposed scheme is a (t, ϵ) cheater identifiable secret sharing scheme with public cheater identification such that*

$$|\mathcal{S}| = p^N, \quad \epsilon = (N-1)/p + 1/q \leq N/p, \quad q \geq n \cdot p, \quad |\mathcal{V}_i| = p^{N+1} \cdot q\,.$$

Proof. As in the proof of Theorem 1, we can assume $P_1, \ldots, P_t$ are cheaters who cooperatively try to fool the other participants by forging (part of) their shares. Suppose that P_1 is a critical cheater who submits invalid share $v'_1 = (v'_{s,1}, v'_{C_e,1}, v'_{C_s,1})$ such that $v'_{s,1} \neq v_{s,1}$. Since $(v_{C_e,i_1}, \ldots, v_{C_e,i_m})$ is a codeword of Reed-Solomon Code capable of correcting up to t errors, t cheaters cannot alter the value of e. Therefore, P_1 is not identified as a cheater only if he submits $(v'_{s,1}, v'_{C_e}, v'_{C_s,1})$ such that $v'_{C_s,1} = C_s(\psi(\sum_{\ell=0}^{N-1} v'_{s,i,\ell} \cdot e^\ell, 1))$ where e is uniformly and randomly distributed over $GF(p)$. There are two cases to consider in computing such probability. In the first case suppose that P_1 forged its share in a way that $v'_{C_s,1} \neq v_{C_s,1}$. In this case, successful cheating probability ϵ_1 of P_1 who

knows that $v_{C_s,i} = C_s(\psi(\sum_{\ell=0}^{N-1} v_{s,i,\ell} \cdot e^\ell, i))$ hold for $1 \le i \le t$ is computed as follows (for simplicity we will denote $\sum_{\ell=0}^{N-1} v_{s,i,\ell} \cdot e^\ell$ by $\phi_e(v_{s,i})$.)

$$\begin{aligned}
\epsilon_1 &= \Pr[v'_{C_s,1} = C_s(\psi(\phi_e(v'_{s,1}), 1)) \mid v_{C_s,i} = C_s(\psi(\phi_e(v_{s,i}), i)) \ \text{(for } 1 \le i \le t)] \\
&= \Pr[\phi_e(v_{s,i}) \neq \phi_e(v'_{s,i})] \\
&\quad \cdot \Pr\left[v'_{C_s,1} = C_s(\psi(\phi_e(v'_{s,1}), 1)) \,\middle|\, \begin{array}{c} v_{C_s,i} = C_s(\psi(\phi_e(v_{s,i}), i)) \ \text{(for } 1 \le i \le t), \\ \phi_e(v_{s,i}) \neq \phi_e(v'_{s,i}) \end{array} \right] \\
&\le 1/q
\end{aligned}$$

where the last inequality directly follows from the fact that $\{C_s\}$ is a family of a strong class of strongly universal hash function with strength $t+1$ (see the proof of Theorem 1 for details.)

Next we consider the second case in which P_1 forged its share in a way that $v'_{C_s,1} = v_{C_s,1}$ holds. In this case ϵ_1 is computed as follows.

$$\begin{aligned}
\epsilon_1 &= \Pr[v'_{C_s,1} = C_s(\psi(\phi_e(v'_{s,1}), 1)) \mid v_{C_s,i} = C_s(\psi(\phi_e(v_{s,i}), i)) \ \text{(for } 1 \le i \le t)] \\
&= \Pr[\phi_e(v_{s,i}) = \phi_e(v'_{s,i})] + \Pr[\phi_e(v_{s,i}) \neq \phi_e(v'_{s,i})] \\
&\quad \cdot \Pr\left[v'_{C_s,1} = C_s(\psi(\phi_e(v'_{s,1}), 1)) \,\middle|\, \begin{array}{c} v_{C_s,i} = C_s(\psi(\phi_e(v_{s,i}), i)) \ \text{(for } 1 \le i \le t), \\ \phi_e(v_{s,i}) \neq \phi_e(v'_{s,i}) \end{array} \right] \\
&\le \Pr[\phi_e(v_{s,i}) = \phi_e(v'_{s,i})] + 1/q \ \le \ (N-1)/p + 1/q
\end{aligned}$$

where the last two inequalities follows from the property of a strong class of universal hash functions and the well-known fact that a polynomial of degree $N-1$ (e.g. ϕ_e) has at most $N-1$ roots. It is easy to see that the successful cheating probability of any critical cheater is upper bounded by N/p since $(N-1)/p + 1/q \le N/p$ holds. □

Note that the bit length of shares $\log|\mathcal{V}_i|$ is approximately $\log|\mathcal{S}| + 2\log(1/\epsilon) + 2\log\log|\mathcal{S}|$ in the above scheme. Therefore, we can determine size of the secret and successful cheating probability flexibly only by paying $\log(1/\epsilon) + 2\log\log|\mathcal{S}|$ additional bits compared to the bound.

4 A Publicly Cheater Identifiable Scheme for $t \le \lfloor \frac{k-2}{2} \rfloor$

In this section we show that we can construct a very efficient publicly cheater identifiable scheme even when the number of cheaters t does not satisfy $t \le \lfloor (k-1)/3 \rfloor$. More precisely, we present a publicly cheater identifiable scheme whose secret reconstruction algorithm can catch up to $\lfloor (k-2)/2 \rfloor$ cheaters. We note that the cheater identifiability of the scheme is nearly optimum since $t = \lfloor (k-1)/2 \rfloor$ is the theoretical upper bound for public cheater identification. Furthermore, the size of share $|\mathcal{V}_i|$ of the proposed scheme is much smaller than that of [12] despite the difference of their cheater identifiabilities.

The share generation algorithm of the proposed scheme is exactly the same as the one presented in §3.1. To identify more than $(k-1)/3$ cheaters, the

secret reconstruction algorithm examines the *consistency* of all the possible $\binom{k}{t+2}$ subsets of k shares input to the algorithm. Here, the consistency of $t+2$ shares (v_{s,i_j}, v_{C,i_j}) $(1 \le j \le t+2)$ is examined by verifying whether $t+2$ points $(\psi(v_{s,i_j}, i_j), v_{C,i_j})$ $(1 \le j \le t+2)$ lie on a polynomial of degree t. The intuition behind the idea is as follows. Suppose t cheaters try to fool the reconstruction algorithm by forging their shares. Since we assume $t \le \lfloor (k-2)/2 \rfloor$, there are at least $t+2$ unforged shares input to the reconstruction algorithm. Therefore, we can guarantee that (1) there exists at least one subset of consistent shares of size $t+2$ (i.e. shares which does not contain a forged share,) and (2) any subsets of size $t+2$ contain at least two unforged shares. We will make use of these facts to catch cheaters since $t+2$ shares containing both forged and unforged shares can be consistent only with very low probability. The detailed description of the proposed reconstruction algorithm is described as follows.

Secret Reconstruction and Cheater Identification for $t \le \lfloor (k-2)/2 \rfloor$: On input a list of $m\,(\ge k)$ shares $((v_{s,i_1}, v_{C,i_1}), \ldots, (v_{s,i_m}, v_{C,i_m}))$, the secret reconstruction algorithm Reconst output a secret or a list of identities of cheaters as follows.

1. If $t \le (m-1)/3$ holds, outputs $(s, L) \leftarrow \mathsf{Reconst}^{(3t+1)}((v_{s,i_1}, v_{C,i_1}), \ldots, (v_{s,i_m}, v_{C,i_m}))$, where $\mathsf{Reconst}^{(3t+1)}$ denotes the secret reconstruction algorithm for $t \le \lfloor (k-1)/3 \rfloor$ (i.e. Reconst presented in §3.1.)
2. Otherwise, let $L \leftarrow \{i_1, \ldots, i_m\}$ and repeat the following steps 2a–2b for all subsets $\mathcal{I} \subseteq \{i_1, \ldots, i_m\}$ such that $|\mathcal{I}| = t+2$.
   (a) Compute $c_{\mathcal{I}}$ by $c_{\mathcal{I}} = \sum_{i \in \mathcal{I}} v_{C,i} \cdot \prod_{\substack{j \in \mathcal{I} \\ j \ne i}} \frac{1}{\psi(v_{s,i}, i) - \psi(v_{s,j}, j)}$, where $c_{\mathcal{I}}$ is the coefficient of x^{t+1} of the polynomial $C(x)$ constructed from the $t+2$ points $(\psi(v_{s,i}, i), v_{C,i})$ $(i \in \mathcal{I}.)$
   (b) If $c_{\mathcal{I}} = 0$ holds, then $L \leftarrow L \setminus \mathcal{I}$ (i.e. remove $\mathcal{I}$ from the list of cheaters.) Note that $c_{\mathcal{I}} = 0$ holds if all of $t+2$ shares are unforged since we choose random polynomial $C(x)$ of degree t in the share generation algorithm.
3. If $|L| \le m-k$ holds then reconstruct $f_s(x)$ from (k or more) shares v_{s,i_j} such that $i_j \notin L$ using Lagrange interpolation, and output $(f_s(0), L)$ if $\deg(f_s) \le k-1$, otherwise Reconst output $(\perp, L)$. Reconst also output $(\perp, L)$ if $|L| > m-k$ holds.

Security of the proposed scheme can be summarized by the following theorem.

Theorem 3. *If $t \le \lfloor (k-2)/2 \rfloor$ then the proposed scheme is a (t, ϵ) cheater identifiable secret sharing scheme with public cheater identification such that*

$$|\mathcal{S}| = p, \quad \epsilon = \frac{(t+1) \cdot 2^{3t-1}}{p}, \quad q \ge n \cdot p, \quad |\mathcal{V}_i| = p \cdot q \ \left(\approx \frac{n \cdot (t+1) \cdot 2^{3t-1} \cdot |\mathcal{S}|}{\epsilon}\right).$$

Proof. As in the proof of Theorem 1, we can assume $P_1, \ldots, P_t$ are cheaters who cooperatively try to fool the other participants $P_{t+1}, \ldots, P_m$ by forging (part of) their shares. Suppose that P_1 is a critical cheater who submits invalid share $v'_1 = (v'_{s,1}, v'_{C,1})$ such that $v'_{s,1} \ne v_{s,1}$. We will show that the probability that the successful cheating probability of P_1 is upper bounded by ϵ $(= \frac{(t+1) \cdot 2^{3t-1}}{p})$.

From the proof of Theorem 1, it is easy to see that, if $t \leq (m-1)/3$ holds, the successful cheating probability of P_1 is upper bounded by $q(< \epsilon)$ since we can apply error correction algorithm of the generalized Reed-Solomon codes to $(v'_{C,1}, \ldots, v'_{C,t}, v_{C,t+1}, \ldots, v_{C,m})$.

Now we will show the proposed reconstruction algorithm can catch cheaters with probability better than $1-\epsilon$ even against $t > (k-1)/3$ cheaters. It suffices to show that the probability that there exists at least one subset $\mathcal{I} \subseteq \{1, \ldots, m\}$ such that (1) $1 \in \mathcal{I}$, (2) $|\mathcal{I}| = t+2$, and (3) $c_{\mathcal{I}} = 0$, is lower bounded by ϵ.

Toward showing the above, we will first show that the probability $\epsilon(\mathcal{I})$ that $c_{\mathcal{I}} = 0$ holds for given $\mathcal{I}$ is lower bounded by $(t+1)/p$ for any $\mathcal{I}$ such that $1 \in \mathcal{I}$ and $|\mathcal{I}| = t+2$. Without loss of generality, we can assume $\mathcal{I} = \{\ell_1, \ell_2, \ldots, \ell_{t+2}\}$ and $P_{\ell_1} \ldots, P_{\ell_{t'}}$ $(t' \leq t)$ are cheaters.

To evaluate $\epsilon(\mathcal{I})$, we will analyze the structure of $c_{\mathcal{I}}$. Here, we will use the notation ψ_i to denote $\psi(v_{s,i}, i)$ and the notation X' to indicate the variable X is owned and controlled by the cheaters.

$$c_{\mathcal{I}} = \sum_{\ell_i \in \mathcal{I}} v_{C,\ell_i} \cdot \prod_{\substack{\ell_j \in \mathcal{I} \\ j \neq i}} \frac{1}{\psi_{\ell_i} - \psi_{\ell_j}} = \sum_{i=1}^{t'} v'_{C,\ell_i} \cdot \prod_{\substack{j=1 \\ j \neq i}}^{t'} \frac{1}{\psi'_{\ell_i} - \psi'_{\ell_j}} \prod_{j=t'+1}^{t+2} \frac{1}{\psi'_{\ell_i} - \psi_{\ell_j}}$$
$$+ \sum_{j=t'+1}^{t+2} v_{C,\ell_j} \cdot \prod_{i=1}^{t'} \frac{1}{\psi_{\ell_j} - \psi'_{\ell_i}} \prod_{\substack{i=t'+1 \\ i \neq j}}^{t+2} \frac{1}{\psi_{\ell_j} - \psi_{\ell_i}}$$

We will rewrite $v'_{C,\ell_i} \cdot \prod_{j=1, j\neq i}^{t'} \frac{1}{\psi'_{\ell_i} - \psi'_{\ell_j}}$ by A_i where each A_i is determined by the shares submitted by the cheaters and is known to the cheaters. Furthermore, to make the proof clearer, we replace $v_{C,i}$ by $C(\psi_i)$ where $C(x)$ is a polynomial chosen by the dealer in the share generation phase.

$$c_{\mathcal{I}} = \sum_{i=1}^{t'} A_i \cdot \prod_{j=t'+1}^{t+2} \frac{1}{\psi'_{\ell_i} - \psi_{\ell_j}} + \sum_{j=t'+1}^{t+2} C(\psi_{\ell_j}) \cdot \prod_{i=1}^{t'} \frac{1}{\psi_{\ell_j} - \psi'_{\ell_i}} \prod_{\substack{i=t'+1 \\ i \neq j}}^{t+2} \frac{1}{\psi_{\ell_j} - \psi_{\ell_i}} \quad (3)$$

With the knowledge about shares owned by the cheaters, the number of possible candidates for $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+2}}, C)$ becomes $p^{t-t'+2} \times q^{t-t'+1}$ since (1) ψ_{ℓ_i} $(t'+1 \leq i \leq t+2)$ look randomly, uniformly and independently distributed over the set $\Psi_{\ell_i} = \{\psi(v_{s,\ell_i}, \ell_i) \mid v_{s,\ell_i} \in GF(p)\}$ even with the knowledge of cheaters, and (2) $(\psi_{\ell_{t'+1}}, v_{C,\ell_{t'+1}}), \ldots, (\psi_{\ell_{t+1}}, v_{C,\ell_{t+1}})$ uniquely determines the polynomial $C(x)$.

Now, we will estimate the upper bound of the number of $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+2}}, C)$ with which $c_{\mathcal{I}} = 0$. For any fixed $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}, C) = (\hat{\psi}_{\ell_{t'+1}}, \ldots, \hat{\psi}_{\ell_{t+1}}, \hat{C})$, eq (3) is rewritten as follows:

$$c_{\mathcal{I}} = \sum_{i=1}^{t'} \frac{\hat{A}_i}{\psi'_{\ell_i} - \psi_{\ell_{t+2}}} + \sum_{j=t'+1}^{t+1} \frac{\hat{B}_j}{\hat{\psi}_{\ell_j} - \psi_{\ell_{t+2}}}$$
$$+ \hat{C}(\psi_{t+2}) \cdot \prod_{i=1}^{t'} \frac{1}{\psi_{\ell_{t+2}} - \psi'_{\ell_i}} \prod_{i=t'+1}^{t+1} \frac{1}{\psi_{\ell_{t+2}} - \hat{\psi}_{\ell_i}}$$

where $\hat{A}_i = A_i \cdot \prod_{j=t'+1}^{t+1} \frac{1}{\psi'_{\ell_i} - \hat{\psi}_{\ell_j}}$, $\hat{B}_j = \hat{C}(\hat{\psi}_{\ell_j}) \cdot \prod_{i=1}^{t'} \frac{1}{\hat{\psi}_{\ell_j} - \psi'_{\ell_i}} \cdot \prod_{i=t'+1}^{t+1} \frac{1}{\hat{\psi}_{\ell_j} - \hat{\psi}_{\ell_i}}$ are constant once ψ'_{ℓ_i} $(1 \le i \le t')$, $\hat{\psi}_{\ell_j}$ $(t'+1 \le j \le t+1)$, and $\hat{C}$ are fixed. It is easy to see that there are at most $t+1$ values of $\psi_{\ell_{t+2}}$ with which $c_{\mathcal{I}} = 0$. Therefore, the upper bound of the number of zeros of eq. (3) can be evaluated as follows:

$$
\begin{aligned}
&|\{(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+2}}, C) \mid c_{\mathcal{I}} = 0 \text{ holds}\}| \\
&\le |\{(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}) \mid \psi_{\ell_i} \in \Psi_{\ell_i} \ (t'+1 \le i \le t+1)\}| \\
&\quad \times |\{C(x) \mid C(\psi_{\ell_i}) = v_{C,\ell_i} \ (1 \le i \le t')\}| \times (t+1) \ = \ p^{t-t'+1} \cdot q^{t-t'+1} \cdot (t+1)
\end{aligned}
$$

Therefore, the lower bound of $\epsilon(\mathcal{I})$ is given as follows: $\epsilon(\mathcal{I}) \le \frac{p^{t-t'+1} \cdot q^{t-t'+1} \cdot (t+1)}{p^{t-t'+2} \cdot q^{t-t'+1}} = (t+1)/p$.

From the above inequality and the fact that the number of subsets $\mathcal{I}$ of $\{1, \ldots, 3t\}$ such that $1 \in \mathcal{I}$ and $|\mathcal{I}| = t+2$ is equal to $\binom{3t-1}{t+1}$, the successful cheating probability ϵ is given as follows:

$$
\begin{aligned}
&\epsilon = \Pr[\text{there exists } \mathcal{I} \text{ such that } c_{\mathcal{I}} = 0,\ 1 \in \mathcal{I},\ |\mathcal{I}| = t+2] \\
&\le \sum_{\left\{\mathcal{I} \,\middle|\, \substack{1 \in \mathcal{I}, \\ |\mathcal{I}| = t+2}\right\}} \epsilon(\mathcal{I}) = \left| \left\{ \mathcal{I} \,\middle|\, \substack{1 \in \mathcal{I}, \\ |\mathcal{I}| = t+2} \right\} \right| \cdot \epsilon(\mathcal{I}) \ \le \ \binom{3t-1}{t+1} \cdot \frac{t+1}{p} \ \le \ \frac{(t+1) \cdot 2^{3t-1}}{p}
\end{aligned}
$$

The size of share satisfies $|\mathcal{V}_i| = p \cdot q$ and is approximately written by $|\mathcal{V}_i| \approx \frac{n \cdot (t+1) \cdot 2^{3t-1} |\mathcal{S}|}{\epsilon}$ since $q \approx n \cdot p$ and $p \le \frac{(t+1) \cdot 2^{3t-1}}{\epsilon}$. □

Though size of share grows exponentially with the number of cheaters, the size of share is much smaller compared to Kurosawa *et al.* [12] whose size of share is as large as $|\mathcal{S}|/\epsilon^{t+2}$ (note that $\frac{1}{\epsilon} \gg 2$.) Even compared to the theoretical lower bound of eq. (1), the bit length of the proposed scheme is only $3t + \log t + \log n$ bit longer. On the other hand, the drawback of the proposed reconstruction algorithm is its computational inefficiency. In fact the reconstruction algorithm requires to compute Lagrange interpolation $\binom{3t}{t+2}$ times to identify cheaters. However, in the usual setting, the cheater identification of the proposed scheme is still feasible. Consider, for example, the situation where we want to catch up to 10 cheaters (i.e. $t = 10$). The number of Lagrange interpolation we have to invoke is $\binom{30}{12} = 4,118,725$, which is indeed feasible even by the current personal computer.

We should note that the similar (brute force search) technique can be applied to the scheme given in the section §3.2.

5 A Publicly Cheater Identifiable Scheme for $t \le \lfloor \frac{k-1}{2} \rfloor$

The scheme presented in Section 4 meets the theoretical upper bound $t = \lfloor (k-1)/2 \rfloor$ on number of cheaters that a scheme can identify *when the threshold k is even*. This is because $\lfloor (k-2)/2 \rfloor = \lfloor (k-1)/2 \rfloor$ holds for even k. When k is

odd, on the other hands, the scheme will fail to catch $\lfloor (k-1)/2 \rfloor (> \lfloor (k-2)/2 \rfloor)$ cheaters. In this section we present a publicly cheater identifiable scheme which can catch $\lfloor (k-1)/2 \rfloor$ cheaters. The size of shares $|\mathcal{V}_i|$ of the proposed scheme is not so small as the scheme for $t \leq \lfloor (k-2)/2 \rfloor$, though, the size of shares of the scheme is still much smaller than that of [12].

Here, we will review the scheme presented in the previous section to explain the idea behind the proposed scheme for $t \leq \lfloor (k-1)/2 \rfloor$. The reconstruction algorithm of the scheme for $t \leq \lfloor (k-2)/2 \rfloor$ identifies cheaters by checking the degree of the polynomial reconstructed from $t+2$ points. Using this technique, it can be ensured that (1) $t+2$ points containing forged share cannot construct a polynomial with degree less than or equal to t and, (2) $t+2$ points containing no forged share construct a polynomial with degree less than or equal to t. Unfortunately, we cannot apply this technique when $t = \lfloor (k-1)/2 \rfloor$ since any $t+2$ shares contain at least one forged share and we cannot find set of honest shares (and, therefore, cannot identify cheaters correctly.)

To make it possible to find honest shares by examining consistency of $t+1$ shares, a share v_i of the proposed scheme consists of $v_i = (v_{s,i}, v_{C_0,i}, v_{C_1,i})$ where $v_{s,i}$ is a share of Shamir's k-out-of-n scheme and $v_{C_0,i}, v_{C_1,i}$ are the points on the polynomials $C_0(x) = \sum_{i=0}^{t} a_{0,i} x^i$ and $C_1(x) = \sum_{i=0}^{t} a_{1,i} x^i$ such that $a_{0,0} = a_{1,t}$. Then we can verify the consistency of $t+1$ shares by examining the equality $\hat{a}_{0,t} = \hat{a}_{1,t}$ where $\hat{a}_{0,0}$ and $\hat{a}_{1,t}$ are coefficients of x^0 and x^t of polynomials $\hat{C}_0$ and $\hat{C}_1$, respectively, where $\hat{C}_0$ and $\hat{C}_1$ are polynomials reconstructed from $t+1$ shares. Since an additional element (i.e. $v_{C_1,i}$) is required in the proposed scheme, the size of share is larger than that of the scheme for $t \leq \lfloor (k-2)/2 \rfloor$.

The share generation algorithm ShareGen and the secret reconstruction algorithm Reconst of the proposed scheme are described as follows where p and q are prime powers such that $q \geq n \cdot p$ and $\psi : GF(p) \times \{1, \ldots, n\} \to GF(q)$ is an injective function.

Share Generation: On input a secret $s \in GF(p)$, the share generation algorithm ShareGen outputs a list of shares $(v_1, \ldots, v_n)$ as follows:

1. Generate a random polynomial $f_s(x) \in GF(p)[X]$ of degree $k-1$ such that $f_s(0) = s$.
2. Generate random polynomials $C_0(x) = \sum_{i=0}^{t} a_{0,i} x^i, C_1(x) = \sum_{i=0}^{t} a_{1,i} x^i \in GF(q)[X]$ such that the $a_{0,0} = a_{1,t}$.
3. Compute $v_i = (f_s(i), C_0(\psi(f_s(i), i)), C_1(\psi(f_s(i), i)))$ and output $(v_1, \ldots, v_n)$.

Secret Reconstruction and Cheater Identification: On input a list of m $(\geq k)$ shares $((v_{s,i_1}, v_{C_0,i_1}, v_{C_1,i_1}), \ldots, (v_{s,i_m}, v_{C_0,i_m}, v_{C_1,i_m}))$, the secret reconstruction algorithm Reconst output a secret or a list of identities of cheaters as follows.

1. If $t \leq \lfloor (m-1)/3 \rfloor$ holds, outputs $(s, L) \leftarrow \mathsf{Reconst}^{(3t+1)}((v_{s,i_1}, v_{C_0,i_1}), \ldots, (v_{s,i_m}, v_{C_0,i_m}))$, where $\mathsf{Reconst}^{(3t+1)}$ denotes the secret reconstruction algorithm for $t \leq \lfloor (k-1)/3 \rfloor$ (i.e. Reconst presented in §3.1.)
2. Otherwise, let $L \leftarrow \{i_1, \ldots, i_m\}$ and repeat the following steps 2a–2b for all subsets $\mathcal{I} \subseteq \{i_1, \ldots, i_m\}$ such that $|\mathcal{I}| = t+1$.

(a) Compute $a_{\mathcal{I},0}$ and $a_{\mathcal{I},1}$ as follows:

$$a_{\mathcal{I},0} = \sum_{\ell\in\mathcal{I}} v_{C_0,\ell} \cdot \prod_{\substack{j\in\mathcal{I}\\ j\neq\ell}} \frac{-\psi(v_{s,j},j)}{\psi(v_{s,\ell},\ell)-\psi(v_{s,j},j)}$$
$$a_{\mathcal{I},1} = \sum_{\ell\in\mathcal{I}} v_{C_1,\ell} \cdot \prod_{\substack{j\in\mathcal{I}\\ j\neq\ell}} \frac{1}{\psi(v_{s,\ell},\ell)-\psi(v_{s,j},j)}$$

where $a_{\mathcal{I},0}$ and $a_{\mathcal{I},1}$ are coefficients of x^0 and x^t of the polynomials $C_0(x)$ and $C_1(x)$ constructed from the $t+1$ points $(\psi(v_{s,i},i), v_{C_0,i})$ $(i\in\mathcal{I})$ and $(\psi(v_{s,i},i), v_{C_1,i})$ $(i\in\mathcal{I})$, respectively.

(b) If $a_{\mathcal{I},0} = a_{\mathcal{I},1}$ holds then $L \leftarrow L\setminus\mathcal{I}$ (i.e. remove $\mathcal{I}$ from the list of cheaters.)

3. If $|L| \le m-k$ holds then reconstruct $f_s(x)$ from (k or more) shares v_{s,i_j} such that $i_j \notin L$ using Lagrange interpolation, and output $(f_s(0), L)$ if $\deg(f_s) \le k-1$, otherwise Reconst output $(\perp, L)$. Reconst also output $(\perp, L)$ if $|L| > m-k$ holds.

Security of the proposed scheme can be summarized by the following theorem.

Theorem 4. *If $t \le \lfloor (k-1)/2 \rfloor$ then the proposed scheme is a (t,ϵ) cheater identifiable secret sharing scheme with public cheater identification such that*

$$|\mathcal{S}| = p, \quad \epsilon = \frac{t\cdot 2^{3t}}{p}, \quad q \ge n\cdot p, \quad |\mathcal{V}_i| = p\cdot q^2 \ \left(\approx \frac{(n\cdot t\cdot 2^{3t})^2\cdot|\mathcal{S}|}{\epsilon^2}\right).$$

Proof. The proof is similar to that of Theorem 3 except that we pay attention to the 0-th and the t-th degree coefficients of polynomials $C_0(x)$ and $C_1(x)$, respectively, in analyzing the security of the proposed scheme.

As in the proof of Theorem 3, we can assume $P_1,\ldots,P_t$ are cheaters who cooperatively try to fool the other participants $P_{t+1},\ldots,P_m$ by forging (part of) their shares. Suppose that P_1 is a critical cheater who submits invalid share $v_1' = (v_{s,1}', v_{C_0,1}', v_{C_1,1}')$ such that $v_{s,1}' \neq v_{s,1}$. We will show that the probability that the successful cheating probability of P_1 is upper bounded by $\epsilon\,(= \frac{t\cdot 2^{3t}}{p})$.

From the proof of Theorem 1, it is easy to see that, if $t \le (m-1)/3$ holds, the successful cheating probability of P_1 is upper bounded by $q(<\epsilon)$.

Now we will show the proposed reconstruction algorithm can catch cheaters with probability better than $1-\epsilon$ even against $t = \lfloor (k-1)/2 \rfloor$ cheaters. It suffices to show that the probability that there exists at least one subset $\mathcal{I}\subseteq\{1,\ldots,m\}$ such that (1) $1\in\mathcal{I}$, (2) $|\mathcal{I}| = t+1$, and (3) $a_{\mathcal{I},0} = a_{\mathcal{I},1}$, is lower bounded by ϵ.

Toward showing the above, we will first show that the probability $\epsilon(\mathcal{I})$ that $a_{\mathcal{I},0} = a_{\mathcal{I},1}$ holds for given $\mathcal{I}$ is lower bounded by $2t/p$ for any $\mathcal{I}$ such that $1\in\mathcal{I}$ and $|\mathcal{I}| = t+1$. Without loss of generality, we can assume $\mathcal{I} = \{\ell_1, \ell_2, \ldots, \ell_{t+1}\}$ and $P_{\ell_1}\ldots, P_{\ell_{t'}}$ $(t' \le t)$ are cheaters.

To evaluate $\epsilon(\mathcal{I})$, we will analyze the structures of $a_{\mathcal{I},0}$ and $a_{\mathcal{I},1}$. As in the proof of Theorem 3, we will use the notation ψ_i to denote $\psi(v_{s,i},i)$ and the notation X' to indicate the variable X is owned and controlled by the cheaters. By the similar discussion to the proof of Theorem 3, $a_{\mathcal{I},0}$ and $a_{\mathcal{I},1}$ can be rewritten as follows:

$$a_{\mathcal{I},0} = \sum_{i=1}^{t'} A_{0,i} \cdot \prod_{j=t'+1}^{t+1} \frac{-\psi_{\ell_j}}{\psi'_{\ell_i} - \psi_{\ell_j}} + \sum_{j=t'+1}^{t+1} C_0(\psi_{\ell_j}) \cdot \prod_{i=1}^{t'} \frac{-\psi'_{\ell_i}}{\psi_{\ell_j} - \psi'_{\ell_i}} \prod_{\substack{i=t'+1 \\ i \neq j}}^{t+1} \frac{-\psi_{\ell_i}}{\psi_{\ell_j} - \psi_{\ell_i}} \quad (4)$$

$$a_{\mathcal{I},1} = \sum_{i=1}^{t'} A_{1,i} \cdot \prod_{j=t'+1}^{t+1} \frac{1}{\psi'_{\ell_i} - \psi_{\ell_j}} + \sum_{j=t'+1}^{t+1} C_1(\psi_{\ell_j}) \cdot \prod_{i=1}^{t'} \frac{1}{\psi_{\ell_j} - \psi'_{\ell_i}} \prod_{\substack{i=t'+1 \\ i \neq j}}^{t+1} \frac{1}{\psi_{\ell_j} - \psi_{\ell_i}} \quad (5)$$

With the knowledge about shares owned by the cheaters, the number of possible candidates for $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}, C_0, C_1)$ becomes $p^{t-t'+1} \times q^{2(t-t')+1}$ since (1) ψ_{ℓ_i} $(t'+1 \leq i \leq t+1)$ look randomly, uniformly and independently distributed over the set $\Psi_{\ell_i} = \{\psi(v_{s,\ell_i}, \ell_i) \mid v_{s,\ell_i} \in GF(p)\}$ even with the knowledge of cheaters, and (2) $(\psi_{\ell_{t'+1}}, v_{C_0,\ell_{t'+1}}, v_{C_1,\ell_{t'+1}}) \ldots (\psi_{\ell_t}, v_{C_0,\ell_t}, v_{C_1,\ell_t})$ and $(\psi_{\ell_{t+1}}, v_{C_0,\ell_{t+1}})$ uniquely determines the polynomials $C_0(x)$ and $C_1(x)$ such that $a_{0,t} = a_{1,t}$ holds.

Now, we will estimate the upper bound of the number of $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}, C_0, C_1)$ with which $a_{\mathcal{I},0} = a_{\mathcal{I},1}$. By the similar discussion to the proof of Theorem 3, we can show that, for any fixed $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_t}, C_0, C_1) = (\hat{\psi}_{\ell_{t'+1}}, \ldots, \hat{\psi}_{\ell_t}, \hat{C}_0, \hat{C}_1)$, eq. (4) and eq. (5) are rewritten as follows:

$$\begin{aligned} a_{\mathcal{I},0} = &\sum_{i=1}^{t'} \frac{-\hat{A}_{i,0} \cdot \psi_{\ell_{t+1}}}{\psi'_{\ell_i} - \psi_{\ell_{t+1}}} + \sum_{j=t'+1}^{t} \frac{-\hat{B}_{j,0} \cdot \psi_{\ell_{t+1}}}{\hat{\psi}_{\ell_j} - \psi_{\ell_{t+1}}} \\ &+ \hat{C}_0(\psi_{t+1}) \cdot \prod_{i=1}^{t'} \frac{-\psi_{\ell_{t+1}}}{\psi_{\ell_{t+1}} - \psi'_{\ell_i}} \prod_{i=t'+1}^{t} \frac{-\psi_{\ell_{t+1}}}{\psi_{\ell_{t+1}} - \hat{\psi}_{\ell_i}} \\ a_{\mathcal{I},1} = &\sum_{i=1}^{t'} \frac{\hat{A}_{i,1}}{\psi'_{\ell_i} - \psi_{\ell_{t+1}}} + \sum_{j=t'+1}^{t} \frac{\hat{B}_{j,1}}{\hat{\psi}_{\ell_j} - \psi_{\ell_{t+1}}} \\ &+ \hat{C}_1(\psi_{t+1}) \cdot \prod_{i=1}^{t'} \frac{1}{\psi_{\ell_{t+1}} - \psi'_{\ell_i}} \prod_{i=t'+1}^{t} \frac{1}{\psi_{\ell_{t+1}} - \hat{\psi}_{\ell_i}} \end{aligned}$$

where $\hat{A}_{i,0}, \hat{B}_{j,0}, \hat{A}_{i,1}$ and $\hat{B}_{j,1}$ are constant once ψ'_{ℓ_i} $(1 \leq i \leq t')$, $\hat{\psi}_{\ell_j}$ $(t'+1 \leq j \leq t)$, $\hat{C}_0$, and $\hat{C}_1$ are fixed. We see that there are at most $2t$ values of $\psi_{\ell_{t+1}}$ with which $a_{\mathcal{I},0} = a_{\mathcal{I},1}$ since solving $\psi_{\ell_{t+1}}$ such that $a_{\mathcal{I},0}(\psi_{\ell_{t+1}}) = a_{\mathcal{I},1}(\psi_{\ell_{t+1}})$ is equivalent to solving the equation $A_{\mathcal{I}}(\psi_{\ell_{t+1}}) = 0$ for a polynomial $A_{\mathcal{I}}$ of degree $2t$ where $A_{\mathcal{I}}$ is uniquely determined from $a_{\mathcal{I},0}$ and $a_{\mathcal{I},1}$. Therefore, the upper bound of the number of $(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}, C_0, C_1)$ such that $a_{\mathcal{I},0} = a_{\mathcal{I},1}$ can be evaluated as follows:

$$\begin{aligned} &|\{(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_{t+1}}, C_0, C_1) \mid a_{\mathcal{I},0} = a_{\mathcal{I},1} \text{ holds}\}| \\ &\leq |\{(\psi_{\ell_{t'+1}}, \ldots, \psi_{\ell_t}) \mid \psi_{\ell_i} \in \Psi_{\ell_i} \ (t'+1 \leq i \leq t)\}| \\ &\quad \times \left| \left\{ (C_0(x), C_1(x)) \;\middle|\; \begin{array}{c} C_0(\psi_i) = v_{C_0,i},\ C_1(\psi_i) = v_{C_1,i} \ (1 \leq i \leq t'), \\ a_{0,0} = a_{1,t} \end{array} \right\} \right| \times 2t \\ &= p^{t-t'} \cdot q^{2(t-t')+1} \cdot 2t \end{aligned}$$

Therefore, we see that $\epsilon(\mathcal{I})$ is lower bounded by $\epsilon(\mathcal{I}) \leq 2t/p$ since $\epsilon(\mathcal{I}) \leq \frac{p^{t-t'} \cdot q^{2(t-t')+1} \cdot 2t}{p^{t-t'+1} \cdot q^{2(t-t')+1}} = 2t/p$ holds.

From the above inequality and the fact that the number of subsets $\mathcal{I}$ of $\{1, \ldots, 3t\}$ such that $1 \in \mathcal{I}$ and $|\mathcal{I}| = t+1$ is equal to $\binom{3t-1}{t}$, the successful cheating probability ϵ is given as follows:

$$\epsilon = \Pr[\text{there exists } \mathcal{I} \text{ such that } \Delta c_{\mathcal{I}} = 0,\ 1 \in \mathcal{I},\ |\mathcal{I}| = t+1]$$
$$\leq \sum_{\{\mathcal{I} \mid \substack{1 \in \mathcal{I}, \\ |\mathcal{I}| = t+1}\}} \epsilon(\mathcal{I}) = \left| \left\{ \mathcal{I} \,\middle|\, \begin{matrix} 1 \in \mathcal{I}, \\ |\mathcal{I}| = t+1 \end{matrix} \right\} \right| \cdot \epsilon(\mathcal{I}) \leq \binom{3t-1}{t} \cdot \frac{t}{p} \leq \frac{t \cdot 2^{3t}}{p}$$

The size of share satisfies $|\mathcal{V}_i| = p \cdot q$ and is approximately written by $|\mathcal{V}_i| \approx \frac{(n \cdot t \cdot 2^{3t})^2 |\mathcal{S}|}{\epsilon^2}$ since $q \approx n \cdot p$ and $p \leq \frac{t \cdot 2^{3t}}{\epsilon}$. □

6 Conclusion

In this paper, we present efficient (t, ϵ) cheater identifiable (k, n) threshold secret sharing schemes under the conditions $t \leq \lfloor (k-1)/3 \rfloor$, $t \leq \lfloor (k-2)/2 \rfloor$ and $t \leq \lfloor (k-1)/2 \rfloor$, respectively. The schemes which can catch $\lfloor (k-1)/3 \rfloor$ cheaters are the first schemes whose share size is independent of any of n, k and t. Further, in one of these schemes, the share size is almost optimum in the sense that the bit length of the share is only one bit longer than the bound given in [12]. The schemes which can catch $t \leq \lfloor (k-2)/2 \rfloor$ cheaters and $t \leq \lfloor (k-1)/2 \rfloor$ cheaters are, though the bit length of shares grows linear to the number of cheaters, shown to be much more efficient with respect to the size of share compared to [12] and the other schemes with private cheater identification.

In our future work, we will focus on finding an efficient scheme under the condition $t \leq \lfloor (k-1)/2 \rfloor$ such that the size of share is independent of any of n, k and t, and the computational cost for identifying cheaters is small.

References

1. Araki, T.: Efficient (k, n) Threshold Secret Sharing Scheme Secure against Cheating from $n-1$ Cheaters. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 133–142. Springer, Heidelberg (2007)
2. Araki, T., Obana, S.: Flaws in Some Secret Sharing Schemes against Cheating. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 122–132. Springer, Heidelberg (2007)
3. Blakley, G.R.: Safeguarding cryptographic keys. In: Proc. AFIPS 1979, National Computer Conference, vol. 48, pp. 137–313 (1979)
4. Brickell, E.F., Stinson, D.R.: The Detection of Cheaters in Threshold Schemes. SIAM Journal on Discrete Mathematics 4(4), 502–510 (1991)
5. Carpentieri, M.: A Perfect Threshold Secret Sharing Scheme to Identify Cheaters. Designs, Codes and Cryptography 5(3), 183–187 (1995)

6. 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)
7. Carpentieri, M., De Santis, A., Vaccaro, U.: Size of Shares and Probability of Cheating in Threshold Schemes. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 118–125. Springer, Heidelberg (1994)
8. Cramer, R., Damgård, I., Fehr, S.: On the Cost of Reconstructing a Secret, or VSS with Optimal Reconstruction Phase. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 503–523. Springer, Heidelberg (2001)
9. Cabello, S., Padró, C., Sáez, G.: Secret Sharing Schemes with Detection of Cheaters for a General Access Structure. Designs, Codes and Cryptography 25(2), 175–188 (2002)
10. den Boer, B.: A Simple and Key-Economical Unconditional Authentication Scheme. Journal of Computer Security 2, 65–71 (1993)
11. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly Secure Message Transmission. Journal of the ACM 40(1), 17–47 (1993)
12. Kurosawa, K., Obana, S., Ogata, W.: t-Cheater Identifiable (k, n) Secret Sharing Schemes. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 410–423. Springer, Heidelberg (1995)
13. Kurosawa, K., Suzuki, K.: Almost Secure (1-Round, n-Channel) Message Transmission Scheme. In: Desmedt, Y. (ed.) ICITS 2007. LNCS, vol. 4883, pp. 99–112. Springer, Heidelberg (2009)
14. MacWilliams, F., Sloane, N.: The Theory of Error Correcting Codes. North Holland, Amsterdam (1977)
15. McEliece, R.J., Sarwate, D.V.: On Sharing Secrets and Reed-Solomon Codes. Communications of the ACM 24(9), 583–584 (1981)
16. Obana, S., Araki, T.: Almost Optimum Secret Sharing Schemes Secure Against Cheating for Arbitrary Secret Distribution. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 364–379. Springer, Heidelberg (2006)
17. Ogata, W., Kurosawa, K.: Optimum Secret Sharing Scheme Secure against Cheating. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 200–211. Springer, Heidelberg (1996)
18. Ogata, W., Kurosawa, K.: Provably Secure Metering Scheme. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 388–398. Springer, Heidelberg (2000)
19. Ogata, W., Kurosawa, K., Stinson, D.R.: Optimum Secret Sharing Scheme Secure against Cheating. SIAM Journal on Discrete Mathematics 20(1), 79–95 (2006)
20. Pedersen, T.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
21. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: Proc. STOC 1989, pp. 73–85 (1989)
22. Rabin, T.: Robust Sharing of Secrets When the Dealer is Honest or Cheating. Journal of the ACM 41(6), 1089–1109 (1994)
23. Shamir, A.: How to Share a Secret. Communications of the ACM 22(11), 612–613 (1979)
24. Stinson, D.R.: On the Connections between Universal Hashing, Combinatorial Designs and Error-Correcting Codes. Congressus Numerantium 114, 7–27 (1996)
25. Tompa, M., Woll, H.: How to Share a Secret with Cheaters. Journal of Cryptology 1(3), 133–138 (1989)

On Linear Hulls, Statistical Saturation Attacks, PRESENT and a Cryptanalysis of PUFFIN

Gregor Leander

DTU Mathematics
Technical University of Denmark
G.Leander@mat.dtu.dk

Abstract. We discuss complexities of advanced linear attacks. In particular, we argue why it is often more appropriate to examine the median of the complexity than the average value. Moreover, we apply our methods to the block ciphers PUFFIN and PRESENT. For PUFFIN, a 128 bit key cipher, we present an attack which breaks the cipher for at least a quarter of the keys with a complexity less than 2^{58}. In the case of PRESENT we show that the design is sound. The design criteria are sufficient to ensure the resistance against linear attacks, taking into account the notion of linear hulls. Finally, we show that statistical saturation attacks and multi dimensional linear attacks are almost identical.

1 Introduction

Block ciphers are probably one of the most studied objects in cryptography in general. The security of block cipher seems well understood and quite a number of secure and efficient block ciphers are available today. The AES is of course the most studied and analyzed block cipher at present, but many other interesting proposals have been made. Recently, there has been a trend to design block ciphers that are not suitable for every environment, but rather tailored to special platforms and/or purposes. What all those designs have in common is that nowadays a detailed analysis against known cryptanalytic methods is almost mandatory when presenting a new design.

One of those known attacks is of course Matsui's linear attack [1]. However, despite its discovery more than 15 years ago, linear cryptanalysis seems to be less understood in comparison to, for example, differential attacks. In particular, for advanced linear attacks, such as attacks using so called linear hulls or multidimensional cryptanalysis, we still do not understand completely how to estimate their running time correctly. Concerning linear attacks using linear hulls Murphy [2] points out very fundamental problems when estimating the impact of those attacks.

Besides the well known cryptanalytic methods, some new attacks appeared recently, and among those are the so called statistical saturation attack. In a nutshell, the main idea of statistical saturation attacks is to use a poor diffusion in a block cipher by fixing certain input bits in the plaintext and disregarding some

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 303–322, 2011.

of the output bits. While this method currently provides the best attacks against the lightweight block cipher PRESENT, a method to estimate its complexity correctly is missing.

This lack of a deeper understanding is especially surprising as the study of block ciphers is one of the classical fields of cryptography. One would expect that the necessary tools to precisely –and formally correctly – formulate attack complexities have been already developed. However, this is apparently not always the case.

1.1 Our Contributions

In Section 3 of this paper we discuss in detail the problems pointed out by Murphy [2] on linear hulls. However, we do not quite agree with the conclusion that linear hulls do not exist. On the contrary, we explain why linear hulls (when defined correctly) always exist and always have to be taken into account when statements about the attack complexity are made. In order to be able to make meaningful statements about the attack complexity we explain why a paradigm shift from discussing average complexities to discussing medians of complexities is necessary. As we will explain, this holds not only for attacks that make use of linear hulls, but actually also for linear attacks based on a single linear trail. We present methods on how one can, in many cases, compute good approximations of the median of the complexities.

As an example, we apply our methods to cryptanalyze the block cipher PUFFIN (see Section 4). We present an attack on full round PUFFIN, a block cipher with a 128 bit key, that allows us to recover 4 bits of the last round key for at least a quarter of the keys with a complexity below 2^{58}.

In Section 5 we use our methods to understand the resistance of PRESENT to linear cryptanalysis. Most interestingly we show that the design principle of PRESENT is sound, in the sense that any sbox and any bit permutation fulfilling the design criteria of PRESENT yield to a cipher secure against this type of linear attacks. In order to do so, we present a link between optimal bit permutations and central digraphs which is interesting in itself. Central digraphs are classical combinatorial objects (see for example [3]) and the link allows us to answer natural questions about optimal permutations. Most importantly, this link allows us to classify all optimal bit permutations. This classification then allows us to get a deeper understanding of the block cipher PRESENT.

Finally, in Section 6 we solve the problem of estimating the biases (or capacities) in statistical saturation attacks. Using a theorem on the Fourier transformations of restrictions of Boolean functions, we demonstrate that statistical saturation attacks are in principle identical to multi dimensional linear attacks. In particular, this link allows us to evaluate the bias used in statistical saturation attacks using well studied tools and well established theory, a major drawback of statistical saturation attacks so far. Furthermore, we believe that this link makes it possible to apply statistical saturation attacks to other ciphers.

2 Preliminaries

In this section, we fix our notation and recall known identities between the bias of a function, the correlation and the Fourier transformation. After doing so, we recall the basic concept of linear hulls and statistical saturation attacks.

2.1 Bias, Correlation and Fourier Transformation

We denote by $\mathbb{F}_2$ the binary field with two elements and by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$. The canonical inner product on $\mathbb{F}_2^n$ is denoted by $\langle \cdot, \cdot \rangle$, i.e.

$$\langle (a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1}) \rangle := \sum_{i=0}^{n-1} a_i b_i.$$

We note that all linear mappings, i.e. all linear functions, $l : \mathbb{F}_2^n \to \mathbb{F}_2$ can be described as $\ell(x) = \langle a, x \rangle$ for a suitable $a \in \mathbb{F}_2^n$. Given a vector $a \in \mathbb{F}_2^n$ we denote by $\mathrm{wt}(a)$ its *Hamming weight*, i.e. $\mathrm{wt}(a) = |\{0 \leq i < n \mid a_i = 1\}|$. Given a (vectorial Boolean) function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$ the *Fourier coefficient* of F at the pair $(a, b) \in \mathbb{F}_2^n \times \mathbb{F}_2^m$ is defined by

$$\widehat{F}(a, b) = \sum_x (-1)^{\langle b, F(x) \rangle + \langle a, x \rangle}.$$

Given the probability p of the linear approximation $\langle a, x \rangle$ of $\langle b, F(x) \rangle$, i.e.

$$p = \frac{\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)}{2^n}$$

the *bias* $\epsilon_F(a, b)$ of the linear approximation $\langle a, x \rangle$ of $\langle b, F(x) \rangle$ is defined as

$$p = \frac{1}{2} + \epsilon_F(a, b)$$

which can be rewritten as

$$\epsilon_F(a, b) = \frac{\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)}{2^n} - \frac{1}{2}.$$

The relation between the Fourier transformation of F and the bias of a linear approximation is derived using

$$\mathrm{wt}(\langle b, F(\cdot) \rangle + \langle a, \cdot \rangle)) = 2^{n-1} - \frac{\widehat{F}(a, b)}{2}, \tag{1}$$

which implies

$$\epsilon_F(a, b) = -\frac{\widehat{F}(a, b)}{2^{n+1}}.$$

Moreover, due to scaling reasons, it is often helpful to talk about the *correlation coefficient* of F. This is defined by

$$C_F(a, b) = 2\epsilon_F(a, b) = -\frac{\widehat{F}(a, b)}{2^n}$$

Given a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^m$, the value used to determine the complexity of both multidimensional linear attacks and statistical saturation attacks is

$$\mathrm{Cap}(F) = \sum_{y \in \mathbb{F}_2^m} \frac{(2^{-n}|\{x \in F_2^n \mid F(x) = y\}| - 2^{-m})^2}{2^{-m}}$$

which is called *capacity* in [4]. In [5] the *squared euclidian distance* was used which is defined as

$$D(F) = \sum_{y \in \mathbb{F}_2^m} \left(2^{-n}|\{x \in F_2^n \mid F(x) = y\}| - 2^{-m}\right)^2$$

and differers from $\mathrm{Cap}(F)$ by a factor of 2^m, i.e. $D(F) = 2^{-m}\,\mathrm{Cap}(F)$.

There is an important, and well known, relation between the capacity (or the squared Euclidian distance) and the Fourier transformation of F which we will use below (see for example [6]).

Lemma 1

$$\mathrm{Cap}(F) = 2^{-2n} \sum_{b \neq 0} \left(\widehat{F}(0,b)\right)^2 = \sum_{b \neq 0} \left(\widehat{C}_F(0,b)\right)^2$$

2.2 Linear Trails, Correlations and Linear Hull

Consider a mapping $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$ given as the composition of mappings, i.e. $F = F_n \circ F_{n-1} \circ \cdots \circ F_1$. The correlation $C_F(a,b)$ can in this case be computed using linear trails. A linear trail consists of an input mask a and output mask b and a vector $U = (u_1, \ldots, u_{r-1})$ with $u_i \in \mathbb{F}_2^n$. The correlation of the trail is defined as

$$C_F(a,b,U) = C_{F_1}(a,u_1)C_{F_2}(u_1,u_2)\cdots C_{F_{r-1}}(u_{r-2},u_{r-1})C_{F_r}(u_{r-1},b).$$

Now, using correlation matrices [7] or the Fourier transformation of composite mappings [8], one can prove that

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-1}} C_F(a,b,U).$$

In contrary to the piling-up lemma [1], no assumption of any kind has to be made for this equation to hold. In the case where F corresponds to a key-alternating iterative block cipher, that is when all F_i are the same up to an addition of a round key, one can rewrite the previous equation as

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-1}} (-1)^{s_U} |C_F(a,b,U)|, \tag{2}$$

where the signs $s_U \in \{0,1\}$ depend on the sign of $C_F(a,b,U)$ and on the round keys. More importantly, the value $|C_F(a,b,U)|$ is independent of the round keys and the only influence of changing the keys is a change of the signs s_U. Again, no assumption is necessary for Equation 2 to hold. What we understand as the *linear hull* is in fact nothing other than Equation 2.

An assumption that will be necessary to understand the distribution of the biases in a linear attack is the following.

Assumption 1. *The signs s_U in Equation 2 are independently and uniformly distributed with respect to the key.*

Note that assuming independent round keys does not necessarily imply Assumption 1. This is only the case when all trails U with non-zero correlation $C_F(a,b,U)$ are linearly independent. In any case, it is important to verify experimentally for each cipher at hand, that Assumption 1 holds, before it can be applied.

Another important result we are going to use (cf. Theorem 1 in [6] and Theorem 7.9.1 in [7]) is the following.

Proposition 1. *Let F be the encryption function of a key alternating block cipher and assume that all round keys are independent. The average squared bias (resp. correlation) between an input and an output mask is the sum of the squared biases (resp. correlations) over all linear trails between the input and the output mask, i.e.*

$$\frac{1}{|K|} C_F(a,b)^2 = \sum_{U \in (\mathbb{F}_2^n)^{r-2}} C_F(a,b,U)^2$$

2.3 Statistical Saturation Attacks

In this section we briefly outline the idea of statistical saturation attacks. We refer to [5] for details. Given an encryption function

$$e : \mathbb{F}_2^n \to \mathbb{F}_2^n$$

statistical saturation attacks study the distribution of e when some of its inputs are fixed. While in general one can imagine the restriction to the coset of any subspace $E \subset \mathbb{F}_2^n$ for the inputs and any subspace $E' \subset \mathbb{F}_2^n$ for the output, for simplicity we restrict ourselves to the case where one fixes the last s bits in the inputs and considers only the first t bits of the output. Thus we write

$$e : \mathbb{F}_2^r \times \mathbb{F}_2^s \to \mathbb{F}_2^t \times \mathbb{F}_2^u \tag{3}$$

$$e(x,y) = \left(e^{(1)}(x,y), e^{(2)}(x,y)\right) \tag{4}$$

where $r+s = t+u = n$ and $e^{(1)}(x,y) \in \mathbb{F}_2^t$, $e^{(2)}(x,y) \in \mathbb{F}_2^u$. For convenience we denote by h_y the restriction of e by fixing the last s bits to y and considering only the first t bits of the output, that is

$$h_y : \mathbb{F}_2^r \to \mathbb{F}_2^t$$
$$h_y(x) = e^{(1)}(x, y) \tag{5}$$

In a statistical saturation attack one considers the capacity of h_y, and the attack complexity is usually a constant times $1/\operatorname{Cap}(h_y)$.

Applying Lemma 1 to h_y we can rewrite the capacity of h_y in terms of Fourier coefficients of h_y.

$$\operatorname{Cap}(h_y) = 2^{-2r} \sum_{b \in \mathbb{F}_2^t} \left(\widehat{h}_y(0, b)\right)^2 = \sum_{b \in \mathbb{F}_2^t} \left(C_{h_y}(0, b)\right)^2 . \tag{6}$$

One fundamental problem in statistical saturation attacks is that a useful method to estimate this capacity was missing. However, in Section 6 we show that in fact statistical saturation attacks are closely related to multi-dimensional linear attacks and in particular this link provides the missing method to estimate the capacity of h_y.

3 On the Linear Hull Effect

Linear hulls have been studied already in [9] and since then have been used in a number of papers. The main idea is to consider several, sometimes a lot of, linear trails with the same input and output mask to decrease the complexity of linear attacks using Matsui's Algorithm 2. However, as Murphy [2] pointed out nicely, there are (at least) two problems often appearing in the literature. In this section we first recall those two very fundamental problems, and discuss their impact on the complexity of linear attacks. While our starting point is clearly the work of Murphy, our conclusions are quite orthogonal. More precisely, we think that a better statement than Murphy's conclusion that *there is no linear hull effect* is that *there is always a linear hull effect and we always have to deal with this*. We furthermore propose methods that allow us to make meaningful statements about the running time of linear attacks. In particular we show the following.

Theorem 2. *Under Assumption 1 in a linear attack using a single trail with squared bias ϵ^2, at least half of the keys yield to a squared bias of at least ϵ^2. Thus, the complexity of this linear attack is less than c/ϵ^2 in more than half of the cases, where c is a small constant.*

In a linear attack using many linear trails with the same squared bias $\epsilon_i^2 = \epsilon^2$ at least one quarter of the keys yield to a squared bias of at least $0.46 \cdot \sum_i \epsilon_i^2$. Thus, the complexity of this linear attack is less than $2.2c/(\sum_i \epsilon_i^2)$ in more than a quarter of the cases, where c is a small constant.

The first problem Murphy points out comes from an incorrect formalization. Consider the simplest case of two linear trails with the same input and output mask, but different intermediate masks. The piling-up lemma is used to estimate the bias (say ϵ_1, ϵ_2) for each of the equations. We assume that $\epsilon_1 \neq 0$ and $\epsilon_2 \neq 0$.

Denoting by α the input mask, by β the output mask and by γ_1, γ_2 the two key masks, we end up with the following two equations

$$\langle \alpha, p \rangle + \langle \beta, c \rangle = \langle \gamma_1, K \rangle \text{ and } \langle \alpha, p \rangle + \langle \beta, c \rangle = \langle \gamma_2, K \rangle$$

where the first equation holds with a bias ϵ_1 and the second with a bias ϵ_2. So far, so good, but now consider any fixed (extended) key K. There are mainly two cases to consider. First, one could have a key such that $\langle \gamma_1, K \rangle = \langle \gamma_2, K \rangle$. In this case one gets the same equation twice, implying that $\epsilon_1 = \epsilon_2$. On the other hand, a different key could yield $\langle \gamma_1, K' \rangle = \langle \gamma_2, K' \rangle + 1$, and in this case we conclude that $\epsilon_1 = -\epsilon_2$. As it is very likely that there is at least one key for each of the cases, the only possible choice for the biases is $\epsilon_1 = \epsilon_2 = 0$, contradicting our assumption. What went wrong? Where does the mistake come from? The main point here is that, by applying the piling-up lemma, one implicitly assumes independent and uniform distribution in each round. However, having the first trail at hand, we already know that the inputs are non-uniformly distributed, and thus this assumption is wrong. It is important to notice the difference to the approach using correlation matrices, that is Equation 2. *Here no assumption about independent or uniform distribution is involved.* Thus, one actually always deals with the expression

$$C_F(a, b) = \sum_{U \in (\mathbb{F}_2^n)^r} C_F(a, b, U).$$

but separating the different trails is not possible. This is why the correct conclusion is that there is always a linear hull effect and we have to deal with this.

So far, we recaptured the first of the two problems pointed out by Murphy, and apply the (known) theory of correlation matrices to overcome it. Now, let us have a closer look at Murphy's second point, which we present slightly differently. Assume someone found *one* linear trail with a non-zero bias ϵ. Usually, the conclusion is that an attack based on this bias (lets say using Matsui second algorithm) is a constant times $1/\epsilon^2$. But what if the attacker overlooked another trail with the same absolute bias. The correct correlation is given by

$$C_F(u, v) = ((-1)^{s_1} + (-1)^{s_2})2\epsilon,$$

where the value of s_1 and s_2 depend on the extended key. Then, assuming a random behavior of the signs, the total bias would be zero for half of the keys.

There are two important remarks to make. First, the absolute bias is actually key dependent and secondly *the average complexity is formally infinite.* Again, to make the point very clear, the original *attack did not take any linear hull effect into account and this is the reason why the claim about the attack complexity is wrong.*

Now the attacker tries to do better and can actually show that all other trails have a (much) smaller bias. Still, for some (maybe only one) keys the biases could cancel and the average complexity is again infinite.

Thus, due to the linear hull effect, which is always there, estimating the average complexity of the attack seems very difficult (and often turns out to be infinite).

In the example, where there are exactly two trails with the same bias, the attack will still work for half of the keys (and for this half even faster than estimated by looking at only one trail). This leads directly to a natural way to overcome this problem. Namely, instead of studying the average complexity, the median of the complexities should be studied.

Definition 1. *The median of the complexities $\tilde{C}$ is the value such that, for half of the keys the complexity of the attack is less than or equal to $\tilde{C}$. More generally, one could study the complexity C_p defined as the complexity such that the probability that for a given key the attack complexity is lower than C_p, is p.*

Note that $\tilde{C} = C_{1/2}$.

Studying the complexity C_p instead of the average complexity has several advantages. First, given the median of the biases $\tilde{\epsilon}$, the median of the complexity is simply $c/\tilde{\epsilon}^2$ (as the inverse is a monotone function). Secondly, the median (or general C_p) is actually a more interesting value to know than the average complexity, especially in the case where the latter is infinite.

Finally, let us see what happens when we ignore or overlook trails in the computation of $C_F(u, v)$. This was exactly where the trouble started for the average complexity. Let us assume we take n trails with correlations γ_i into account. Furthermore, let us denote by γ_p the correlation such that the probability that a given key yields a correlation larger or equal to γ_p is p, given the n trails. That is

$$\mathrm{Prob}_K\left(\left|\sum_i (-1)^{s_i}\gamma_i\right| > |\gamma_p|\right) = \frac{1}{2}.$$

Denoting the correlations of the remaining trails by η_j, we get

$$C_F(u, v) = \left(\sum_i (-1)^{s_i}\gamma_i\right) + \left(\sum_j (-1)^{s'_j}\eta_j\right).$$

With a probability of $1/2$ the sum $\sum_j (-1)^{s'_j}\eta_j$ has the same sign as the sum $\sum_i (-1)^{s_i}\gamma_i$. Thus

$$\mathrm{Prob}_K(|C_F(u, v)| > |\gamma_p|) \geq \frac{p}{2}. \tag{7}$$

This inequality implies that the probability of having a complexity less than a given bound, might actually be smaller than estimated due to linear trails that have not been taken into account. However, this probability drops by at most a factor of 2.

Coming back to the case where an attacker just considered one trail with bias ϵ. As we saw, it is not possible to conclude *anything* meaningful about the average, but using the above considerations, one can conclude that for at least half of the keys the data complexity is below c/ϵ^2.

One important point not discussed so far is how to estimate the medians of the correlations. Here we consider two cases. First, in an attack where only a

few trails are used, one can easily compute the median by running through all possible values for the signs. This gets infeasible when there are too many trails. However, in the case where one deals with many trails with the same absolute correlation, one can estimate the median nicely by using a normal approximation, as explained below. In Section 4 we furthermore show by example, how one can estimate the median in the case of many trails with different absolute values.

3.1 Many Trails with the Same Absolute Value

As already done before (see [10]) in the case of many linear trails with the same absolute value, the distribution of the correlation

$$C_F(a,b) = \sum_{U \in (\mathbb{F}_2^n)^{r-2}} (-1)^{s_U} |C_F(a,b,U)| = \sum_i (-1)^{s_i} 2\epsilon_i,$$

where ϵ_i are the absolute biases of the trails, can be approximated by a normal distribution. This approximation implicitly makes use of Assumption 1. Clearly, this assumption has to be justified for each cipher by experiments (and we do so below for the block cipher PUFFIN). Denoting by X the random variable corresponding to the bias, we will approximate its distribution by

$$X \sim \mathcal{N}(0, \sum \epsilon_i^2),$$

that is, X is normally distributed with mean zero and variance $\sigma^2 := \sum \epsilon_i^2$. The probability density function is thus given by

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}x^2}.$$

Again, when trying to compute the average complexity of the attack, that is, the mean value of the random variable c/X^2, it turns out that this value is formally infinite. Therefore, and for reasons outlined above, we focus on the median of the squared biases corresponding to the random variable $Y := X^2$. Denoting by F the cumulative distribution function of X, the cumulative distribution function G of $Y = X^2$ can be computed as

$$\begin{aligned} G(t) &= \text{Prob}(Y \leq t) = \text{Prob}(-\sqrt{t} \leq X \leq \sqrt{(t)}) \\ &= F(\sqrt{t}) + F(-\sqrt{t}) - 1 = 2F(\sqrt{t}) - 1. \end{aligned}$$

Using the relation to the normal distribution (or ask maple) we can simplify $G(t)$ to

$$G(t) = \text{erf}(\sqrt{\frac{t}{2\sigma^2}})$$

where erf is the Gauss error function. The median $\tilde{\epsilon}^2$ of Y is by definition the value $\tilde{\epsilon}^2$ such that $G(\tilde{\epsilon}^2) = \frac{1}{2}$. We get

$$\text{erf}(\sqrt{\frac{\epsilon_m^2}{2\sigma^2}}) = \frac{1}{2}$$

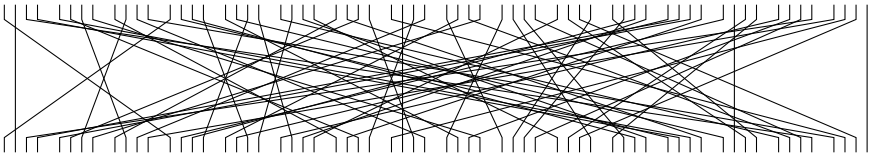
and using the approximation $\text{erf}^{-1}(1/2) \approx 0.48$ we conclude that $\epsilon_m^2 \approx 0.46\sigma^2$.

For completeness, we can furthermore compute the mean of Y as $E[Y] = \sigma^2$ which naturally corresponds to Proposition 1 (without using the normal approximation).

Thus, using the normal approximation and Equation 7 we can conclude that for a quarter of the keys the attack has a complexity lower than a small constant times $1/(0.46\sigma^2) \approx 2.21/\sigma^2$. A similar calculation shows that fraction of approximately 0.317 of the keys lead to an attack complexity of a small constant times σ^2.

4 Linear Hulls and PUFFIN

This section applies the ideas outlined above to the block cipher PUFFIN [11]. PUFFIN, a very PRESENT like SP-network, is a 64 bit block cipher with 32 rounds and a 128 bit master key. The only components we are interested in here are the linear layer and the sbox-layer. The linear layer is the following bit permutation.



The sbox-layer consists of 16 parallel executions of a single 4 bit sbox given by the following table.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	D	7	3	2	9	A	C	1	F	4	5	E	6	0	B	8

The main difference to PRESENT is that all components are involutions, thus allowing to save area when implementing the decryption circuit. For more details on PUFFIN we refer to [11].

We show how one can estimate quite precisely the distribution of the biases and thus in particular estimate the median of the attack complexity. Applying Theorem 2, our results indicate that, for at least a quarter of the keys, PUFFIN can be broken with a complexity less than 2^{58}. Because everything in our attack, except the estimation of the attack complexity, is a standard application of Matsui's second algorithm, we skip some details of the attack.

4.1 Linear Trails in PUFFIN

We focus only on trails, where all intermediate masks have Hamming weight one, i.e we have exactly one active sbox in each round. As it turns out, the highest median of biases can be expected for the input and output mask $e_1 =$ `0x2000000000000000`, that is between the first (counting from zero) bit of the plaintext and the first bit of the ciphertext. The number of all such trails together

with their absolute squared correlation can easily be computed for up to 31 rounds. Those results are summerized below. An entry t in this table at level ℓ and round r means that there are 2^t one bit trails, each with an absolute bias of 2^{-r-l}. Note that there is exactly one trail with maximal absolute bias of 2^{-r} and this trail is not included in the table.

Round\Level	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
4	2.00	-	1.00	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	2.81	-	2.58	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	3.32	-	4.00	2.00	1.00	-	-	-	-	-	-	-	-	-	-	-	-
7	3.81	-	5.13	3.32	4.09	2.58	-	-	-	-	-	-	-	-	-	-	-
8	4.17	-	6.00	4.58	5.95	4.46	3.32	-	-	-	-	-	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
28	7.70	-	13.41	12.69	17.88	18.04	21.51	22.05	24.47	25.18	26.89	27.63	28.84	29.50	30.34	30.87	31.41
29	7.80	-	13.61	12.90	18.19	18.36	21.91	22.48	24.98	25.72	27.52	28.30	29.58	30.30	31.22	31.81	32.43
30	7.89	-	13.80	13.09	18.47	18.66	22.30	22.89	25.48	26.25	28.12	28.94	30.30	31.07	32.06	32.71	33.41
31	7.99	-	13.99	13.29	18.76	18.95	22.67	23.28	25.95	26.75	28.70	29.55	30.99	31.80	32.86	33.57	34.34

4.2 Approximation of the Bias Distribution

In the case of PUFFIN we use a normal approximation to approximate the many linear trails for levels greater than 1. Denote by σ^2 the sum of squares of all those biases, i.e.

$$\sigma^2 = \sum_i \epsilon_i^2,$$

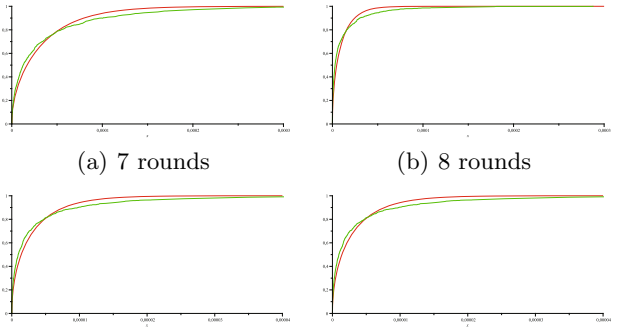
where ϵ_i runs through all one bit linear trails of level greater than 1 and smaller than 19. We denote the bias of the unique trail with maximal absolute bias by η. In order to incorporate this one maximal biased trail as well, we expect that the cumulative distribution function G of the total bias as is given by

$$G(t) = \frac{1}{2}\left(F(t-\eta) + F(t+\eta)\right), \tag{8}$$

where F is the cumulative distribution function of $\mathcal{N}(0, \sigma^2)$.

In order to justify the approximation, that is the implicit assumption on the random behavior of the signs, we experimentally compute the bias for 1000 keys for rounds 7 to 10. The results are shown in Figure 1.

Note that one reason for the small difference is that the estimates for very small biases are wrong, due to a naturally limited amount of samples. Apart from this small error, the distributions are quite close and in particular the median is predicted very precisely. Using Equation 8 one can easily compute the median numerically (using for example Maple). It turns out that the base two logarithm of the medians of the squared biases is almost an affine function. A good approximation of the median of the square bias for r rounds is given by $2^{-1.71r-3.13}$. In particular, applying Theorem 2 this implies that for a quarter of the keys, the data complexity of attacking r rounds of PUFFIN is proportional to $2^{1.71(r-1)+3.13}$. Experimental results for 7 to 12 round attacks (again using 1000 randomly chosen keys per round) indicate that using $4 \cdot 2^{1.71(r-1)+3.13}$ gives full gain, that is it recovers four key bits of the last round key successfully, in over 40% of the cases. In particular for $r = 32$, that is for the full PUFFIN, we get a complexity of about 2^{58}.



(a) 7 rounds (b) 8 rounds

(c) 9 rounds (d) 10 rounds

Fig. 1. Theoretical estimates vs. experimental bias for 7 to 10 rounds. The experimental data is based on 1000 randomly chosen keys for each round.

We expect that this is not the optimal attack. For example partial decryption of two instead of only one round might further reduce the data complexity (at the cost of increasing the computational complexity). Another improvement is likely obtained by applying the below mentioned statistical saturation attack (however estimating the exact data requirements is difficult). As the main objective of this section was to demonstrate how one can estimate the distribution of biases in the case of many linear trails, and use this finally to allow meaningful statements about the behavior of a linear attack, those improvements are out of scope of this paper and we leave them as a topic for further investigation.

5 Linear Hulls and PRESENT

PRESENT is a 64-bit block cipher developed by A. Bogdanov et al. [12] and was designed to be particular suitable for low-cost devices like RFID-tags. There are two versions, a 80 bit key version, called PRESENT-80 and a 128 bit version PRESENT-128. PRESENT is an substitution-permutation-network with 31 rounds and one final key exclusive-or at the end.

In the substitution layer (called sBoxLayer in PRESENT) a single 4-bit to 4-bit sbox is applied 16 times in parallel. The action of the sbox in hexadecimal notation is given by the following table.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

The permutation layer (called pLayer in PRESENT) is given as a simple bit permutation. Bit i of the current state is moved to bit position $P(i)$, where

$$P(i) = \begin{cases} 16 \times i \bmod 63 & \text{for } 0 \leq i \leq 62 \\ 63 & \text{for } i = 63 \end{cases}$$

The sbox in PRESENT $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ has been chosen to fulfill several criteria (see [12] for details) to ensure resistance against differential and linear cryptanalysis. We call a sbox fulfilling these criteria optimal.

Compared to a general linear layer the permutation layer of PRESENT provides relatively low diffusion. However, as stated in [12], for a bit permutation the pLayer is optimal in the sense that full dependency is reached already after a minimal number of rounds. After three rounds each of the 64 input bits influence each of the 64 output bits. For convenience we call such a permutation optimal.For more details on PRESENT we refer to [12].

5.1 Linear Attacks on PRESENT

Several papers discussed linear attacks on PRESENT. In [10] linear hulls where used to attack up to 25 rounds. Moreover, in [6] a multi dimensional linear attack on up to 26 rounds of PRESENT is described. The main reason why linear cryptanalysis works against up to 26 rounds of PRESENT is the relative high number of linear trails with only one active sbox per round. As the number of those linear trails was not discussed in [12] an important question is how sound the design of PRESENT is, when taking linear hulls into account. For $a, b \in \{0, \ldots, 63\}$ we denote by $N(a, b)$ the number of linear trails starting with the input bit a, ending with the output bit b and with exactly one active sbox per round.

5.2 On the Choice of Sbox and Permutation in PRESENT

Here we are interested in understanding the influence of the choice of the sbox and the bit permutation on the maximal number of trails $N(a, b)$. As explained below, using classification results on optimal sboxes and central digraphs, we can conclude that

- Given the PRESENT bit permutation, the PRESENT sbox is among the 8% worst of all optimal sboxes.
- Given the PRESENT sbox, the PRESENT bit permutation was the worst among roughly 2^{21} optimal permutations tested.

It should be noted that the PRESENT bit permutation is a natural choice, also reflected by the fact that it corresponds to what is known as the standard example in central digraphs (see below). However, the sbox in PRESENT was selected among all possible optimal sboxes as one with the smallest hardware circuit. We leave it as a topic for further investigation to explore, if there is a correlation between the size of the hardware circuit and the number of trails.

Furthermore, using those classification results we conclude that there is not a particular good bit permutation nor a particular good sbox. The number of one bit linear trails is mainly determined by the combination of both.

Most importantly, the results outlined below imply that the design principle of PRESENT is sound, in the following sense.

Fact 3. *For no combination of an optimal sbox and an optimal bit permutation, a linear attack based on one bit trails seems possible on* 31 *rounds.*

Influence of the Sbox. In this section we fix the bit permutation to the one used in PRESENT and compute the maximal number of trails for various choices of the sbox. Up to adding constants before and after the sbox, which clearly does not change any of these criteria and furthermore does not change the number of linear one bit trails, there are exactly 8064 such sboxes (see for example [13]). For each possible sbox fulfilling these criteria we computed the maximal number of one bit trails over all input/output bit combination for 31 rounds. It turns out that there are only 31 possible values for this maximum, ranging from 0 to approx 2^{39}. In Table 1 the number of sboxes (out of the 8064 possible ones) for each of the 30 possible values for the maximum of trails is shown. The PRESENT sbox has a maximal trail number of approx. $2^{38.17}$ and thus is among the 8 percent worst optimal sboxes with this respect.

Table 1. Maximal number of trails ($\log_2 \max_{a,b} N(a,b)$) vs number of sboxes (out of 8064) with the given trail number. In all cases the PRESENT bit permutation was used.

Maximal Number of Trails	$-\infty$	0	3.000	6.965	7.754	9.509	9.965	10.75	12.26
Number of sboxes	96	1056	192	48	48	192	768	48	48
Maximal Number of Trails	15.00	16.47	17.42	19.42	21.47	21.71	22.86	24.29	25.25
Number of sboxes	144	48	48	816	96	48	48	96	864
Maximal Number of Trails	25.30	25.54	25.75	26.03	26.04	26.08	26.33	26.34	26.37
Number of sboxes	96	96	192	96	96	96	96	96	96
Maximal Number of Trails	26.60	27.00	38.17	39.47					
Number of sboxes	96	1728	384	192					

Influence of the Bit Permutation. Different choices of an optimal bit permutation might result in different resistance against linear attacks. Below, we discuss the influence of the bit permutation on the number of one bit linear trails.

The first problem one encounters is, that it is actually not straight forward to find a permutation of 64 bits that, in a PRESENT style SP-network, yield to full dependency after three rounds. Optimal permutations are very rare and a naive trial to construct such objects is likely to fail. However, there is an interesting link to well studied objects in graph theory, namely central digraphs, that allows us to overcome this obstacle.

Definition 2. *Let n be an integer and D be a directed graph with n vertices. D is said to be a central digraph if there is a unique oriented path of length two between any two of its vertices.*

It is known (see [3]) that central digraphs exist only for $n = k^2$ and necessarily every vertex has in and out degree 4.

Any optimal bit permutation gives rise to a central digraph as follows. Thinking about the 16 sboxes as 16 vertices, we add a directed arc from vertex i to

vertex j if and only if there is an output bit of sbox i that gets mapped to an input bit of sbox j. Clearly, each of the 16 vertices has in and out degree 4. Being an optimal bit permutation now translates to the fact that each vertex (that is each sbox) can be reached from each vertex (that is any other sbox) in exactly two steps. A counting argument shows that such a path has to be unique and therefore the resulting graph is indeed a central digraph.

On the other hand, the converse construction, works as well. That is, given a central digraph of order 16 we can easily construct a optimal bit permutation. Note that this correspondence is unique only up to a permutation of the input and output bits of each sbox, as clearly permuting the input and output bits of an sbox does not change the corresponding graph (neither the optimality of the permutation). We thus have the following theorem.

Theorem 4. *Up to a permutation of the input and output bits of each sbox there is a one to one correspondence between optimal permutations of* 64 *bits and central digraphs of order* 16.

It is interesting to note that the PRESENT bit permutation actually corresponds to what is known as the "standard example" in terms of central digraphs. The vertex set of the standard example consists of all pairs (x, y) with $1 \leq x, y \leq k$ and we let $(x, y) \to (x', y')$ precisely when $y = x'$.

Using this link a couple of interesting questions can be easily answered. For example, one might want to avoid optimal permutations where a bit coming from one sbox gets mapped to the same sbox in the next round. However, it is well known (see for example [3]) that every central digraph on k^2 vertices has exactly k loops. This translate to the property that an optimal permutation on 64 bits will map exactly 4 input bits (coming from 4 distinct sboxes) back into the source sbox. Furthermore, for implementation reasons, one might want to chose a optimal permutation that is an involution. Again, it follows from the theory of central digraphs, that such a permutation does not exist.

For our purpose, the most interesting fact about central digraphs is that a classification of all central digraphs of order 16 is known (see [14]) and the actual number of (non-isomorphic) central digraphs of order 16 is reasonable small. Up to isomorphism there are precisely 3492 central digraphs of order 16. Thus, this link allows us to compare a great variety of choices for the optimal bit permutation.

We fixed the sbox to the one specified by PRESENT and computed the number of trails for all the 3492 possible central digraphs. Here, we randomly assigned the 4 incoming vertices and the 4 outgoing vertices to input and output bits of the sbox in 1000 different ways for each of the 3492 central digraphs. The result is shown in Table 2.

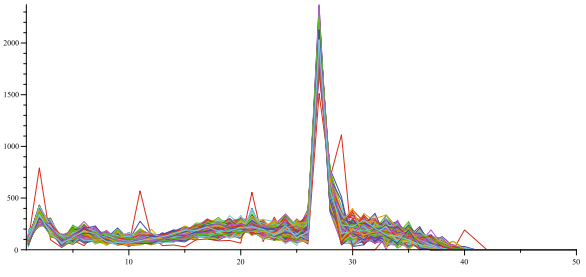
Again, the PRESENT permutation gives a maximal number of trails of approx $2^{38.17}$ and is therefore the worst of all $3492 \cdot 1000 \approx 2^{21}$ cases.

Influence of both Components. In a next step, instead of fixing the sbox and varying the permutation we varied both, that is we run through all 8064 possible sboxes and all 3492 possible central digraphs, again with 1000 randomly

Table 2. Maximal number of trails ($\lceil \log_2 \max_{a,b} N(a,b) \rceil$) vs number of optimal permutations (out of $3492 \cdot 1000$) with the given trail number. In all cases the PRESENT sbox was used.

♯Trails	1	2	3	4	5	6	7	8	9	10
♯permu.	2	5	7	8	16	21	46	51	86	144
♯Trails	11	12	13	14	15	16	17	18	19	20
♯permu.	234	351	559	990	1780	3260	5951	11187	21033	39284
♯Trails	21	22	23	24	25	26	27	28	29	30
♯permu.	71712	125520	205411	313402	431188	524990	553858	494911	359864	205508
♯Trails	31	32	33	34	35	36	37	38		
♯permu.	87875	26257	5344	941	184	18	1	1		

chosen permutations for the input and output bits of each sbox. As this is far too much data to be included in the paper, we only give parts of in the picture below.



There are two important observations to make. First, all curves follow pretty much the same pattern. This is to say there is no specially good or bad choice of sbox or bit permutation, only the combination of both can be good or bad. Second, in non of the $3492 \cdot 8064 \cdot 1000 \approx 2^{34}$ cases the number of trails was high enough to allow a linear attack based on one bit trails for 31 rounds.

6 Understanding Statistical Saturation Attacks

In this section we show how the capacity of statistical saturation attacks can be explained using tools from linear cryptanalysis. The main technical ingredient is an identity between the Fourier transform of a Boolean function and the biases of its restrictions (cf. Theorem V.1 in [15], see also Proposition 9 in [16])

Proposition 2 (Theorem V.1 in [15]). *Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a Boolean function. Furthermore, let E and E' be subspaces of $\mathbb{F}_2^n$ such that $E \cap E' = \{0\}$ and whose direct sum equals $\mathbb{F}_2^n$. For every $a \in \mathbb{F}_2^n$ let h_a be the restriction of f to the coset $a + E$ (h_a can be identified with a function on $\mathbb{F}_2^k$ where k is the dimension of E. Then*

$$\sum_{u\in E^{\perp}} \left(\widehat{f}(u)\right)^2 = |E^{\perp}| \sum_{a\in E'} \left(\widehat{h}_a(0)\right)^2. \tag{9}$$

Here $E^{\perp}$ is the orthogonal space of E.

Recall that we consider the encryption function $e : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and its restrictions by fixing the last s bits of the input and considering only the first t bits of its output, that is the function $h_y(x)$ defined by Equation 5. For statistical saturation attacks we are interested in the capacity given by Equation 6. Using the proposition above, we now state the main result of this Section.

Theorem 5. *With the above notation, the average capacity in statistical saturation attacks where the average is taken over all possible fixations is given by*

$$\begin{aligned}\overline{\mathrm{Cap}(h_y)} &= 2^{-s} \sum_{y\in\mathbb{F}_2^s} \mathrm{Cap}(h_y) = 2^{-2n} \sum_{a\in\{0\}\times\mathbb{F}_2^s, b\in\mathbb{F}_2^t\times\{0\}} \left(\widehat{e}(a,b)\right)^2 \\ &= \sum_{a\in\{0\}\times\mathbb{F}_2^s, b\in\mathbb{F}_2^t\times\{0\}} \left(C_e(a,b)\right)^2.\end{aligned}$$

Proof. By definition

$$\overline{C(h_y)} = 2^{-s} \sum_{y\in\mathbb{F}_2^s} \mathrm{Cap}(h_y) = 2^{-s} \sum_{a\in\{0\}\times\mathbb{F}_2^s, b\in\mathbb{F}_2^t} 2^{-2r} \left(\widehat{h}_a(0,b)\right)^2 \tag{10}$$

Applying identity (9) to all component function $\langle b, e\rangle$ (and its restrictions $\langle b, h_y\rangle$) where we choose $E = \mathbb{F}_2^r \times \{0\}$ and $E' = E^{\perp} = \{0\} \times \mathbb{F}_2^s$ yields

$$\sum_{u\in\{0\}\times\mathbb{F}_2^s} \left(\widehat{e}(u,b)\right)^2 = 2^s \sum_{a\in\{0\}\times\mathbb{F}_2^s} \left(\widehat{h}_a(0,b)\right)^2$$

Using this we deduce from (10).

$$\overline{\mathrm{Cap}(h_y)} = 2^{-2s-2r} \sum_{u\in\{0\}\times\mathbb{F}_2^s, b\in\mathbb{F}_2^t\times\{0\}} \left(\widehat{e}(u,b)\right)^2$$

as claimed. □

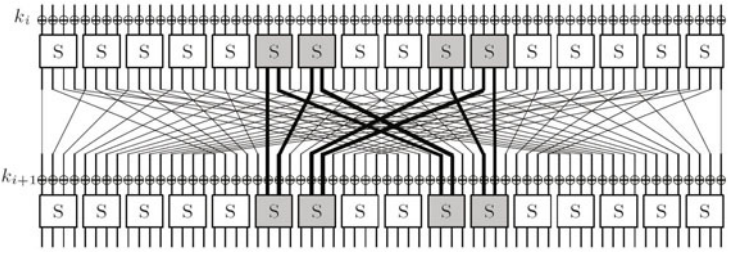
In general, statistical saturation attacks work well if one can identify subspaces $U, U'' \in \mathbb{F}_2^n$ (where U corresponds to output masks and U' to input masks) such that the sum $\sum_{u\in U', U\in E} \left(\widehat{e}(u,b)\right)^2$ is big. Moreover, Theorem 5 allows us to estimate the capacity, which is a first step in estimating the attack complexity of a statistical saturation attack.

From this point of view, statistical saturation attacks are very closely related to multi dimensional linear attacks. Especially, the statistical saturation attack on PRESENT presented in [5] and the multi dimensional linear cryptanalysis on PRESENT presented in [6] are in principle the same attack.

6.1 Statistical Saturation Attacks on PRESENT

For a description of PRESENT we refer to Section 5 and for more details to [12].

In this section we take a closer look at the statistical saturation trails used in [5] and explain its capacity using the above link to linear attacks. A picture of the trail used in [5] is given below.



In this trail the bits (counting from right to left, starting with 0)

$$S = \{21, 22, 25, 26, 37, 38, 41, 42\}$$

are fixed. At the output the same set of bits is used to compute the bias. In light of Theorem 5 this corresponds to taking $e : \mathbb{F}_2^{64} \to \mathbb{F}_2^{64}$ and its restrictions by fixing the 8 bits of the trail and restricting the output to the 8 bits in the trail as well $h_y : \mathbb{F}_2^{56} \to \mathbb{F}_2^8$.

Defining $E = \text{span}\{e_i \mid i \in S\}$, where $e_i \in \mathbb{F}_2^{64}$ is the canonical basis vector with a single one at position i (counting from zero), Theorem 5 states that

$$\overline{\text{Cap}(h_y)} = \sum_{a,b \in E} \left(C_e(a,b)\right)^2 .$$

To compute this capacity, we have to compute the correlation coefficients $C_e(a,b)$ for $a, b \in E$.

Like in Section 5 we restrict to a, b of weight one. This was done as well in [10,6], the argument being that it can be expected that those correlation coefficients have a much higher absolute value. Again, this assumptions is confirmed by the experimental data, see below. Given $a, b \in E$ of weight one, we recalled in Section 5 that there are many possible linear trails, starting with the input mask a and ending in the output mask b where all intermediate masks have weight one as well. Recall that we denoted the number of these linear trails by $N(a,b)$. Furthermore it is easy to compute the exact number of such paths for any pair (a,b).

The correlation $C_e(a,b,U_i)$ of the linear trails U_i, using the fact that in each round the bias is 2^{-3}, is given by $C_e(a,b,U_i) = 2^{-2R}$. Applying Proposition 1, the average square correlation is given by

$$\left(\overline{C}_e(a,b)\right)^2 = 2^{-4R} N(a,b),$$

and

$$\overline{C(h_y)} = \sum_{a,b \in E} \left(\overline{C}_e(a,b)\right)^2 \approx 2^{-4R} \sum_{\substack{a,b \in E \\ \mathrm{wt}(a)=\mathrm{wt}(b)=1}} N(a,b) \tag{11}$$

We compared experimental computations of $C(h_y)$, averaged over 100 different keys and 10 different values of y for each key with the results of the approximation (11). Except for the first two rounds the experimental date follow quite closely the approximation.

Round	2	3	4	5	6	7	8	9
$\log_2 \sum N(a,b)$	5.00	6.00	7.32	8.64	9.97	11.34	12.72	14.10
approx. (11)	−11.00	−14.00	−16.68	−19.36	−22.03	−24.66	−27.28	−29.90
experimental	−10.38	−13.82	−16.27	−18.90	−21.60	−24.13	−26.78	−29.26

The next observation that is immediate from looking at the numbers $N(a,b)$ is that this trail is likely to not be the best choice. Indeed, using the trail defined by fixing the same input bits as before, i.e. using $S = \{21, 22, 25, 26, 37, 38, 41, 42\}$ but this time restricting the output to the bits $S' = \{21, 23, 29, 31, 53, 55, 61, 63\}$ gives better results theoretically. Defining $E' = \mathrm{span}\{e_i \mid i \in S'\}$, the sum $\sum_{a \in E, b \in E'} N(a,b)$ is higher compared to the original trail (for example the capacity for 9 rounds is 2^{-29} instead of $2^{-29.9}$). Again, we verified this behavior experimentally and the experimental data confirm the approximations used quite nicely.

7 Conclusion and Further Work

We explained in detail why an estimate of the complexity of linear attacks is difficult and statements on the average complexity are often wrong. This is a very fundamental problem and we conclude that a paradigm shift from studying the average complexity to studying the median of the complexity is necessary. To simplify statements on the median, an important problem for further research is to give a general lower bound of the median in terms of the capacities of trails. In the case where the correlation for all trails have the same absolute value, this is not too difficult. However, as shown in Section 3 in this case an approximation by a suitable normal distribution provides nice results anyway.

Furthermore, we explained in Section 6 that statistical saturation attacks are almost identical to multidimensional linear attacks. This link allowed us to nicely estimate the average capacity of statistical saturation attacks. Of course, knowing only the average capacity for statistical saturation and multidimensional linear attacks suffers from the same problems as knowing the average in linear attacks. Namely, a useful statement on the running time is difficult. One important topic of further research is therefore to extend the ideas outlined in Section 3 to these cases.

Acknowledgment. The author likes to thank Sean Murphy for very valuable comments.

References

1. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
2. Murphy, S.: The Effectiveness of the Linear Hull Effect. Technical Report, RHUL-MA-2009-19 (2009)
3. Knuth, D.: Notes on central groupoids. J. Combin. Theory 8, 376–390 (1970)
4. Hermelin, M., Cho, J.Y., Nyberg, K.: Multidimensional extension of matsui's algorithm 2. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 209–227. Springer, Heidelberg (2009)
5. Collard, B., Standaert, F.X.: A statistical saturation attack against the block cipher present. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
6. Cho, J.Y.: Linear cryptanalysis of reduced-round present. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 302–317. Springer, Heidelberg (2010)
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
8. Carlet, C.: Vectorial (multi-output) Boolean Functions for Cryptography. Cambridge University Press, Cambridge (to appear)
9. Nyberg, K.: Linear approximation of block ciphers. In: Santis, A.D. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
10. Ohkuma, K.: Weak keys of reduced-round present for linear cryptanalysis. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 249–265. Springer, Heidelberg (2009)
11. Cheng, H., Heys, H.M., Wang, C.: Puffin: A novel compact block cipher targeted to embedded digital systems. In: Fanucci, L. (ed.) DSD, pp. 383–390. IEEE, Los Alamitos (2008)
12. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: Present: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
13. Leander, G., Poschmann, A.: On the classification of 4 bit s-boxes. In: Carlet, C., Sunar, B. (eds.) WAIFI 2007. LNCS, vol. 4547, pp. 159–176. Springer, Heidelberg (2007)
14. Kündgen, A., Leander, G., Thomassen, C.: Switchings, extensions, and reductions in central digraphs (2010) (preprint)
15. Canteaut, A., Carlet, C., Charpin, P., Fontaine, C.: On cryptographic properties of the cosets of r(1, m). IEEE Transactions on Information Theory 47(4), 1494–1513 (2001)
16. Carlet, C.: Boolean Functions for Cryptography and Error Correcting Codes (to appear)

Domain Extension for MACs Beyond the Birthday Barrier

Yevgeniy Dodis[1] and John Steinberger[2]

[1] Department of Computer Science, New York University
dodis@cs.nyu.edu
[2] Institute for Theoretical Computer Science, Tsinghua University, China
jpsteinb@gmail.com

Abstract. Given an n-bit to n-bit MAC (e.g., a fixed key blockcipher) with MAC security ε against q queries, we design a variable-length MAC achieving MAC security $O(\varepsilon q\,\mathrm{poly}(n))$ against queries of total length qn. In particular, our construction is the first to break the "birthday barrier" for MAC domain extension from noncompressing primitives, since our security bound is meaningful even for $q = 2^n/\mathrm{poly}(n)$ (assuming ε is the best possible $O(1/2^n)$). In contrast, the previous best construction for MAC domain extension for n-bit to n-bit primitives, due to Dodis and Steinberger [11], achieved MAC security of $O(\varepsilon q^2(\log q)^2)$, which means that q cannot cross the "birthday bound" of $2^{n/2}$.

1 Introduction

Most primitives in symmetric-key cryptography are built from block ciphers, such as AES. In this paper, we will concentrate on the question of designing variable-input-length (VIL) message authentication codes (MACs) from block ciphers. This question is very well studied, as we survey below. However, with few exceptions, most existing constructions and their analyses make the following two assumptions: (a) **Pseudorandomness**: the block cipher is modeled as a pseudorandom permutation (PRP); and (b) **Secrecy of Intermediate Results**: the attacker only learns the input/output behavior of the corresponding VIL-MAC, but does not learn any of the intermediate results. As observed by Dodis et al. [9, 10, 11], each of these assumptions might either be unnecessarily strong, or simply too unrealistic in the following two scenarios.

DOMAIN EXTENSION OF MACS. This is our main question. Since the desired MAC primitive only needs to be *unpredictable*, it would be highly desirable to only assume that the block cipher is unpredictable as well, as opposed to pseudorandom. Indeed, it seems that assuming the block cipher is unpredictable is a *much weaker* assumption than assuming full pseudorandomness: to break the latter, all one needs to do is to predict one bit of "random-looking" information about the block cipher with probability just a little over $1/2$, while the former requires one to fully compute the value of the block cipher on a new point with non-trivial probability. For example, in the non-uniform model, any block cipher (in fact, even non-trivial pseudorandom generator) with an n-bit key can be very efficiently distinguished from random with advantage $2^{-n/2}$ [8, 11]. To the best of our knowledge, no such lower bounds are known for breaking unpredictability, meaning that close to 2^{-n} MAC security might be possible for such a block cipher.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 323–342, 2011.

To put it differently, while we hope that existing block ciphers are actually PRPs, it seems quite reasonable to assume that their MAC security could be *noticeably better* than their PRP security. Thus, if we can design a VIL-MAC whose security is tightly related to the *unpredictability* of the block cipher, this VIL-MAC might be more secure than the MAC whose analysis assumes the *pseudorandomness* of the cipher.

Of course, one might hope that existing block-cipher based VIL-MACs, such as CBC-MAC [5, 26] and HMAC [3, 6] (whose compression function, under the hood, also uses a block cipher), are already secure when the block cipher is unpredictable. Unfortunately, as detailed in Dodis et al. [9, 10, 11] (see especially [11]), this is not the case: with few exceptions mentioned shortly, standard constructions are *completely insecure* when instantiated with unpredictable block ciphers, — often despite having simple proofs of security when one models the block cipher as a PRP.

RESILIENCE TO SIDE-CHANNELS. Even if the block cipher is a very good PRP, in practice many cryptographic implementations fall prey to various forms of side-channel attacks [13, 15, 16, 17, 28], where the physical realization of a cryptographic primitive can leak additional information, such as the computation-time, power-consumption, radiation/noise/heat emission etc. Thus, hardware people are paying special attention to securing block ciphers, such as AES, against such side-channel attacks. Although this might be a daunting task, it appears reasonable that specialized hardware implementations of AES might be pretty resistent to common forms of side-channel attacks. On the other hand, when the block cipher is used in some more complicated application, such as the design of a VIL-MAC, it might be hard or impractical to design a specialized "leakage-resilient" implementation for each such application, instead of doing so for a single, fixed-length building block (such as AES). Motivated by these considerations, Dodis et al. [9, 10, 11] proposed the model where the internals of the block cipher implementation are assumed secure, as usual, but all the external input/output behavior of the block cipher could potentially leak to the attacker (say, via side-channel attack).

To give this model a name while simultaneously making it more general, we say that a construction of a (deterministic) MAC P using some lower level keyed primitive(s) f is *transparent* (w.r.t. f), if (a) the key for P only consists of one of more keys for f; (b) when making a query M to P, the attacker not only gets $P(M)$, but also gets all the input/output pairs for every call to f made during the evaluation of $P(M)$. Since P is deterministic and all keys reside "inside" f, this indeed provides the attacker with the entire *transcript* of $P(M)$, short of what is happening during the calls to f. Coming back to our setting, we are interested in building a *transparent* VIL-MAC out of a block cipher. As we will see, this question is highly non-trivial even if the block cipher is assumed pseudorandom, let alone unpredictable. Indeed, as observed by [9, 10, 11], most existing VIL-MACs, including CBC-MAC [5, 26] and HMAC [3, 6], are *completely insecure* when the intermediate results are leaked, even when instantiated with a PRP.

OUR MAIN RESULT. Motivated by these applications, we ask the same question as Dodis et al. [9, 10, 11], which simultaneously addresses both of the above concerns.

Question 1. Can one build a *transparent* VIL-MAC P out of a block cipher f which is only assumed *unpredictable*?

As already mentioned, most standard VIL-MACs built from block ciphers fail to address either MAC-preservation or transparency (even with a PRP). So we turn to the

known secure approaches. As it turns out, all of them followed the principle of An and Bellare [2] of first constructing a compressing *Weakly Collision Resistant* (WCR)[1] hash function F from m to n bits, for some fixed $m > n$, then iterating this fixed-length WCR F using some variant of the Merkle-Damgård transform, and finally composing the output with a freshly keyed block cipher. As argued by Preneel and van Oorschot [27], any construction of this kind can achieve at most *birthday security*. Translated to the MAC-preservation setting, even if our original MAC f cannot be forged with probability ε using q queries, the resulting VIL-MAC P cannot have security greater than $O(\varepsilon q^2)$, meaning that q cannot cross $2^{n/2}$, even is ε is assumed to be (the best possible) $1/2^n$.

Interestingly, even achieving birthday security turns out to be challenging when the block cipher is only assumed unpredictable. The first secure construction of Dodis and Puniya [10], based on the Feistel network, only achieved security $O(\varepsilon q^6)$. The bound was then improved to $O(\varepsilon q^4)$ by Dodis, Pietrzak and Puniya [9] using the "enhanced CBC" construction. Finally, Dodis and Steinberger [11] showed (nearly) birthday security $\tilde{O}(\varepsilon q^2)$ using a new analysis of the Shrimpton-Stam [29] compression function. All these constructions were also transparent.

We ask the question if it is possible to build (hopefully, transparent) VIL-MACs from block ciphers with *beyond birthday security*. Most ambitiously, if f cannot be forged with probability ε using q queries, we would like to build a VIL-MAC P with security close to $O(\varepsilon q)$, meaning our security is meaningful even for values of q approaching 2^n, provided ε is assumed to be (the best possible) $1/2^n$. As our main result, we answer this question in the affirmative. Informally (see Section 4 for more details),

Theorem 1. *There exist fixed polynomials $a(n)$ and $b(n)$ and a construction P of a transparent VIL-MAC from an n-bit block cipher f, such that the rate[2] of P is $a(n)$ and the MAC security ε' of P against q' queries of total length qn is at most $O(b(n)q\varepsilon)$, where ε is the MAC-security of f against q queries. In particular, this bound is meaningful for q (and q') approaching 2^n.*

OTHER RELATED WORK. As we mentioned, the question of achieving beyond-birthday security for building VIL-MACs from unpredictable block ciphers was open prior to our work. In fact, the only domain extension results for MACs with beyond birthday security we obtained just recently by Yasuda [31] and Lee and Steinberger [18]. However, both results started with a shrinking MAC from strictly more than $2n$ to n bits. As we will see below, building such shrinking MACs (with beyond birthday security) from unpredictable block ciphers is highly non-trivial, and will be one of the key challenges we resolve on route to proving our main result. (However, we note that our result does not[3] simply reduce to building a $2n$ to n bit MAC from an n-bit to n-bit MAC.)

Another related area is that of for building VIL *pseudorandom functions* (PRFs) with beyond birthday security from PRPs, or more generally, fixed-length PRFs. In particular, several such constructions were found by [1, 4, 20, 23, 24, 25]. However, it is easy to see that none of them work either for the MAC domain extension, or even

[1] WCR security states that it is infeasible to find collisions in F given oracle access to F.

[2] Defined as the average number of calls to the block cipher f per n-bit input block.

[3] We cannot just build (say) a beyond birthday $3n$ to n bit MAC and then compose it with the beyond birthday VIL-MAC constructions of [18, 31], as each construction would lose a factor of q in exact security, resulting in already known "birthday" security $O(\varepsilon q^2)$.

for building VIL-MACs (let alone PRFs) when the intermediate computation results are leaked. For example, the corollary of our main result, giving a *transparent* VIL-MAC from a (q, ε_{prp})-secure PRP with security $\varepsilon_{prp} + \tilde{O}(q/2^n)$, appears to be new.

Perhaps the closest work to ours is a paper of Maurer and Tessaro [22], who showed how to build a variable-length random oracle from an n-to-n bit random oracle. Their construction, analyzed in the indifferentiability framework of [7, 21], has fixed polynomial rate (just like our construction) and security $2^{(1-\delta)n}$, for any $\delta > 0$. However, the two settings appear incomparable. On the one hand, the Maurer-Tessaro paper has to build an "indifferentiability simulator" for their setting (which required "input extraction" not required in our setting). However, they assumed a truly random function, and could use various probability calculations in deriving their result. In our setting, the block cipher is only unpredictable, and we have to make an explicit reduction to unforgeability, which makes matters substantially more delicate.

1.1 Outline of Our Construction

Our construction is quite involved, although we abstract it into several self-contained layers. As a side benefit, some of these layers (see below) are of potentially independent interest, and could be used for other purposes.

STEP 1: REDUCING TO $3n$-TO-$2n$ WCR AND $2n$-TO-n MAC. First, we notice that the above mentioned birthday limitation [27] of the An-Bellare approach no longer holds provided we build a WCR hash function F from m to $2n$ bits (for some $m > 2n$, say $m = 3n$). Namely, "birthday on $2n$ bits" might still give good enough security 2^n. However, even if we succeed in doing so with beyond birthday security (which will be one of our key results), we now also have to build a "final" MAC G from $2n$ to n bits. Thus, using known techniques but with different parameters (see Lemma 1 and Figure 1), our problem reduces to building beyond birthday WCR F from $3n$ to $2n$ bits and a beyond birthday MAC G from $2n$ to n bits.

STEP 2: REDUCING TO COVER-FREE FUNCTIONS. It so turns out that both of these tasks—i.e. the construction of the WCR function F and the construction of the MAC G—can be achieved from a more powerful (keyed) primitive which we introduce, called a *cover-free* function. Informally, a keyed function g from $\{0,1\}^m$ (recall, we will have $m = 3n$) to $(\{0,1\}^n)^t$ (for some parameter t), where $g(s) = (z_1(s), \ldots, z_t(s)) \in (\{0,1\}^n)^t$, is called *cover-free* (CF) if, given oracle access to g, it is infeasible to produce a sequence of (distinct) queries $s_1, s_2, \ldots, s_q \in \{0,1\}^m$ such that, for some $1 \leq j \leq q$, $z_\ell(s_j) \in \{z_\ell(s_1), \ldots, z_\ell(s_{j-1})\}$ for all $\ell \in [t]$. In other words, for each new query s_j one of the coordinates of $g(s_j)$ must be "uncovered" by previous coordinates of that index. The case $t = 1$ corresponds to the standard m to n bit WCR security, however better (and in particular beyond-birthday) cover-free security can be achieved with larger values of t.

First, as depicted on the left side of Figure 2, we can compose a CF g with t independently keyed block ciphers $f_1, \ldots f_t$, by setting $G(s) = f_1(z_1) \oplus \ldots \oplus f_t(z_t)$, where $g(s) = (z_1, \ldots, z_t)$. We show that the resulting G is easily seen to be a secure MAC from m bits to n bits. More precisely, the MAC security of G is tightly related to the CF security of g and the MAC security of f (see Lemma 2). Intuitively, a new forgery for G will give a new forgery for at least one of the f_ℓ's, by the CF security of g. Since $m = 3n > 2n$, this already gives us the needed $2n$ to n bit MAC.

More interestingly, as depicted on the right side of Figure 2, we show how to compose a CF function g with $2t$ independently keyed block ciphers $f_1, \ldots f_t, f'_1, \ldots, f'_t$ (in a variant of the "double-pipe" mode of [19]) to get a WCR function F from m bits to $2n$ bits. Moreover, the WCR security of F will be "roughly" $O(\varepsilon' + q\varepsilon)$, where ε' is the CF security of g and ε is the MAC security of f (see Lemma 3). Thus, as long as we can build CF g with security ε' close to $O(q\varepsilon)$, the WCR security of F will also be such. The proof of this result critically uses the bin-filling bin-guessing games of Dodis and Steinberger [11].

Summarizing the discussion above, our task of building a VIL-MAC P thus reduces to building a CF function g with security $\varepsilon' \approx O(q\varepsilon)$ where ε is the MAC security of the underlying n-bit to n-bit primitive f. We also wish to build the CF function g with t as small as possible (which is relevant since the efficiency of P, including the size of the key, is proportional to t). See Lemma 4.

STEP 3: BUILDING CF FUNCTIONS. This is, by far, the most involved part of our construction. The inspiration for this construction came from the afore-mentioned paper of Maurer and Tessaro [22], who showed how to build a VIL random oracle from an n-to-n bit random oracle. As we mentioned already, the setting of [22] is incomparable to our setting, especially since we cannot assume that our block cipher is (pseudo)random. However, our actual construction of CF functions is quite similar to the corresponding "cover-free" layer of the construction of [22], although we made some changes (actually, simplifications) to the construction of [22], and our analyses are completely different. Our CF construction has three layers which we informally call combinatorial, cryptographic and algebraic. An impatient reader can look at Figure 3 for a concrete example (with $t = 3$ and other notation explained below).

STEP 3A: USING INPUT-RESTRICTING FAMILIES. This purely combinatorial step is precisely the same as in [22], and is also the most expensive step of our construction. We will use an *unkeyed* function E from $\{0,1\}^m$ to $(\{0,1\}^n)^r$ (here r is a parameter) called an *input-restricting function family* (IRFF; see Definition 1). Intuitively, an IRFF has the property that after any q queries $s_1 \ldots s_q$ to E, the number Q of new inputs s for which the r-tuple $E(s)$ is covered by the union of $E(s_1), \ldots, E(s_q)$ is "not much larger" than q, and this should be true even when q is almost 2^n. Recall, our final goal is to ensure that it is hard to produce *any* such new input s. While IRFFs do not (and *cannot*!)[4] quite get us there, they ensure that there are not that many choices for the attacker of which new inputs to "cover" by old inputs.

We discuss the known constructions of IRFFs in Section 4, but mention that the constructions of IRFFs are completely combinatorial, and closely related to constructions of certain types of highly unbalanced bipartite expander graphs. While well-studied, these types of expander graphs are not yet completely understood, and in particular the "extreme" setting of parameters relevant to our case has not been the object of much attention. Therefore, although the existence of IRFFs with "good parameters" is known (and lead to the asymptotic bound claimed in Theorem 1), the concrete constructions are pretty inefficient. Nevertheless, as these parameters and efficiency are improved by future research in computational complexity, so will our final construction.

[4] Because they do not have a key and do not rely on any computational assumptions.

STEPS 3B-C: ADDING CONFUSION AND MIXING. Recall, IRFFs convert our input s into an r-tuple $(x_1. \ldots, x_r)$. To get the final t-tuple $(z_1, \ldots, z_t)$ for our CF function g, we can imagine repeating the following two-step precedure (steps 3b and 3c) t times, each time with a freshly keyed block cipher $\mathbf{F}$ (so the total number of block cipher keys for g will be t). First, we pass all r values $x_1, \ldots, x_r$ through the block cipher $\mathbf{F}$ ("confusion step"), getting the values $y_1. \ldots, y_r$. This is the cryptographic "confusion" layer. Then we algebraically "mix" all $2r$ values $(x_1 \ldots x_r, y_1 \ldots y_r)$ through a fixed, degree-r multivariate polynomial p (see Equation 3). This gives us one of the t outputs values $z_1 \ldots z_t$.

The intuition for these last two steps is hard to explain (and, indeed, our analysis is quite involved). At a high level, the confusion step (evaluating $\mathbf{F}(x_1) \ldots \mathbf{F}(x_r)$) is certainly needed to make a reduction to unforgeability, while the mixing step uses the fact that low-degree polynomials have few roots, so a "non-trivial" collision on the output of p will enable one to guess one of the values y_ℓ we are trying to forge. Of course, the difficulty is to make a successful guess for when and where the non-trivial collision to p will happen, with probability roughly $1/Q$, where Q is the guarantee given by IRFF (so Q is close to q). It turns out, there is a trivial strategy to make such a guess with "birthday" probability $1/Q^2 \approx 1/q^2$, even when $t = 1$. Of course, such probability is too low, and this is why we repeat steps 3b-c t times, for an appropriately chosen parameter t. We then show that the required guessing strategy can be reduced to the analysis of two bin-filling bin-guessing games. The relevance of such games to the domain extension of MACs was first introduced by Dodis and Steinberger [11]. Unfortunately, these two games are significantly more complicated than the game of [11] or than the game used in the proof of Lemma 3. Nevertheless, as our most involved technical step, we show that both games can be won with probability roughly $1/(Q \cdot Q^{1/t})$. Thus, by choosing $t > \log Q$ (say, $t = n$), we get the desired bound $O(1/Q) \approx O(1/q)$.

EFFICIENCY. Our final VIL-MAC construction uses $5t$ keys for f, where we recall that the minimal value of $t \approx \log q \leq n$. Theoretically, we can reduce key material down to a single key for f, by "keying" f via fixed, reserved input bits. Namely, to simulate (at most) $5n$ keys this way we need to reserve $\lceil \log_2(5n) \rceil$ bits of input (and "truncate" the same number of bits in the output), effectively reducing the block length of the construction from n down to $n' = n - \lceil \log_2(5n) \rceil$. Due to the output truncation, we now also need to guess the missing $\lceil \log_2(5n) \rceil$ output bits not returned by our forger, incurring an (acceptable) additional $O(n)$ degradation of the security bound.

Our final VIL-MAC also achieves rate roughly proportional to $O(rt) = O(rn)$. Achieving a low value of r (coming from the combinatorial IRFF part) is more problematic (see Section 4), although existentially one can also make $r = O(n)$. So the best rate we can hope to achieve using our approach is $O(n^2)$. Therefore, we primarily view our result as an important *feasibility* result, much like the result of Maurer and Tessaro [22]. Nevertheless, our feasibility opens the door for future, potentially more efficient constructions.

2 Preliminaries

A *keyed function family* is a map $f : \{0,1\}^\kappa \times Dom(f) \to \{0,1\}^v$ where $Dom(f) \subseteq \{0,1\}^*$. The strings in $\{0,1\}^\kappa$ are the *keys* of f and we write $f_k(x)$ for $f(k,x)$ for $k \in \{0,1\}^\kappa$ and $x \in Dom(f)$.

For MACs we consider the following game, where A is a halting adversary with oracle access to f_k:

Game Forge(A, f):
$k \leftarrow \{0,1\}^\kappa$; $(x, y) \leftarrow A^{f_k}$
If $x \in Dom(f)$, $f_k(x) = y$ and x was not a query of A then A wins, otherwise A looses.

We define the insecurity of f as a MAC to be

$$\mathbf{InSec}_f^{\mathrm{mac}}(T, q, \mu) := \max_A \Pr[A \text{ wins Forge}(A, f)]$$

where the maximum is taken over all adversaries A making at most q queries of total combined length at most μ (after padding, if any) and of "running time" at most T. The "running time" is defined to be the total running time of the experiment, including the time necessary to compute the answers to A's queries. Moreover we "bill" the final verification query $f_k(x)$ (and its length) to A (so that A must in fact make $q-1$ queries if $x \in Dom(f)$; seen another way, we ask A to verify its own forgery, if it attempts one). When f has fixed input length (i.e. $Dom(f) = \{0,1\}^m$ for some $m \in \mathbb{N}$) then μ is a function of q and it is convenient to elide the last argument, writing $\mathbf{InSec}_f^{\mathrm{mac}}(T, q)$ instead of $\mathbf{InSec}_f^{\mathrm{mac}}(T, q, \mu)$.

The *weak collision resistance* or "wcr" security of a function family f is measured as the maximum advantage of an adversary in finding a collision for a randomly keyed member of f when given oracle access to this member. We write $\mathbf{InSec}_f^{\mathrm{wcr}}(T, q)$ for the maximum such advantage over all adversaries A making at most q queries of running time at most T. (Here we do not measure the total query length, as we will only measure the wcr security of fixed input length constructions.) We skip a formal pseudocode-based definition of this standard notion, but mention that the adversary must make the queries verifying its collision, not merely output a colliding pair.

Given a block length n and a message x, we let $\mathrm{Pad}_n(x)$ be a suffix-free encoding of x of length a multiple of n bits (for example, the standard Merkle-Damgård padding of x, which appends the length of x as the last block[5]). Furthermore, given two keyed compression functions $F : \{0,1\}^{\kappa_1} \times \{0,1\}^{3n} \to \{0,1\}^{2n}$, $G : \{0,1\}^{\kappa_2} \times \{0,1\}^{2n} \to \{0,1\}^n$ we define a keyed function $\mathrm{MD}[F, G] : \{0,1\}^{\kappa_1+\kappa_2} \times \{0,1\}^* \to \{0,1\}^n$ by

$$\mathrm{MD}[F, G]_{k_1^*, k_2^*}(x) = G_{k_2^*}(F_{k_1^*}(x_b \| F_{k_1^*}(x_{b-1} \cdots F_{k_1^*}(x_1 \| 0^{2n}) \cdots)))$$

where $\mathrm{Pad}_n(x) = x_1 x_2 \cdots x_b$ and each x_i has n bits, for all $k_1^* \in \{0,1\}^{\kappa_1^*}, k_2^* \in \{0,1\}^{\kappa_2}$ (see Fig. 1).

The proof of the following (standard) lemma is given in in the full version [12]:

Lemma 1. *Let $F : \{0,1\}^{\kappa_1} \times \{0,1\}^{3n} \to \{0,1\}^{2n}$, $G : \{0,1\}^{\kappa_2} \times \{0,1\}^{2n} \to \{0,1\}^n$, and consider $\mathrm{MD}[F, G]$ as a function of key space $\{0,1\}^{\kappa_1+\kappa_2}$. Then, for $q = \mu/n$,*

$$\boldsymbol{InSec}_{\mathrm{MD}[F,G]}^{\mathrm{mac}}(T, \tilde{q}, \mu) \le \boldsymbol{InSec}_F^{\mathrm{wcr}}(T, q) + \boldsymbol{InSec}_G^{\mathrm{mac}}(T, q)$$

Informally speaking, Lemma 1 reduces our task to building, from an n-bit to n-bit primitive f, compression functions F and G such that F has beyond-birthday wcr security

[5] This limits the message length to at most 2^n blocks, but this is not a serious limitation for practical values of n.

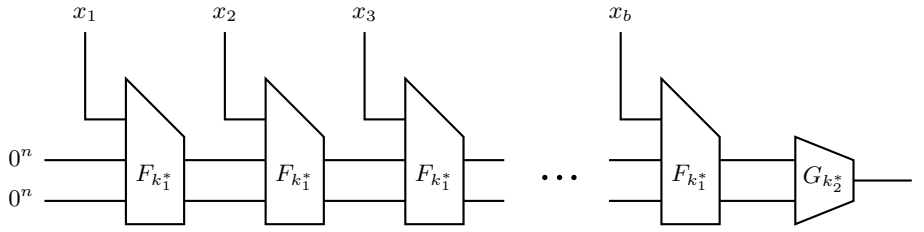


Fig. 1. A high-level view of our construction $\mathrm{MD}[F, G]$. The input x is padded in a suffix-free manner into n-bit blocks $x_1, \ldots, x_b$. All wires shown carry n-bit values. $F_{k_1^*} : \{0,1\}^{3n} \to \{0,1\}^{2n}$ and $G_{k_2^*} : \{0,1\}^{2n} \to \{0,1\}^n$ are compression functions keyed by independent keys k_1^*, k_2^*.

and G has beyond-birthday mac security, where these securities must be based only the mac security of f (i.e., breaking the wcr security of F must imply breaking the mac security of f, and breaking the mac security of G must likewise imply breaking the mac security of f).

To the latter end we introduce in this paper the notion of a *cover-free* keyed function family $g : \{0,1\}^\kappa \times \{0,1\}^m \to (\{0,1\}^n)^t$. Here t is a parameter of the definition and we write the output of $g_k(x)$ as $(z_1^k(x), \ldots, z_t^k(x)) \in (\{0,1\}^n)^t$ where $z_i^k(x) \in \{0,1\}^n$ for each i; later we will simply write $(z_1(x), \ldots, z_t(x))$ when the dependence on a key k is understood. In the cover-free game, an adversary adaptively queries g_k on distinct points $s_1, s_2, \ldots \in \{0,1\}^m$, and wins if for some j each block of output of $g_k(s_j)$ is "covered" by a previous output, in the sense that $z_\ell^k(s_j) \in \{z_\ell^k(s_i) : i < j\}, 1 \le \ell \le t$. The following game formalizes this:

Game $\mathrm{Cover}(A, g)$:
$k \leftarrow \{0,1\}^\kappa$;
If A^{g_k} makes distinct queries $s_1, \ldots, s_q \in \{0,1\}^m$ to g_k such that
$z_\ell^k(s_j) \in \{z_\ell^k(s_i) : i < j\}, 1 \le \ell \le t$, for some $j \le q$,
Then A wins; Otherwise, A looses.

We define the cover-free (CF) insecurity of g as

$$\mathbf{InSec}_g^{\mathrm{cover}}(T, q) := \max_A \Pr[A \text{ wins } \mathrm{Cover}(A, g)]$$

where the maximum is taken over all adversaries A making at most q queries and of running time at most T, with the same conventions as above on the running time. We (informally) say that a function family is *cover-free* to mean it has small cover-free insecurity.

Given a (cover-free) function family $g : \{0,1\}^\kappa \times \{0,1\}^m \to (\{0,1\}^n)^t$ where the ℓ-th block of g_k is given by the function $z_\ell^k : \{0,1\}^m \to \{0,1\}^n$ and a function family $f : \{0,1\}^{\kappa'} \times \{0,1\}^n \to \{0,1\}^n$ we define the composed function family $f \circ g : \{0,1\}^{\kappa + t\kappa'} \times \{0,1\}^m \to \{0,1\}^n$ by

$$(f \circ g)_{k k_1 \cdots k_t}(s) = \bigoplus_{\ell=1}^{t} f_{k_\ell}(z_\ell^k(s))$$

where $k \in \{0,1\}^\kappa$ and $k_1, \ldots, k_t \in \{0,1\}^{\kappa'}$, and $k k_1 \cdots k_t$ is a shorthand for the concatenation of $k, k_1, \ldots, k_t$. See Figure 2. We also define a *parallel composition*

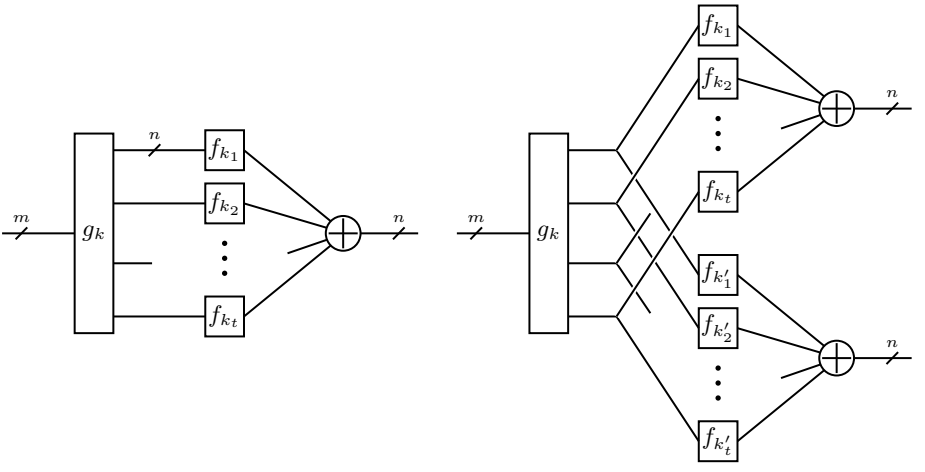


Fig. 2. On the left, the composition $(f \circ g)_{kk_1 \cdots k_t} : \{0,1\}^m \to \{0,1\}^n$. On the right, the parallel composition $(f \mathbin{\overline{\circ}} g)_{kk_1 \cdots k_t k'_1 \cdots k'_t} : \{0,1\}^m \to \{0,1\}^{2n}$.

$f \mathbin{\overline{\circ}} g : \{0,1\}^{\kappa+2t\kappa'} \times \{0,1\}^m \to \{0,1\}^{2n}$ of f and g, defined by

$$(f \mathbin{\overline{\circ}} g)_{kk_1 \cdots k_t k'_1 \cdots k'_t}(s) = (f \circ g)_{kk_1 \cdots k_t}(s) \| (f \circ g)_{kk'_1 \cdots k'_t}(s).$$

In other words, $f \mathbin{\overline{\circ}} g$ is simply the concatenation of two functions $f \circ g$ instantiated with the same g-key but independent f-keys.

Recall that our construction $\mathrm{MD}[F, G]$ takes as parameters keyed compression functions $F : \{0,1\}^{\kappa_1} \times \{0,1\}^{3n} \to \{0,1\}^{2n}$ and $G : \{0,1\}^{\kappa_2} \times \{0,1\}^{2n} \to \{0,1\}^n$. Given a cover-free function family $g : \{0,1\}^\kappa \times \{0,1\}^{3n} \to (\{0,1\}^n)^t$ and a function family $f : \{0,1\}^{\kappa'} \times \{0,1\}^n \to \{0,1\}^n$, we will set $\kappa_1 = \kappa + 2t\kappa'$, $\kappa_2 = \kappa + t\kappa$, and define

$$F_{k_1^*}(s) = (f \mathbin{\overline{\circ}} g)_{k_1^*}(s) \tag{1}$$

$$G_{k_2^*}(r) = (f \circ g)_{k_2^*}(0^n \| r) \tag{2}$$

for all $s \in \{0,1\}^{3n}$, $r \in \{0,1\}^{2n}$, $k_1^* \in \{0,1\}^{\kappa_1}$, $k_2^* \in \{0,1\}^{\kappa_2}$. The specification of our construction is thus now reduced to defining the cover-free function family g. We note that the n-bit to n-bit function family f is a parameter of the scheme (not constructed from any lower-level primitive) whereas g must be instantiated from f, and its cover-free security reduced to the mac security of f; see the next section for details on the construction of g.

Recall that, by Lemma 1, we are interested in bounding $\mathbf{InSec}_F^{\mathrm{wcr}}(T, q)$ and $\mathbf{InSec}_G^{\mathrm{mac}}(T, q)$ in terms of $\mathbf{InSec}_f^{\mathrm{mac}}(T, q)$. Towards this goal, we give two lemmas that upper bound $\mathbf{InSec}_{f \overline{\circ} g}^{\mathrm{wcr}}(T, q)$ and $\mathbf{InSec}_{f \circ g}^{\mathrm{mac}}(T, q)$ as a function of $\mathbf{InSec}_g^{\mathrm{cover}}(T, q)$ and $\mathbf{InSec}_f^{\mathrm{mac}}(T, q)$. The proofs of both lemmas are given in the full version [12].

Lemma 2. *Let* $g : \{0,1\}^\kappa \times \{0,1\}^m \to (\{0,1\}^n)^t$, $f : \{0,1\}^{\kappa'} \times \{0,1\}^n \to \{0,1\}^n$. *Then*

$$\boldsymbol{InSec}_{f \circ g}^{\mathrm{mac}}(T, q) \leq \boldsymbol{InSec}_g^{\mathrm{cover}}(T, q) + t \cdot \boldsymbol{InSec}_f^{\mathrm{mac}}(T, q).$$

Lemma 3. *Let* $g : \{0,1\}^{\kappa} \times \{0,1\}^{m} \to (\{0,1\}^{n})^{t}$, $f : \{0,1\}^{\kappa'} \times \{0,1\}^{n} \to \{0,1\}^{n}$. *Then*

$$\boldsymbol{InSec}^{\mathrm{wcr}}_{f \circ g}(T,q) \leq \boldsymbol{InSec}^{\mathrm{cover}}_{g}(T,q) + 2tq \log q \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T + \tilde{O}(q), q).$$

(We note that, unlike Lemmas 1 and 2, the proof of Lemma 3 is not a triviality. In particular, it requires the analysis of a balls-and-bins game of the type used in [11].) Combining Lemmas 1, 2 and 3 we directly obtain:

Lemma 4. *Let* $g : \{0,1\}^{\kappa} \times \{0,1\}^{3n} \to (\{0,1\}^{n})^{t}$, $f : \{0,1\}^{\kappa'} \times \{0,1\}^{n} \to \{0,1\}^{n}$ *and let* F, G *be as in* (1), (2). *Then, if* $q = \mu/n$,

$$\boldsymbol{InSec}^{\mathrm{mac}}_{\mathrm{MD}[F,G]}(T,\tilde{q},\mu) \leq 2 \cdot \boldsymbol{InSec}^{\mathrm{cover}}_{g}(T,q) + (2tq \log q + t) \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T + \tilde{O}(q), q)$$

Lemma 4 reduces our problem to constructing the cover-free function family g from the function family f such that $\mathbf{InSec}^{\mathrm{cover}}_{g}(T,q)$ can be bounded in terms of $\mathbf{InSec}^{\mathrm{mac}}_{f}(T,q)$. This is the topic of the next section, and the paper's main technical achievement.

When a keyed function is built from a smaller primitive, where the function's key consists of a finite set of keys for the smaller primitive (which is potentially called several times with different keys), the notions of MAC, WCR and cover-free securities naturally extend to a *transparent* model, where the adversary receives a full transcript of the function's computation at each query, up to calls to the primitive (namely, calls to the lower-level primitive appear as oracle calls in the transcript, so as not to reveal the primitive's keys). In fact, *all* results and proofs of this paper can be (easily) interpreted and are valid in this stronger "transparent" model. However, to keep the presentation simple, we will not further remind this from here on.

3 Building Cover-Free Function Families from MACs

This section contains our main result, the construction of a cover-free function family based on n-bit to n-bit primitives, that achieves beyond-birthday security assuming only good MAC security from the primitives. We note in passing that an *unkeyed* function $g : \{0,1\}^{m} \to (\{0,1\}^{n})^{t}$ cannot be cover-free against information-theoretic adversaries unless $t2^{n} \geq 2^{m}$ or unless t is as large as the desired query security, which gives values of t that are too large to be practical for most settings.

Our construction uses the notion of an *input-restricting function family* (IRFF), introduced by Maurer and Tessaro [22]. The following definition is slightly modified for our purposes.

Definition 1. *Let* $K = K(n) \leq 2^{n}$ *and let* $m > n$. *A* (m,n,r,δ,K)-IRFF *is a set* $\mathcal{E}$ *of functions* $E_1, \ldots, E_r : \{0,1\}^{m} \to \{0,1\}^{n}$ *such that* (i) $r \geq 2$ *and* $E_h(s) \neq E_{h'}(s)$ *for all* $s \in \{0,1\}^{m}$ *and all* $h \neq h'$, (ii) *for all* $s \neq s' \in \{0,1\}^{m}$ *there exists* $h \in \{1,\ldots,r\}$ *such that* $E_h(s) \neq E_h(s')$, *and* (iii) *for any subset* $\mathcal{U} \subseteq \{0,1\}^{n}$ *such that* $|\mathcal{U}| \leq rK$ *we have*

$$\left|\{s \in \{0,1\}^{m} : E_h(s) \in \mathcal{U} \text{ for all } h = 1 \ldots r\}\right| \leq \delta|\mathcal{U}|.$$

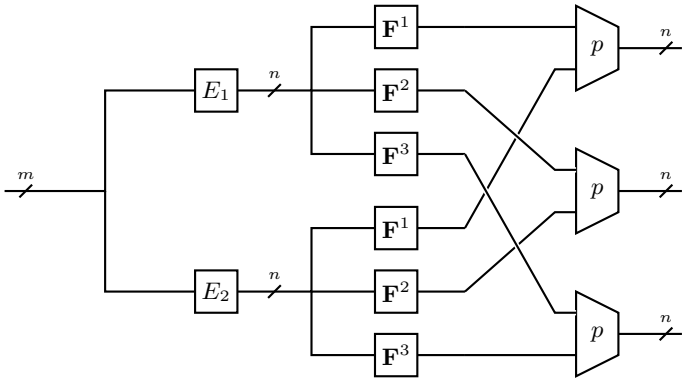


Fig. 3. Illustration of the cover-free function $\mathbf{Z}^{\mathcal{E},r,t}_{m,n} : \{0,1\}^m \to (\{0,1\}^n)^t$ for parameters $r = 2, t = 3$. Additional wires not shown on the diagram carry the input of each $\mathbf{F}^i$ to the i-th copy of p.

The constructions of input-restricting function families are discussed in Section 4.

Our cover-free function family is also adapted from [22]. The construction takes as parameters $m \geq n$ as well as integers $r, t \geq 1$ and a (m, n, r, δ, K)-IRFF $\mathcal{E} = \{E_1, \ldots, E_r\}$. Let $\mathbf{F}^1, \ldots, \mathbf{F}^t$ be n-bit to n-bit primitives (later to be instantiated as members of function family $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, possibly fixed-key block-ciphers). The construction also uses a (concrete, unkeyed) function $p : \{0,1\}^{2rn} \to \{0,1\}^n$ described below. Let $\mathbf{Z}^{\mathcal{E},r,t}_{m,n} : \{0,1\}^m \to (\{0,1\}^n)^t$ be defined by

$$\mathbf{Z}^{\mathcal{E},r,t}_{m,n}(s) = (z_1(s), \ldots, z_t(s))$$

where

$$z_\ell(s) = p(E_1(s), \ldots, E_r(s), \mathbf{F}^\ell(E_1(s)), \ldots, \mathbf{F}^\ell(E_r(s)))$$

for $1 \leq \ell \leq t$ (see Figure 3). From $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$ we obtain a keyed function family of signature $\{0,1\}^{t\kappa} \times \{0,1\}^m \to (\{0,1\}^n)^t$ by instantiating each $\mathbf{F}^\ell$ with a member of a function family $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$; however, we opt for the unkeyed notation (in which $\mathbf{F}^1, \ldots, \mathbf{F}^t$ are implicitly keyed) when possible to reduce notational overhead.

As for the function p, it is the polynomial

$$p(x_1, \ldots, x_r, y_1, \ldots, y_r) = \sum_{j=1}^{r} \sum_{i=1}^{r} x_i y_j^i \tag{3}$$

where $x_1, \ldots, y_r$ are n-bit strings treated as elements of the field $\mathbb{F}_{2^n}$. The only properties of p that matter are the two following:

I. **Invertibility.** For any $1 \leq j \leq r$ and any values $x_1, \ldots, x_r, y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_r$, $z \in \mathbb{F}_{2^n}$ such that $x_1, \ldots, x_r$ are not all zero, there are few values y_j such that $p(x_1, \ldots, x_r, y_1, \ldots, y_r) = z$, and these values y_j are efficiently enumerable.

II. **Collision Invertibility.** For any $1 \le j, j' \le r$ and any values $x_1, \ldots, x_r$, $y_1, \ldots, y_{j-1}, y_{j+1}, \ldots, y_r$, $x'_1, \ldots, x'_r$, $y'_1, \ldots, y'_{j'-1}, y'_{j'+1}, \ldots, y'_r \in \mathbb{F}_{2^n}$ such that $(x_1, \ldots, x_r) \neq (x'_1, \ldots, x'_r)$ there are few values $y_j = y'_{j'}$ such that

$$p(x_1, \ldots, x_r, y_1, \ldots, y_r) = p(x'_1, \ldots, x'_r, y'_1, \ldots, y'_r),$$

and these values are efficiently enumerable.

Both properties are easily verifiable from the fact that $p(x_1, \ldots, x_r, y_1, \ldots, y_r)$ is a polynomial of y_j of the type $c + x_1 y_j + \cdots + x_r y_j^r$, where c does not depend on y_j. Maurer and Tessaro use a different construction instead of p which does not obviously satisfy either property above, that requires enlarging the set of functions $\{\mathbf{F}^\ell\}$ to a set $\{\mathbf{F}^{\ell,v}\}$ where v ranges from 1 to $\lceil m/n + 1 \rceil$.

To state our main theorem, let $\mathsf{InvTime}(\mathcal{E}, q)$ be the amount of time required to list the values $\{s \in \{0,1\}^m : E_{h_0}(s) = v \text{ and } E_h(s) \in \mathcal{U} \text{ for } h \neq h_0\}$ for any given $h_0 \in [r]$, $v \in \{0,1\}^n$ and set $\mathcal{U} \subseteq \{0,1\}^n$ such that $|\mathcal{U}| \le rq$. We have:

Theorem 2. *Let $\mathcal{E}$ be a (m, n, r, δ, K)-IRFF, let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be a function family, and consider $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$ as a keyed function family of key space $\{0,1\}^{\kappa t}$ by setting $\mathbf{F}^\ell = f_{k_\ell}$ for any $k_1 \cdots k_t \in \{0,1\}^{\kappa t}$. Then*

$$\boldsymbol{InSec}^{\mathrm{cover}}_{\mathbf{Z}^{\mathcal{E},r,t}_{m,n}}(T, q) \le 6rQt^3Q^{1/t} \cdot \boldsymbol{InSec}^{\mathrm{mac}}_f(T_{mac}, q) \tag{4}$$

for any $q \le K$, where $Q = qr\delta$ and

$$T_{mac} = T + \tilde{O}(Qt) + qr\mathsf{InvTime}(\mathcal{E}, q) + \mathsf{RootTime}_r(n)$$

where $\mathsf{RootTime}_r(n)$ is the time required to find all the roots of a polynomial of degree r in a field of size $\mathbb{F}_{2^n}$. In particular, when $t = n$ and $q \le 2^n/(r\delta)$, we have

$$\boldsymbol{InSec}^{\mathrm{cover}}_{\mathbf{Z}^{\mathcal{E},r,t}_{m,n}}(T, q) \le (12r^2\delta n^3) \cdot q \cdot \boldsymbol{InSec}^{\mathrm{mac}}_f(T_{mac}, q)$$

Proof. Let A' be an adversary for the game $\mathrm{Cover}(\cdot, \mathbf{Z}^{\mathcal{E},r,t}_{m,n})$ that runs in time T and that has success probability $\varepsilon_{A'}$. It suffices to design an adversary B for the game $\mathrm{Forge}(\cdot, f)$ with advantage at least

$$\varepsilon_{A'}(6rQt^3Q^{1/t})^{-1}$$

and that runs in time T_{mac}.

B has access to a random member f_{k_0} of f. B chooses t random keys $k_1, \ldots, k_t \in \{0,1\}^\kappa$, and selects a random index $\ell_0 \in [t]$. Then B simulates A' with oracle $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$, instantianting the function $\mathbf{F}^\ell$ with f_{k_ℓ} if $\ell \neq \ell_0$ and instantianting $\mathbf{F}^{\ell_0}$ with f_{k_0}, using its oracle. Moreover B proceeds to "forget" the value of ℓ_0, treats each of the functions $\mathbf{F}^\ell$ as an oracle, and tries to forge any one of them (predicting their output on an unqueried input), making only one such forgery attempt during the game. Since B has chance $1/t$ of forging $\mathbf{F}^{\ell_0}$ if it does make a correct forgery, it suffices for B to make such a forgetful forgery with chance at least

$$\varepsilon_{A'}(6rQt^2Q^{1/t})^{-1}$$

in order for it to forge f_{k_0} with chance at least $\varepsilon_{A'}(6rQt^3Q^{1/t})^{-1}$.

It is easier to consider a modified version of A', which we call simply A, that directly issues $\mathbf{F}$-queries rather than $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$-queries; more precisely, A issues a sequence of queries $x_1, \ldots, x_{q'}$ where $q' \leq qr$ and each $x_j \in \{0,1\}^n$; B answers the query x_j with the tuple $(\mathbf{F}^1(x_j), \ldots, \mathbf{F}^t(x_j))$. We can assume A never makes the same query twice. We let $\mathcal{Q}_i = \{x_j : j \leq i\}$ and let $\mathcal{S}_i = \{s \in \{0,1\}^m : E_h(s) \in \mathcal{Q}_i \text{ for } 1 \leq h \leq r\}$ for $0 \leq i \leq q'$ (with $\mathcal{Q}_0 = \mathcal{S}_0 = \emptyset$). Note that

$$|\mathcal{S}_i| \leq |\mathcal{S}_{q'}| \leq |\mathcal{Q}_{q'}|\delta \leq qr\delta = Q$$

by the input-restricting property of $\mathcal{E}$. We also let $\Delta\mathcal{S}_i = \mathcal{S}_i \backslash \mathcal{S}_{i-1}$ for $1 \leq i \leq q'$ and put $z_\ell(\mathcal{C}) = \{z_\ell(s) : s \in \mathcal{C}\}$ for any $\mathcal{C} \subseteq \{0,1\}^m$ (which B can compute after it answers A's i-th query as long as $\mathcal{C} \subseteq \mathcal{S}_i$). We say A "wins the generous cover-free game" at the i-th query if there exists an $s \in \mathcal{S}_i$ such that $z_\ell(s) \in z_\ell(\mathcal{S}_i \backslash \{s\})$ for $1 \leq \ell \leq t$. Clearly, there exists an A of same running time as A' whose advantage ε_A in the generous game is at least as great as $\varepsilon_{A'}$, since A can simply simulate A' and ask the various $\mathbf{F}$-queries needed to compute the answers to A''s queries; by definition, A wins if A' wins $\text{Cover}(A', \mathbf{Z}^{\mathcal{E},r,t}_{m,n})$. (It is easy to check that if A' makes (distinct) queries $z_1, \ldots z_j \in \{0,1\}^m$ such that $z_\ell(s_j) \in \{z_\ell(s_i) : i < j\}$, then A wins the generous cover-free game by the time it has finished asking the queries necessary to compute the answer to the query s_j of A'.) Thus it is sufficient to have B forge one of the $\mathbf{F}$-functions with probability at least $\varepsilon_A(6rQt^2Q^{1/t})^{-1}$. We now view B as simply answering A's $\mathbf{F}$-queries (as opposed to computing answers to $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$-queries) though in reality B is running the whole computation, including the simulation of A' by A.

We view each value $s \in \mathcal{S}_i$ as a "bin" with t "slots"; the ℓ-th slot of bin s "receives a ball" or "becomes full" at query $j \geq i$ if $s \in S_j$ (namely, if the bin already exists at that point), if $z_\ell(s) \in z_\ell(\mathcal{S}_j \backslash \{s\})$, and if either $s \notin \mathcal{S}_{j-1}$ or $z_\ell(s) \notin z_\ell(\mathcal{S}_{j-1} \backslash \{s\})$. Once a bin receives a ball in a slot, the slot remains full. A slot cannot receive more than one ball, and bins are never removed; we note that no bins exist at the start, and that $|\Delta\mathcal{S}_i|$ bins are added at the i-th query. Under these definitions, A wins the "generous" cover-free game precisely if some bin becomes full (i.e., all its slots become full). It is helpful to picture A and B as playing an adversarial game in which A wins if it fills a bin without B forging one of the functions $\mathbf{F}^1, \ldots, \mathbf{F}^t$, and where B wins otherwise (in fact, we may even picture A as choosing the answers to its queries, while B observes and tries to guess an answer before it is revealed).

We say that ball ℓ of a bin $s \in \Delta\mathcal{S}_i$ is "early" if $z_\ell(s) \in z_\ell(\mathcal{S}_i \backslash \{s\})$ and "late" otherwise; thus a ball is early if and only if it is added to a bin at the same A-query which creates the bin. B plays one of two different forging strategies with equal probability. The first strategy is designed to prevent too many early balls from appearing in bins while the second strategy is designed to prevent A from filling a bin (the second strategy only functions well if not too many early balls appear in bins, whence the necessity of the first strategy). We name the two strategies "early prevention" and "late prevention"; despite these names, we emphasize the two strategies are not played sequentially; instead, B flips a coin at the start to decide which strategy to use.

We start by describing B's early prevention strategy. Let $Q = qr\delta$; as noted above, $Q \geq |\mathcal{S}_{q'}|$, so Q is an upper bound for the total number of bins created during the game. The goal of B's early prevention strategy is to prevent A from creating, for every

$1 \leq k \leq t$, $Q^{1-k/t}$ or more bins that each have k or more early balls in them. In other words, we only require this strategy to work (i.e. forge a function $\mathbf{F}^\ell$ with "good enough" probability) if there is some $1 \leq k \leq t$ such that $Q^{1-k/t}$ or more bins are created with k or more early balls in them.

We model the early prevention strategy via a slightly simplified balls-in-bins game described below. To connect this balls-in-bins game with the "real" game played by B and A, it is helpful to first review the process via which bins are created and early balls are added to them. Consider a query x_i made by A. Then

$$\Delta\mathcal{S}_i = \{s \in \{0,1\}^m : E_{h_0}(s) = x_i \text{ for some } h_0 \in [r] \text{ and } E_h(s) \in \mathcal{Q}_{i-1} \text{ for } h \neq h_0\}$$

and the elements of $\Delta\mathcal{S}_i$ are the new bins created by this query. Each bin $s \in \Delta\mathcal{S}_i$ has t slots and the "value" $z_\ell(s)$ of the ℓ-th slot of s is revealed when B makes the query $\mathbf{F}^\ell(x_i)$; after the value $z_\ell(s)$ is revealed, an early ball is added to the ℓ-th slot of s according to whether there exists an $s' \in \mathcal{S}_i \backslash \{s\}$ such that $z_\ell(s) = z_\ell(s')$ or not (notice that $z_\ell(s')$ is known at this point for all $s' \in \mathcal{S}_i$). Thus, the process of filling the newly created bins with early balls consists in t "phases" (the queries $\mathbf{F}^1(x_i), \ldots, \mathbf{F}^t(x_i)$, which are made sequentially by B), where the ℓ-th phase simultaneously reveals the values of the ℓ-th slots of all the new bins, and whether these slots receive early balls or not. The following balls-in-bins game thus abstracts the process of creation of new bins and early balls:

'EARLY PREVENTION' BALLS-AND-BINS GAME. This game is played between two adversaries $\overline{A}$ and $\overline{B}$. Parameters are integers $t, q', Q \geq 1$. Rules are as follows:

- The game proceeds in q' rounds. At round i, $\overline{A}$ announces some number $v_i \geq 0$ of bins such that $\sum_{j \leq i} v_j \leq Q$.
- At the beginning of each round the v_i bins are empty. Each bin has t slots. Each round consists of t phases. At the ℓ-th phase, $\overline{A}$ reveals which of the v_i bins have their ℓ-th slot "filled" by a "ball".
- Before each phase of each round, $\overline{B}$ is allowed to secretly predict a bin that will receive a ball at that phase; $\overline{B}$ wins if it makes a correct guess, but it is only allowed to make one guess during the entire game.
- Let $b_{k,i}$ be the number of bins that receive k or more balls at round i, and let $b_k = \sum_i b_{k,i}$ where the sum is taken over all the rounds. Then $\overline{A}$ is required to fill bins such that $b_k \geq Q^{1-k/t}$ for at least one value of k, $1 \leq k \leq t$.

In the full version [12] we exhibit a strategy for $\overline{B}$ that gives it at least $(t^2 Q^{1/t})^{-1}$ chance of winning the above game, regardless of $\overline{A}$'s strategy. Thus, if $Q^{1-k/t}$ or more bins each receive k or more early balls for some $1 \leq k \leq t$, and if B uses this strategy, B has chance $(t^2 Q^{1/t})^{-1}$ of correctly predicting, before the answer to some query $\mathbf{F}^\ell(x_i)$ is given, that the value returned by this query will result in slot ℓ of some (specific) bin $s \in \Delta\mathcal{S}_i$ receiving an early ball. To guess $\mathbf{F}^\ell(x_i)$, B further chooses a random $s' \in \mathcal{S}_i \backslash \{s\}$, and solves $z_\ell(s) = z_\ell(s')$ in order to guess $\mathbf{F}^\ell(x_i)$ (since s receives an early ball in slot ℓ precisely when there exists an $s' \in \mathcal{S}_i \backslash \{s\}$ such that $z_\ell(s) = z_\ell(s')$). To see that $z_\ell(s) = z_\ell(s')$ is really "solvable" two different cases must be considered, according to whether $s' \in \Delta\mathcal{S}_i$ or not. If $s' \notin \Delta\mathcal{S}_i$ then s' was created by an earlier A-query and the value of its slots are known, in particular the value $z_\ell(s')$ of its ℓ-th slot

is known. Let $\overline{x}_h = E_h(s)$ for $1 \le h \le r$, let $h_0 \in [r]$ be the unique index such that $\overline{x}_{h_0} = x_i$ and let $\overline{y}_h = \mathbf{F}^\ell(\overline{x}_h)$ for $1 \le h \le r$. Then all the values $\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_r$ are known to B except for the value $\overline{y}_{h_0}$, which it needs to guess using the equation

$$p(\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_r) = z_\ell(s'). \tag{5}$$

By condition (i) of Definition 1 $(\overline{x}_1, \ldots, \overline{x}_r) \neq (0, \ldots, 0)$ so, by the 'Invertibility' property of p, there are few values $\overline{y}_{h_0}$ that solve (5). More precisely, since $p(\overline{x}_1, \ldots, \overline{y}_r)$ is a nonzero polynomial of degree at most r in $\overline{y}_{h_0}$, B has to choose from the at most r roots of $p(\overline{x}_1, \ldots, \overline{y}_r) - z_\ell(s')$, where $z_\ell(s')$ is just a constant. In the second case, $s' \in \Delta\mathcal{S}_i$ and $z_\ell(s')$ is not known (like $z_\ell(s)$, it is about to be revealed). Let $\overline{x}'_h = E_h(s')$, let $h'_0 \in [r]$ be the unique index such that $\overline{x}'_{h'_0} = x_i$ and let $\overline{y}'_h = \mathbf{F}^\ell(\overline{x}'_h)$ for $1 \le h \le r$. Then all the values $\overline{x}'_1, \ldots, \overline{x}'_r, \overline{y}'_1, \ldots, \overline{y}'_r$ are known to B except $\overline{y}'_{h'_0}$, and B needs to solve

$$p(\overline{x}_1, \ldots, \overline{x}_r, \overline{y}_1, \ldots, \overline{y}_r) = p(\overline{x}'_1, \ldots, \overline{x}'_r, \overline{y}'_1, \ldots, \overline{y}'_r) \tag{6}$$

(this is $z_\ell(s) = z_\ell(s')$) for $\overline{y}_{h_0}, \overline{y}'_{h_0}$ (or at least for $\overline{y}_{h_0}$). But $\overline{y}_{h_0} = \overline{y}'_{h_0}$; since $\overline{x}_{h_0} = \overline{x}'_{h'_0} = x_i$; also, by the injectivity of $\mathcal{E}$, $(\overline{x}_1, \ldots, \overline{x}_r) \neq (\overline{x}'_1, \ldots, \overline{x}'_r)$, so it follows by the 'Collision Invertibility' property of p that there are few values $\overline{y}_{h_0} = \overline{y}'_{h'_0}$ solving (6); in fact these are the at most r different roots of $p(\overline{x}_1, \ldots, \overline{y}_r) - p(\overline{x}'_1, \ldots, \overline{y}'_r)$, considered as a polynomial in $\overline{y}_{h_0} = \overline{y}'_{h'_0}$. The term $\mathsf{RootTime}_r(n)$ in Theorem 2 accounts for B's root-finding costs, which are incurred only once in the computation.

Naturally, B's further guessing of s' and of the correct root $\overline{y}_{h_0}$ erodes its probability of making a correct forgery even it has correctly guessed an early ball is about to be added to a bin slot, but it is easy to bound this erosion: B has chance at least $1/|\mathcal{S}_i| \ge 1/Q$ of correctly guessing s' and chance at least $1/r$ of correctly guessing the root. Thus, if $Q^{1-k/t}$ or more bins each receive k or more early balls for some $1 \le k \le t$ and if B is using its 'early prevention' strategy (which we have just finished describing), then B has chance at least

$$\frac{1}{rQt^2Q^{1/t}}$$

of forging. As B uses this strategy with probability $\frac{1}{2}$, we can therefore assume that fewer than $Q^{1-k/t}$ bins receive k early balls for every $1 \le k \le t$, or else B already reaches the requisite probability of success of $\varepsilon_A(6rQt^2Q^{1/t})^{-1}$.

We now discuss B's 'late prevention' strategy. Here B attempts to prevent A from filling a bin with t balls by guessing the arrival of late balls. We note that, if a query $\mathbf{F}^\ell(x_i)$ results in some late ball being placed in the ℓ-th slot of bin s, then $s \notin \Delta\mathcal{S}_i$ (by definition of 'late') and so the values $z_1(s), \ldots, z_t(s)$ are already known prior to the answer of the query $\mathbf{F}^\ell(x_i)$. Moreover the fact that the query $\mathbf{F}^\ell(x_i)$ results in a late ball appearing in bin s means there is some $s' \in \Delta\mathcal{S}_i$ such that (i) $E_{h_0}(s') = x_i$ for some $h_0 \in [r]$, (ii) the queries $\mathbf{F}^\ell(E_h(s'))$ have already been made[6] for $h \neq h_0$, and (iii) $z_\ell(s) = z_\ell(s')$ (the value $z_\ell(s')$ will become known when $\mathbf{F}^\ell(x_i)$ is answered). Let $\overline{x}'_1 = E_1(s'), \ldots, \overline{x}'_r = E_r(s')$ (so $\overline{x}'_{h_0} = x_i$) and $\overline{y}'_1 = \mathbf{F}^\ell(\overline{x}'_1), \ldots, \overline{y}'_r = \mathbf{F}^\ell(\overline{x}'_r)$,

[6] This means A has made the queries $E_h(s')$ for $h \neq h_0$ so that, in fact, all queries $\mathbf{F}^{\ell'}(E_h(s'))$ for $1 \le \ell' \le t$ and $h \neq h_0$ have already been made (not just $\ell' = \ell$).

all of which are known to B except $\overline{y}'_{h_0}$. Then, if B has correctly guessed a late ball is going to appear in the ℓ-th slot of bin s *and* has correctly guessed the value of $s' \in \Delta\mathcal{S}_i$, it can predict $\mathbf{F}^\ell(x_i)$ by solving

$$p(\overline{x}'_1, \ldots, \overline{x}'_r, \overline{y}'_1, \ldots, \overline{y}'_r) = z_\ell(s) \tag{7}$$

for $\overline{y}'_{h_0}$, for which there are at most r solutions. (This is the second (and last) place we require the 'Invertibility' property of p.) Given these observations, the following balls-and-bins game clearly models B's 'late prevention' task, up to but not including guessing the root of (7):

'LATE PREVENTION' BALLS-AND-BINS GAME. This game is played between two adversaries $\overline{A}$ and $\overline{B}$. Parameters are integers $t, q', Q \geq 1$. Rules are as follows:

- The game involves "bins" with t slots each, where each slot can contain either contain a ball or not. At the beginning of the game, there are no bins. Bins are added to the game as described below, and never removed.
- The game proceeds in q' rounds, each of which consists of t "phases".
- At the beginning of round i, $\overline{A}$ announces some number $v_i \geq 0$ such that $\sum_{j \leq i} v_j \leq Q$. If $v_i = 0$, the round is skipped.
- At phase ℓ of round i, $1 \leq \ell \leq t$, $\overline{A}$ chooses a subset (possibly empty) of the currently existing bins that do not yet have a ball in their ℓ-th slot, and places balls in all of their ℓ-th slots, simultaneously. Moreover, $\overline{A}$ labels each ball placed with a number from 1 to v_i. (Multiple balls with the same label are allowed, and not all labels are required to appear.)
- At the end of round i, $\overline{A}$ introduces v_i new bins to the game, each possibly already containing some balls. Throughout the game, the total number of new bins introduced with k or more balls already in them must be less than $Q^{1-k/t}$ for all $1 \leq k \leq t$.
- Before each phase of each round, $\overline{B}$ is allowed to secretly predict a bin that will receive a ball at that phase and a label for that ball; $\overline{B}$ wins if it guesses both correctly. It is only allowed to make one guess during the game.
- $\overline{A}$ must fill some bin with t balls by the end of the game.

We note that the new bins introduced at the end of round i correspond to the elements of $\Delta\mathcal{S}_i$ and that v_i corresponds to $|\Delta\mathcal{S}_i|$. The "label" for a ball placed in a bin s at phase ℓ corresponds to an element $s' \in \Delta\mathcal{S}_i$ such that $z_\ell(s) = z_\ell(s')$, discussed above. (In the 'real game' between B and A several such elements s' may exist, so that more accurate modelization would allow $\overline{A}$ to choose a nonempty list of labels rather than a single label for each ball; however, seeking to minimize the guessing advantage of $\overline{B}$, $\overline{A}$ would automatically make each of these lists a singleton anyway.)

In the full version [12] we exhibit a strategy for $\overline{B}$ in the 'late prevention' game that succeeds with probability $(3Qt^2Q^{1/t})^{-1}$ regardless of $\overline{A}$'s strategy. The 'late prevention' strategy of B consists simply of coupling the $\overline{B}$-strategy mentioned above with a guessing of the root of (7). Thus, as long as fewer than $Q^{1-k/t}$ bins receive k or more early balls for $1 \leq k \leq t$, as long as A fills some bins with t balls and as long as B uses its late prevention strategy, B has chance at least

$$\frac{1}{3rQt^2Q^{1/t}}$$

of forging. Since B uses the 'late prevention' strategy with probability $\frac{1}{2}$, this concludes the proof.

4 Implications

Replacing g in Lemma 4 by our cover-free function $\mathbf{Z}^{\mathcal{E},r,t}_{m,n}$ and using Theorem 2 with $m = 3n$, we obtain:

Theorem 3. *Let $\mathcal{E}$ be a $(3n, n, r, \delta, K)$-IRFF, let $f : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$, and consider $\mathbf{Z}^{\mathcal{E},r,t}_{3n,n}$ as a keyed function family of key space $\{0,1\}^{\kappa t}$ like in Theorem 2. Define F, G by (1), (2) with $g = \mathbf{Z}^{\mathcal{E},r,t}_{3n,n}$. Then*

$$\begin{aligned}\boldsymbol{InSec}^{\mathrm{mac}}_{\mathrm{MD}[F,G]}(T, \tilde{q}, \mu) \le 12rQt^3Q^{1/t} \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T_{mac}, q) \\ + (2tq \log q + t) \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T + \tilde{O}(q), q)\end{aligned} \tag{8}$$

where $q = \mu/n$ and $Q = qr\delta$ as long as $q \le K$, and where

$$T_{mac} = T + \tilde{O}(Qt) + qr\,\mathsf{InvTime}(\mathcal{E}, q) + \mathsf{RootTime}_r(n).$$

In particular, when $t = n$ and $Q \le 2^n$ (i.e. $q \le 2^n/r\delta$) and $q \le K$ we have

$$\begin{aligned}\boldsymbol{InSec}^{\mathrm{mac}}_{\mathrm{MD}[F,G]}(T, \tilde{q}, \mu) \le 24r^2\delta n^3 q \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T_{mac}, q) \\ + (2nq \log q + n) \cdot \boldsymbol{InSec}^{\mathrm{mac}}_{f}(T + \tilde{O}(q), q)\end{aligned} \tag{9}$$

By default we shall apply the second part of Theorem 3, choosing $t = n$. In order to interpret (9) we need to know what values of r, δ and K are achievable via IRFFs and to know $\mathsf{InvTime}(\mathcal{E}, q)$ for those IRFFs, as this term dominates T_{mac}.

The question of instantiating the IRFF $\mathcal{E}$ was already studied by Maurer and Tessaro [22], who reduced it to the construction of certain types of highly unbalanced bipartite expander graphs. While well-studied, these types of expander graphs are not yet completely understood, and in particular the setting of parameters relevant to our case has not been the object of much attention. Here we mention bounds achieved by two explicit constructions as well as those achieved by a non-explicit, probabilistic construction. In all cases we set $m = 3n$. We note that $\mathsf{InvTime}(\mathcal{E}, q)$ can always be upper bounded by q^3 by appending three functions to the IRFF that read off each block of input via the identity. Moreover, we can easily enforce condition (i) of Definition 1 as long as $r \le 2^n$. Since the family sizes r in question are anyway polynomial in n, we assume these tweaks without further mention.

Existential construction. A probabilistic construction [22] achieves a $(3n, n, r, \delta, K)$-IRFF $\mathcal{E}$ with $r = O(n)$, $\delta \approx 1$ and $K = \Omega(\frac{2^n}{n})$. In this case $Q = qr\delta = O(nq)$. Then the right-hand side of (9) becomes

$$O(n^5 q) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q).$$

Assuming $\mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q) \approx 1/2^n$, $\mathrm{MD}[F, G]$ achieves query security up to $q = \Omega(2^n/n^5)$. However, this construction is inexplicit.

Expanders of [30]. Expanders of Ta-Shma, Umans and Zuckerman yield an explicit $(3n, n, r, \delta, K)$-IRFF $\mathcal{E}$ with $r = \mathrm{poly}(n)$, $\delta = \mathrm{poly}(n)$ and $K = \Omega(\frac{2^n}{\mathrm{poly}(n)})$. In this case $Q = q\mathrm{poly}(n)$. The right-hand side of (9) becomes

$$O(\mathrm{poly}(n)q) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q).$$

Assuming $\mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q) \approx 1/2^n$ we can then achieve query security up to $q = \Omega(2^n/\mathrm{poly}(n))$. (We note this construction is strictly better from all standpoints than the one presented by Maurer and Tessaro [22].)

Expanders of [14]. Expanders of Guruswami, Umans and Venkatesan yield an explicit $(3n, n, r, \delta, K)$-IRFF $\mathcal{E}$ with $r = n^{O(\frac{1}{\varepsilon})}$, $\delta = \mathrm{poly}(n)$ and $K = 2^{n(1-\varepsilon)}$ for any $\varepsilon \in (0, 1)$. In this case $Q = q\mathrm{poly}(n)n^{O(\frac{1}{\varepsilon})}$. We can set $t = \log(Q) = \log q + O(\frac{1}{\varepsilon}\log n)$. For constant ε the right-hand side of (9) again becomes

$$O(\mathrm{poly}(n)q) \cdot \mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q).$$

Assuming $\mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q) \approx 1/2^n$ the insecurity thus remains negligible as long as $q \leq K = 2^{n(1-\varepsilon)}$. The advantage of this construction is that it affords efficient inversion time of $O(q\,\mathrm{poly}(n))$ (as opposed to $O(q^3)$ for the previous two constructions).

Interpretation. The assumption $\mathbf{InSec}_f^{\mathrm{mac}}(T_{\mathrm{mac}}, q) \approx 1/2^n$ is only realistic as long as T_{mac} does not allow to do an exhaustive search over the key space of f; assuming the latter has size $2^\kappa \geq 2^n$, this implies that our upper bounds are only meaningful if $T_{\mathrm{mac}} \approx \mathsf{InvTime}(\mathcal{E}, q) \ll 2^\kappa$ (since T_{mac} is dominated by $\mathsf{InvTime}(\mathcal{E}, q)$). The first two constructions, which are only known to have $\mathsf{InvTime}(\mathcal{E}, q) = O(q^3)$, therefore only give a meaningful bound for $q \ll 2^{\kappa/3}$. Thus, with the current understanding of $\mathsf{InvTime}(\mathcal{E}, q)$, they might become beyond birthday only if $\kappa > 3n/2$ (and approach $q \approx 2^n$ only if $\kappa > 3n$). However, the last construction, having $\mathsf{InvTime}(\mathcal{E}, q) = O(q\,\mathrm{poly}(n))$, yields beyond-birthday security even if $\kappa = n$, which is the case of AES-128. Once again, though, we stress that the current limitations of our approach are due only to the limitations in the current constructions of expander graphs, and are not related to any "cryptographic" difficulties. Needless to say, future advances in the constructions of expander graphs will not only improve our parameters, but will likely have other applications in many areas of theoretical computer science.

Heuristic Instantiation. In practice, we expect to nearly match the good IRFF parameters of the existential construction (including $r = O(n)$ and $\delta = O(1)$) by simply implementing each $E_i : \{0,1\}^{3n} \to \{0,1\}^n$ as the XOR of three (independently keyed) fixed key blockciphers, i.e. $E_i(x\|y\|z) = f_{k_{i,1}}(x) \oplus f_{k_{i,2}}(y) \oplus f_{k_{i,3}}(z)$. We note that in

this case the $3r$ keys $k_{1,1}, \ldots, k_{r,3}$ do not constitute key material, but are instead fixed constants of the construction.

ACKNOWLEDGMENTS. Yevgeniy Dodis was supported by by NSF grants 1017471, 0831299, 0716690 and the Google Faculty Award. John Steinberger was supported by the National Natural Science Foundation of China Grant 60553001, by the National Basic Research Program of China Grant 2007CB807900, 2007CB807901 and by NSF grant CNS 0904380.

References

1. Aiello, W., Venkatesan, R.: Foiling birthday attacks in length-doubling transformations — Benes: a non-reversible alternative to Feistel. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 307–320. Springer, Heidelberg (1996)
2. 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)
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., Goldreich, O., Krawczyk, H.: Stateless Evaluation of Pseudorandom Functions: Security beyond the Birthday Barrier. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 270–287. Springer, Heidelberg (1999)
5. Bellare, M., Kilian, J., Rogaway, P.: The security of cipher block chaining. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 341–358. Springer, Heidelberg (1994)
6. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
7. 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)
8. De, A., Trevisan, L., Tulsiani, M.: Time Space Tradeoffs for Attacks against One-Way Functions and PRGs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 649–665. Springer, Heidelberg (2010)
9. 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)
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., Steinberger, J.: Message Authentication Codes from Unpredictable Block Ciphers. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 267–285. Springer, Heidelberg (2009), full version http://people.csail.mit.edu/dodis/ps/tight-mac.pdf
12. Dodis, Y., Steinberger, J.: Domain Extension for MACs Beyond the Birthday Barrier. In: EUROCRYPT 2011. LNCS, vol. 6632, pp. 323–342. Springer, Heidelberg (2011), full version of this paper http://people.csail.mit.edu/dodis/ps/optimal-mac.pdf
13. 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)
14. Guruswami, V., Umans, C., Vadhan, S.P.: Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. J. ACM 56(4) (2009)

15. 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. Commun. ACM 52(5), 91–98 (2009)
16. 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)
17. 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)
18. Lee, J., Steinberger, J.: Multi-property-preserving domain extension using polynomial-based modes of operation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 573–596. Springer, Heidelberg (2010)
19. 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)
20. Maurer, U., Pietrzak, K.: The security of Many-Round Luby-Rackoff Pseudo-Random Permutations. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 544–561. Springer, Heidelberg (2003)
21. Maurer, U., Renner, R., Holenstein, R.: 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)
22. Maurer, U., Tessaro, S.: Domain Extension of Public Random Functions: Beyond the Birthday Barrier. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 187–204. Springer, Heidelberg (2007)
23. Patarin, J.: Luby-Rackoff: 7 rounds are enough for $2^{n(1-\varepsilon)}$ security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 513–529. Springer, Heidelberg (2003)
24. 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)
25. Patarin, J., Montreuil, A.: Benes and Butterfly Schemes Revisited. In: ISISC 2005 (2005)
26. Petrank, E., Rackoff, C.: CBC MAC for Real-Time Data Sources. J. Cryptology 13(3), 315–338 (2000)
27. 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)
28. Quisquater, J.-J., Samyde, D.: ElectroMagnetic analysis (EMA): Measures and countermeasures for smart cards. In: Attali, I., Jensen, T.P. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
29. 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)
30. Ta-Shma, A., Umans, C., Zuckerman, D.: Lossless Condensers, Unbalanced Expanders, And Extractors. Combinatorica 27(2), 213–240 (2007)
31. Yasuda, K.: A double-piped mode of operation for MACs, PRFs and PROs: Security beyond the birthday barrier. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 242–259. Springer, Heidelberg (2009)

Statistical Attack on RC4
Distinguishing WPA

Pouyan Sepehrdad, Serge Vaudenay, and Martin Vuagnoux

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. In this paper we construct several tools for manipulating pools of biases in the analysis of RC4. Then, we show that optimized strategies can break WEP based on 4000 packets by assuming that the first bytes of plaintext are known for each packet. We describe similar attacks for WPA. Firstly, we describe a distinguisher for WPA of complexity 2^{43} and advantage 0.5 which uses 2^{40} packets. Then, based on several partial temporary key recovery attacks, we recover the full 128-bit temporary key by using 2^{38} packets. It works within a complexity of 2^{96}. So far, this is the best attack against WPA. We believe that our analysis brings further insights on the security of RC4.

1 Introduction

RC4 was designed by Rivest in 1987. It used to be a trade secret until it was anonymously posted in 1994. Nowadays, RC4 is widely used in SSL/TLS and Wi-Fi 802.11 wireless communications. 802.11 [1] used to be protected by WEP (Wired Equivalent Privacy) which is now being replaced by WPA (Wi-Fi Protected Access) due to security weaknesses.

WEP uses RC4 with a pre-shared key. Each packet is encrypted by a XOR to a keystream generated by RC4. The RC4 key is the pre-shared key prepended with a 3-byte nonce IV. The IV is sent in clear for self-synchronization. There have been several attempts to break the full RC4 algorithm but it has only been devastating so far in this scenario. Indeed, the adversary knows that the key is constant except the IV, which is known. An active adversary can even alter the IV. Nowadays, WEP is considered as being terribly weak since passive attacks can recover the full key easily by assuming that the first bytes of every plaintext frames are known. This happens to be the case due to protocol specifications.

In order to fix this problem, the Wi-Fi Alliance has replaced WEP by WPA [1]. Peer authentication is based on IEEE 802.1X which accommodates a simple authentication mode based on a pre-shared key (WPA-PSK). Authentication creates a Temporary Key (TK). The TK then goes through the temporary key integrity protocol (TKIP) to derive per-packet keys (PPK). The idea is that TK is derived into a TTAK key to be used for a number of frames limited to 2^{16}. Each frame applies a simple transformation to TTAK and a counter TSC to derive the RC4 per-packet key PPK. Again, the 3 first bytes of the RC4 key are known (they actually depend on the counter).

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 343–363, 2011.

In addition to the key derivation, WPA provides a packet integrity protection scheme which prevents from replaying or altering the IV. Thus, only passive key recovery attacks can be considered.

Our contribution. In this paper, we construct tools for manipulating pools of biases. With our theory, we then analyze several statistical strategies for partial key recovery. We apply it to recover the 8 weak bits of the WPA key TK by using 2^{38} to 2^{40} packets. Incidentally, we apply our analysis to WEP and show that the best attacks so far can still be improved. We then transform our partial key recovery attack into a practical distinguisher for WPA. Finally, we build a full session key recovery with complexity 2^{96} and 2^{38} packets.

Related Work. We mention three approaches for the cryptanalysis of RC4: attacks based on the weaknesses of the Key Scheduling Algorithm (KSA) and attacks based on the weaknesses of the Pseudorandom Generator Algorithm (PRGA), and blackbox analysis.

As for the KSA, one of the first weaknesses published on RC4 was discovered by Roos [32] in 1995. This correlation binds the secret key bytes to the initial state S'_0. Roos [32] and Wagner [38] identified classes of weak keys which reveal the secret key if the first key bytes are known. This property has been largely exploited to break WEP (see [5,9,13,18,19,33,34,35,37]). Another class of result concerns the inversion problem of KSA: given the final state of the KSA, the problem is to recover the secret key [4,28].

Regarding PRGA, the analysis has been largely motivated by distinguishing attacks [8,11,22,24] or initial state reconstruction from the keystream bytes [10,17,25,36] with complexity of 2^{241} for the best state recovery attack. Relevant studies of the PRGA reveal biases in the keystream output bytes in [23,29]. Mironov recommends in [26] that the first 512 initial keystream bytes must be discarded to avoid these weaknesses.

Jenkins published in 1996 on his website [14] two biases in the PRGA of RC4. These biases have been generalized by Mantin in his Master Thesis [21]. Paul, Rathi and Maitra [30] discovered in 2008 a biased output index of the first keystream word generated by the PRGA. Another bias on the PRGA has been experimentally discovered by Maitra and Paul [20]. Finally, Sepehrdad, Vaudenay and Vuagnoux [33] discovered 48 new correlations in PRGA and 9 new correlations between the key bits and the key stream. This led to the fastest attack on WEP at the moment.

In practice, key recovery attacks on RC4 must bind KSA and PRGA weaknesses to correlate secret key words to keystream words. Some biases on the PRGA [16,30,20] have been successfully bound to the Roos correlation [32] to provide known plaintext attacks. Another approach is blackbox analysis, which does not require any binding. This was exploited in [33].

In 2004, Moen, Raddum and Hole [27] discovered that the recovery of at least two RC4 packet keys in WPA leads to a full recovery of the temporal key and the message integrity check key. Once from the same segment of 2^{16} consecutive packets, two keys are successfully recovered, the Moen, Raddum and Hole attack can be applied. This leads to a TK key recovery attack on WPA with complexity 2^{103} using 2 packets. Almost all known and new key recovery attacks on WEP could have been applied to WPA if there were several packets using the same RC4 key. Indeed, only the Fluhrer, Mantin and Shamir attack [9] is filtered. However, WPA uses a different secret key for every encrypted packet. In 2009, Tews and Beck [34] found a practical attack on WPA-PSK

to inject data in encrypted communication. Note that this attack does not recover the encryption key and need some additional quality of services features (described by IEEE 802.11e) which are not activated by default.

Structure of the paper. We first present in Section 2 RC4, WEP, and WPA, known biases in RC4 and some tools to be able to manipulate a pool of biases for target key bytes. Then, we study key recovery attacks to be able to recover some "weak bits" of the temporary key in Section 3. We show applications to WEP in Section 4, then present a distinguisher for WPA and a full temporary key recovery for WPA in Section 5.

2 Preliminaries

2.1 Description of RC4 and Notations

The stream cipher RC4 consists of two algorithms: the Key Scheduling Algorithm (KSA) and the Pseudo Random Generator Algorithm (PRGA). The RC4 engine has a state defined by two registers (words) i and j and one array (of N words) S defining a permutation over $\mathbf{Z}_N$. The KSA generates an initial state for the PRGA from a random key K of L words as described on Fig. 1. It starts with an array $\{0,1,\ldots,N-1\}$, where $N=2^8$ and swaps N pairs, depending on the value of the secret key K. At the end, we obtain the initial state $S'_0 = S_{N-1}$.

Once the initial state S'_0 is created, it is used by the second algorithm of RC4, the PRGA. Its role is to generate a keystream of words of $\log_2 N$ bits, which will be XORed with the plaintext to obtain the ciphertext. Thus, RC4 computes the loop of the PRGA each time a new keystream word z_i is needed, according to the algorithm on Fig. 1. Note that each time a word of the keystream is generated the internal state of RC4 is updated.

Notations. In this paper, we define all the operators such as addition, subtraction and multiplication in the group $\mathbf{Z}_N$ where $N=256$ (i.e. *words* are *bytes*). Thus, $x+y$ should be read as $(x+y) \bmod N$.

Let $S_i[k]$ (resp. $S'_i[k]$) denote the value of the permutation defined by array S at index k, after round i in KSA (resp. PRGA). We also denote $S_{N-1} = S'_0$. Let j_i (resp. j'_i) be

KSA

1: **for** $i = 0$ to $N-1$ **do**
2: $\quad S[i] \leftarrow i$
3: **end for**
4: $j \leftarrow 0$
5: **for** $i = 0$ to $N-1$ **do**
6: $\quad j \leftarrow j + S[i] + K[i \bmod L]$
7: $\quad \mathsf{swap}(S[i], S[j])$
8: **end for**

PRGA

1: $i \leftarrow 0$
2: $j \leftarrow 0$
3: **loop**
4: $\quad i \leftarrow i+1$
5: $\quad j \leftarrow j + S[i]$
6: $\quad \mathsf{swap}(S[i], S[j])$
7: $\quad$ output $z_i = S[S[i] + S[j]]$
8: **end loop**

Fig. 1. RC4 KSA and PRGA Algorithms

the value of j after round i of KSA (resp. PRGA) where the rounds are indexed with respect to i. Thus, the KSA has rounds $0, 1, \ldots, N-1$ and the PRGA has rounds $1, 2, \ldots$. The KSA and PRGA are defined by

KSA

$$\begin{aligned} j_{-1} &= 0 \\ j_i &= j_{i-1} + S_{i-1}[i] + K[i \bmod L] \\ S_{-1}[k] &= k \\ S_i[k] &= \begin{cases} S_{i-1}[j_i] & \text{if } k = i \\ S_{i-1}[i] & \text{if } k = j_i \\ S_{i-1}[k] & \text{otherwise} \end{cases} \end{aligned}$$

PRGA

$$\begin{aligned} j'_0 &= 0 \\ j'_i &= j'_{i-1} + S'_{i-1}[i] \\ S'_0[k] &= S_{N-1}[k] \\ S'_i[k] &= \begin{cases} S'_{i-1}[j'_i] & \text{if } k = i \\ S'_{i-1}[i] & \text{if } k = j'_i \\ S'_{i-1}[k] & \text{otherwise} \end{cases} \\ z_i &= S'_i[S'_i[i] + S'_i[j'_i]] \end{aligned}$$

2.2 Description of WEP

WEP [2] uses a 3-byte IV concatenated to a secret key of 40 or 104 bits (5 or 13 bytes) as an RC4 key. Thus, the RC4 key size is either 64 or 128 bits. In this paper, we do not consider the 40-bit key variant. So, $L = 16$. We have

$$K = K[0]\|K[1]\|K[2]\|K[3]\|\cdots\|K[15] = \mathsf{IV}_0\|\mathsf{IV}_1\|\mathsf{IV}_2\|K[3]\|\cdots\|K[15]$$

where IV_i represents the $(i+1)$th byte of the IV and $K[3]\|...\|K[15]$ the fixed secret part of the key. In theory, the value of the IV should be random but in practice, it is a counter, mostly in little-endian, and incremented by one each time a new 802.11b frame is encrypted. Sometimes, some particular values of IV are skipped to thwart specific attacks based on "weak IVs". Thus, each packet uses a slightly different key. RC4 then produces a keystream which is XORed to the plaintext to obtain the ciphertext.

It is well known [31] that a relevant portion of the plaintext is practically constant and that some other bytes can be predicted. They correspond to the LLC header and the SNAP header and some bytes of the TCP/IP encapsulated frame. For example, by XORing the first byte of the ciphertext with the constant value 0xAA, we obtain the first byte of the keystream. Thus, even if these attacks are called known plaintext attacks, they are ciphertext only in practice.

2.3 Description of WPA

WPA includes a key hash function [12] to defend against the Fluhrer-Mantin-Shamir attack [9], a Message Integrity Code (MIC) [7] and a key management scheme based on 802.1X [3] to avoid key reuse and to ease the key distribution.

The 128-bit Temporal Key (TK) is a per-session key. It is derived from the key management scheme during the authentication and is given as an input to the phase1 key hash function (key mixing algorithm) together with the 48-bit Transmitter Address (TA) and a 48-bit TKIP Sequence Counter (TSC) which is sometimes called IV. We will avoid this latter name to avoid confusion with the first 3 bytes of the RC4 key (which indeed only depend on TSC but are not equal).

TK can be used to encrypt up to 2^{48} packets. Every packet has a 48-bit index TSC which is split into IV32 and IV16. The IV32 counter is incremented every 2^{16} packets.

The packet is encrypted using a 128-bit RC4KEY which is derived from TK, TSC, and some other parameters (e.g. device addresses) which can be assumed constant and known by the adversary for our purpose. As for WEP, the first three bytes of RC4KEY only depend on TSC so they are not secret. The derivation works in two phases. The first phase does not depend on IV16 and is done once every 2^{16} packets for efficiency reasons. It derives a 80-bit key TTAK, called TKIP-mixed Transmit Address and Key (TTAK) in the standard (but denoted P1K in the reference code).

$$\mathsf{TTAK} = \mathsf{phase1}(\mathsf{TK}, \mathsf{TA}, \mathsf{IV32})$$

The second phase uses TK and IV16 to derive a 96-bit key PPK which is then turned into RC4KEY:

$$\mathsf{RC4KEY} = \mathsf{phase2}(\mathsf{TK}, \mathsf{TTAK}, \mathsf{IV16})$$

The key derivation of WPA based on a pre-shared key is depicted on Fig. 2 (without protocol parameters such as TA).

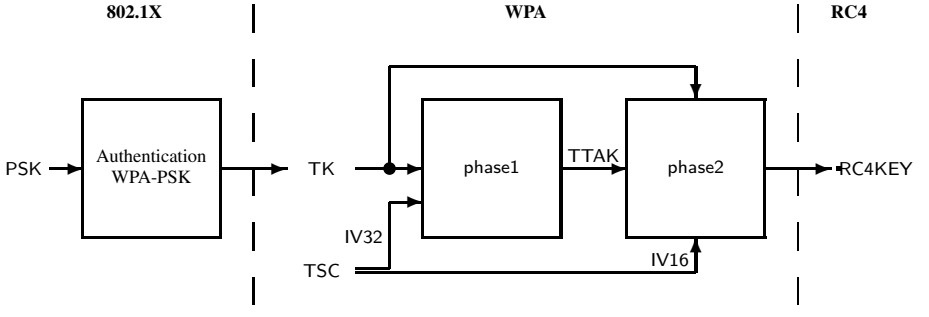


Fig. 2. WPA Key Derivation based on Pre-Shared Key Authentication Method

The RC4KEY is simply defined from PPK, TK, and IV16 by

$$\begin{array}{ll}
\mathsf{RC4KEY}[0] = \mathsf{high8}(\mathsf{IV16}) & \mathsf{RC4KEY}[1] = (\mathsf{high8}(\mathsf{IV16}) \text{ or } 0\mathsf{x}20) \text{ and } 0\mathsf{x}7\mathsf{f} \\
\mathsf{RC4KEY}[2] = \mathsf{low8}(\mathsf{IV16}) & \mathsf{RC4KEY}[3] = \mathsf{low8}((\mathsf{PPK}[5] \oplus (\mathsf{TK}[1] \| \mathsf{TK}[0])) \gg 1) \\
\mathsf{RC4KEY}[4] = \mathsf{low8}(\mathsf{PPK}[0]) & \mathsf{RC4KEY}[5] = \mathsf{high8}(\mathsf{PPK}[0]) \\
\mathsf{RC4KEY}[6] = \mathsf{low8}(\mathsf{PPK}[1]) & \mathsf{RC4KEY}[7] = \mathsf{high8}(\mathsf{PPK}[1]) \\
\vdots & \vdots
\end{array}$$

In what follows, we denote $K[i] = \mathsf{RC4KEY}[i \bmod 16]$ and $\mathsf{IV} = K[0] \| K[1] \| K[2]$ to use the same notations as in WEP. By convention, TTAK and PPK are considered as vectors or 16-bit words. TK and RC4KEY are considered as vectors or 8-bit words. Vectors are numbered starting from 0.

Note that a filter avoids the use of some weak IV classes. Actually, only the weak IV class discovered by Fluhrer, Mantin, and Shamir [9].

2.4 Biases in RC4

Throughout this paper, we denote $\bar{K}[i] = K[0] + \cdots + K[i]$. We let z denote the keystream derived from K using RC4. The first bytes of a plaintext frame are often known (see [37]), as well as IV, the first 3 bytes of K. That is, we assume that the adversary can use z and IV in a known plaintext attack.

We let I_0 be a set of integers, which represent the key byte indices which are already known. We call an I_0-clue a value clue for all $\bar{K}$ bytes whose index are in I_0. To begin with, we have $I_0 = \{0,1,2\}$ and $\mathsf{clue} = \mathsf{IV}$.

Given a set of indices I_0 and an index i, we assume that we have a list $\mathsf{row}^{\mathsf{RC4}}_{i|I_0}$ of pairs $(\bar{f}_j, p_j)$ in which $\bar{f}_j$ is a function such that for any I_0-clue clue, we have

$$\Pr\left[\bar{K}[i] = \bar{f}_j(z, \mathsf{clue})\right] = p_j$$

For simplicity, we assume that for i, z, and clue given, all $\bar{f}_j(z, \mathsf{clue})$ are pairwise different. We further assume that the events $\bar{K}[i] = \bar{f}_j(z, \mathsf{clue})$ with different i's are independent. We will also assume that $\bar{f}_j$ is of form $\bar{f}_j(z, \mathsf{clue}) = f_j(h(z, \mathsf{clue}))$ where $\mu = h(z, \mathsf{clue})$ lies in a domain of size N_μ. h is just a function compressing data to the minimum necessary.

We use a list of classes of biases from Table 1 (see [33]). More specifically, we use the rows $\mathsf{row}^{\mathsf{RC4}}_{i|I_0}$ in Table 2 taken from [33]. This table applies to RC4 in general but can be transformed for the WEP or WPA context due to $L = 16$. Indeed, we have $\bar{K}[i+16j] = \bar{K}[i] + j\bar{K}[15]$ for $0 \le i \le 15$ and $j = 0,1,2$. We define $\mathsf{deduce}(I)$ to be the set of all i's such that we can compute $\bar{K}[i]$ using this property, based on the values of $\bar{K}$ with indices in I. For instance, $\mathsf{deduce}(0,1,2,5) = \{0,1,2,5\}$ and $\mathsf{deduce}(0,1,2,5,15) = \{0,1,2,5,15,16,17,18,21,31,32,33,34,37,\ldots\}$. Next, we transform the above table by removing some rows for keys which can be deduced and by merging rows leading to the same key byte. Namely, we define $\mathsf{row}_{i|I_0}$ as follows: if $i \in \mathsf{deduce}(I_0)$, the row has a single "bias" $\bar{f}_1(z, \mathsf{clue}) = \bar{K}[i]$ with probability $p_1 = 1$ since $\bar{K}[i]$ can be computed

Table 1. Classes of Biases in RC4

I_0 is the set of $\bar{K}$ indices which are already known, P_{MP}, P_{KI}, P_{008}, and P_{009} are defined in Appendix, t is the largest integer such that $0,1,\ldots,t \in I_0$, and

$$\sigma_i = \sum_{j=0}^{t} S_{j-1}[j] + \sum_{j=t+1}^{i} S_t[j]$$

(e.g. $\sigma_i = \frac{i(i+1)}{2}$ and $t = -1$ when $0 \notin I_0$).

row	reference	$\bar{f}$	p	comment
$i \neq 1$	$\mathsf{MaitraPaul}(i, I_0)$	$\bar{K}[i] = z_{i+1} - \sigma_i$	$P_{\mathsf{MP}}(i,t)$	see [20]
i	$\mathsf{KleinImproved}(i, I_0)$	$\bar{K}[i] = -z_i + i - \sigma_i$	$P_{\mathsf{KI}}(i,t)$	see [37]
1	SVV_bb_000	$\bar{K}[1] = z_1 - 1$	$1.04237/N$	see [33]
2	SVV_bb_003	$\bar{K}[2] = z_2 - 3$	$0.65300/N$	see [33]
$i = 16i'$	$\mathsf{SVV\_008}(i, I_0)$	$\bar{K}[i] = z_i - i - \sigma_i$	$P_{008}(i,t)$	see [33]
$i = 16i'$	$\mathsf{SVV\_009}(i, I_0)$	$\bar{K}[i] = -z_i - i - \sigma_i$	$P_{009}(i,t)$	see [33]

Table 2. Table of Unconditional Biases in RC4 from known Key Bytes I_0

i	biases
0	$\mathsf{MaitraPaul}(i,I_0)$
1	$\mathsf{KleinImproved}(i,I_0)$ SVV_bb_000
2	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$ SVV_bb_003
3	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$
⋮	⋮ ⋮
15	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$
16	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$ $\mathsf{SVV\_008}(i,I_0)$ $\mathsf{SVV\_009}(i,I_0)$
17	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$
⋮	⋮ ⋮
31	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{MaitraPaul}(i,I_0)$
32	$\mathsf{KleinImproved}(i,I_0)$ $\mathsf{SVV\_008}(i,I_0)$ $\mathsf{SVV\_009}(i,I_0)$
33	$\mathsf{KleinImproved}(i,I_0)$
⋮	⋮
47	$\mathsf{KleinImproved}(i,I_0)$

from clue. Otherwise, the row is the concatenation of all $\mathsf{row}^{\mathsf{RC4}}_{i'|I_0}$ for i' in $\mathsf{deduce}(I_0 \cup \{i\}) - \mathsf{deduce}(I_0)$. For instance, $\mathsf{row}_{2|\{0,1,2\}}$ has a single bias, $\mathsf{row}_{5|\{0,1,2\}} = \mathsf{row}^{\mathsf{RC4}}_{5|\{0,1,2\}}$, and

$$\mathsf{row}_{5|\{0,1,2,15\}} = \mathsf{row}^{\mathsf{RC4}}_{5|\{0,1,2,15\}} \| \mathsf{row}^{\mathsf{RC4}}_{21|\{0,1,2,15\}} \| \mathsf{row}^{\mathsf{RC4}}_{37|\{0,1,2,15\}}$$

Given two lists of byte indices I_0 and $I = (i_1, \ldots, i_{\#I})$, we construct a new table $\Pi(I|I_0)$ in which the list of rows is $\mathsf{row}_{i_1|I_0}$, $\mathsf{row}_{i_2|I_0,i_1}$, ..., $\mathsf{row}_{i_{\#I}|I_0,i_1,i_2,\ldots,i_{\#I-1}}$. For instance, $I_0 = \{0,1,2\}$ and I is a list of key byte indices which are sequentially obtained using biases. We assume that I_0 is a minimal set in the sense that there is no strictly smaller set with same $\mathsf{deduce}(I_0)$. We further assume that I is a minimal set in the sense that there is no strictly smaller set with same $I \cap I_0$ and $\mathsf{deduce}(I \cup I_0)$. For instance, $I = (2,3,13,14,15)$ is minimal for $I_0 = \{0,1,2\}$, but $I = (2,3,13,14,15,16)$ is not. We define $\nu = (\bar{K}[i])_{i \in I}$ which belongs to a set of size $N_\nu(I) = N^{\#I}$. Given $i \in I$, we let $d_i^{\Pi(I|I_0)}$ be the length of row for $\bar{K}[i]$ in $\Pi(I|I_0)$. Given a tuple $(j_i)_{i \in I}$ such that $1 \le j_i \le d_i^{\Pi(I|I_0)}$ for all $i \in I$, by collecting together the j_ith bias of row i, we obtain an agglomerated bias to compute ν from z and an I_0-clue clue. Note that for technical reasons, we may have to keep elements of I_0 in I. This is why we may have rows for $i \in I_0$ in $\Pi(I|I_0)$ with a single bias of probability 1. We let

$$k(I|I_0) = \prod_{i \in I} d_i^{\Pi(I|I_0)}$$

be the number of possible agglomerated biases. For convenience, we number the agglomerated biases with an index ℓ from 1 to $k(I|I_0)$ and each number defines a tuple $(j_i)_{i \in I}$. So, the ℓth bias is defined by $\nu = f_\ell(z, \mathsf{clue})$ with probability

$$p_\ell^{\Pi(I|I_0)} = \prod_{i \in I} p_{i,j_i}^{\Pi(I|I_0)}$$

where $p_{i,j}^{\Pi(I|I_0)}$ is the probability of the jth bias in the row corresponding to $\bar{K}[i]$ in $\Pi(I|I_0)$.

We let $N_\mu(\Pi(I))$ be N raised to the power of the number of z_i bytes and I_0 bytes appearing in any of the biased equations from $\Pi(I)$. E.g., $N_\mu(\Pi(3,13,14|0,1,2)) = N^8$ since biases for $\bar{K}[3]$ are based on z_3 and z_4, and biases for $\bar{K}[13]$ and $\bar{K}[14]$ are based on z_{13}, z_{14}, and z_{15}. We further need IV to compute S_t. So, we have 8 bytes in total: z_i for $i \in \{3,4,13,14,15\}$ and IV. Given a key stream z, we define $\mu = h^{\Pi(I)}(z,\mathsf{clue})$ as the vector of all z_i and clue bytes which are useful. We define $\nu = f_\ell^{\Pi(I)}(\mu)$.

For simplicity, we write Π, k, N_ν, N_μ, p_ℓ, h, and f_ℓ when I and I_0 will be made clear from context. That is, the range of h has size N_μ, and f_ℓ goes from a domain of N_μ elements to a range of N_ν elements.

In the following, we use

$$s_e = \sum_{\ell=1}^{k} p_\ell^e = \sum_{\substack{(j_i)_{i\in I} \\ 1\le j_i\le d_i}} \prod_{i\in I} p_{i,j_i}^e = \prod_{i\in I} \sum_{j=1}^{d_i} p_{i,j}^e$$

for an integer e, and

$$\varepsilon_e = \left(\sum_{\ell=1}^{k} \left(p_\ell - \frac{1}{N_\nu}\right)^e\right)^{\frac{1}{e}} = \left(\sum_{i=0}^{e} \binom{e}{i} s_{e-i}(-N_\nu)^{-i}\right)^{\frac{1}{e}}$$

ε_e is called the *cumulated bias of order e*. The table below gives a few examples of cumulated biases.

I_0	I	N_ν	k	N_μ	ε_1	ε_2	ε_4
$\{0,1,2\}$	$(3,13,14)$	2^{24}	2^3	N^8	$2^{-21.37}$	$2^{-22.79}$	$2^{-23.40}$
$\{0,1,2\}$	$(15,3,14)$	2^{24}	$2^{8.81}$	N^{20}	$2^{-16.60}$	$2^{-20.79}$	$2^{-22.69}$
$\{0,1,2\}$	$(15,3,13,14)$	2^{32}	$2^{11.13}$	N^{23}	$2^{-21.82}$	$2^{-27.19}$	$2^{-29.69}$

2.5 Conditional Biases in RC4

We extend the notion of bias to the notion of conditional bias. We now assume that for each i we have d_i functions $\bar{f}_{i,j}$ and corresponding predicates $\bar{g}_{i,j}$ such that

$$\Pr\left[\bar{K}[i] = \bar{f}_j(z,\mathsf{clue}) | \bar{g}_j(z,\mathsf{clue})\right] = p_j$$

for some probability $p_j \neq \frac{1}{N}$. We further define

$$\Pr[\bar{g}_j(z,\mathsf{clue})] = q_j$$

and call q_j the *density* of the bias. For simplicity, we assume that for some given i, z, and clue, all suggested $\bar{f}_j(z,\mathsf{clue})$ when $\bar{g}_j(z,\mathsf{clue})$ holds, are pairwise distinct. We further assume that the events $\bar{K}[i] = \bar{f}_j(z,\mathsf{clue})$ with different i's are independent. We will also assume that $\bar{f}_j$ and $\bar{g}_j$ are of form $\bar{f}_j(z,\mathsf{clue}) = f_j(h(z,\mathsf{clue}))$ and $\bar{g}_j(z,\mathsf{clue}) = g_j(h(z,\mathsf{clue}))$ where $\mu = h(z,\mathsf{clue})$ lies in a domain of size N_μ.

We use the conditional biases in Table 5. All of them except SVV_db were taken from Korek [18] (they can be extracted from Aircrack, see [6,37]). We used some new formulas to compute their probabilities which are given in Appendix.

Given two minimal sets of byte indices I_0 and I as in the previous section, we also make a table $\Pi(I|I_0)$ and collect a list of ℓ agglomerated biases in which probabilities and densities are multiplied. We define

$$\bar{s}_e = \sum_{\ell=1}^{k} q_\ell p_\ell^e \quad , \quad \bar{s}_e^{(N_x)} = \sum_{\ell=1}^{k} \frac{q_\ell}{1-\frac{q_\ell}{N_x}} p_\ell^e$$

and

$$\bar{\varepsilon}_e = \sqrt[e]{\sum_{\ell=1}^{k} q_\ell \left(p_\ell - \frac{1}{N_v}\right)^e} = \sqrt[e]{\sum_{i=0}^{e} \binom{e}{i} \bar{s}_{e-i}(-N_v)^{-i}}$$

$$\bar{\varepsilon}_e^{(N_x)} = \sqrt[e]{\sum_{\ell=1}^{k} \frac{q_\ell}{1-\frac{q_\ell}{N_x}} \left(p_\ell - \frac{1}{N_v}\right)^e} = \sqrt[e]{\sum_{i=0}^{e} \binom{e}{i} \bar{s}_{e-i}^{(N_x)}(-N_v)^{-i}}$$

$\bar{s}_e^{(N_x)}$ resp. $\bar{\varepsilon}_e^{(N_x)}$ is the regular $\bar{s}_e$ resp. $\bar{\varepsilon}_e$ with a special correcting factor depending on some value N_x. This correction may look arbitrary. It will appear in the analysis of Section 3. The $\bar{s}$ values can be computed easily by

$$\bar{s}_e = \sum_{(j_i)_{i\in I}} \prod_{i\in I} q_{i,j_i} p_{i,j_i}^e = \prod_{i\in I} \sum_{j=1}^{d_i} q_{i,j} p_{i,j}^e$$

In the sequel, when $q_\ell \neq 1$ we assume $q_\ell \ll 1$ to approximate $\frac{1}{1-\frac{q_\ell}{N_x}} \approx 1 + \frac{1}{N_x-1} 1_{q_\ell=1}$. So, we compute $\bar{s}_e^{(N_x)}$ like for $\bar{s}_e$ but add a fraction of the regular s_e term for unconditional biases.

$$\bar{s}_e^{(N_x)} = \sum_{(j_i)_{i\in I}} \prod_{i\in I} \frac{q_{i,j_i}}{1-\frac{q_{i,j_i}}{N_x}} p_{i,j_i}^e \approx \frac{s_e}{N_x - 1} + \prod_{i\in I} \sum_{j=1}^{d_i} q_{i,j} p_{i,j}^e$$

The approximation is very useful to estimate $\bar{s}_e^{(N_x)}$ with low complexity. Namely, we can compute all useful $\bar{\varepsilon}_e$'s in time $O(ed)$ where d is the total number of biases, although the number of agglomerated biases k is of order $d^{\#I}$ which can be very large.

2.6 More Definitions

We denote

$$\varphi(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\lambda} e^{-\frac{x^2}{2}} dx = \frac{1}{2} \mathrm{erfc}\left(-\frac{\lambda}{\sqrt{2}}\right)$$

In particular, $\varphi(-\lambda/\sqrt{2}) = \frac{1}{2}\mathrm{erfc}(\frac{\lambda}{2})$.

3 Attacking Weak Bits Based on Biases

There are 8 bits of TK that we call *weak* because they have a simple relation with bits in PPK. These bits consists of the 7 most significant bits of $\mathsf{TK}[0]$ and the least significant bit of $\mathsf{TK}[1]$. We will define statistical attacks using the following mappings:

$$z^m, \mathsf{IV}^m \xrightarrow{h} \mu \xrightarrow[\text{if } g_\ell(\mu)]{f_\ell} \nu \xrightarrow{\pi} x$$

Here, z^m is the mth keystream using IV^m, μ is a compressed information to compute ν, some RC4 key bytes which are used to compute x, some information about TK which we want to recover using statistics. We define N_x the number of possible values for x.

3.1 First Attack: Recovering some Weak Bits of TK

We use $I_0 = \{0,1,2\}$ and $I = (2,3,13,14)$. Given $\bar{K}[2]$, $\bar{K}[3]$, $\bar{K}[13]$, $\bar{K}[14]$, the adversary can compute $K[3] = \bar{K}[3] - \bar{K}[2]$ and $K[14] = \bar{K}[14] - \bar{K}[13]$. We have

$$\begin{aligned} \mathsf{PPK}[5] &= K[15] \| K[14] \\ K[3] &= \mathsf{low8}((\mathsf{PPK}[5] \oplus (\mathsf{TK}[1] \| \mathsf{TK}[0])) \gg 1) \end{aligned}$$

So, given $\nu = (\bar{K}[2], \bar{K}[3], \bar{K}[13], \bar{K}[14])$ the adversary can compute $x = \mathsf{high7}(\mathsf{TK}[0])$ by

$$\pi(\nu) = \mathsf{low7}((\bar{K}[3] - \bar{K}[2]) \oplus ((\bar{K}[14] - \bar{K}[13]) \gg 1))$$

$N_\nu = 2^{32}$ is the total number of possible ν's and $N_x = 2^7$ is the total number of possible x's. We have $N_\mu = 2^{48}$, the total number of $\mu = h(z, \mathsf{IV})$.

We can recover the 7 weak bits as follows: for each candidate value x, each packet m, and each $\ell = 1, \ldots, k$ (corresponding to a tuple $(j_2, j_3, j_{13}, j_{14})$), if agglomerated condition $g_\ell(h(z^m, \mathsf{IV}^m))$ holds, we define $\nu = f_\ell(h(z^m, \mathsf{IV}^m))$ the value of RC4 key bytes suggested by bias ℓ on packet m, which is correct with probability p_ℓ. We let $x = \pi(\nu)$ the suggested value of x computed as explained. We let $X_{x,m,\ell}$ be some magic coefficient a_ℓ (to be optimized later) if $\pi(f_\ell(h(z^m, \mathsf{IV}^m))) = x$ and 0 otherwise. We let $Y_x = \sum_{m=1}^{n} \sum_{\ell=1}^{k} X_{x,m,\ell}$ where n is the total number of packets to be used. Clearly, the correct value for ν is suggested with probability p_ℓ and others are obtained randomly. We assume incorrect ones are suggested with the same probability $\frac{1-p_\ell}{N_\nu - 1}$.

If x is not the correct value, it is not suggested for sure when ν is correct. Since π is balanced, this incorrect x has $\frac{N_\nu}{N_x}$ values ν belonging to the set of $N_\nu - 1$ incorrect ones. So, x is suggested with with probability $\frac{N_\nu}{N_x} \times \frac{1-p_\ell}{N_\nu - 1}$. So, the $X_{x,m,\ell}$ for incorrect x's are random variables (r.v.) with expected values

$$a_\ell q_\ell N_\nu \frac{1 - p_\ell}{N_x (N_\nu - 1)}$$

if x is not the correct value.

If x is the correct value, it is suggested with probability p_ℓ for the correct ν and when ν is one of the $\frac{N_\nu - N_x}{N_x}$ (incorrect) preimages of x by π. That is, with overall probability

$p_\ell + \frac{N_\nu - N_x}{N_x} \times \frac{1-p_\ell}{N_\nu - 1}$. So, the $X_{x,m,\ell}$ for the correct x are r.v. with expected values

$$a_\ell q_\ell N_\nu \frac{1-p_\ell}{N_x(N_\nu - 1)} + a_\ell q_\ell \frac{N_\nu p_\ell - 1}{N_\nu - 1}$$

The difference between these two expected values is important but it is not the same for the variance. Since every x is suggested with probability roughly $\frac{q_\ell}{N_x}$, we assume that the variance of all $X_{x,m,\ell}$ can be approximated by $\frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right) a_\ell^2$. Let Δ be the operator making the difference between distributions for a good x and a bad one. We have

$$E(Y_{\mathsf{bad}}) = \frac{n}{N_x\left(1 - \frac{1}{N_\nu}\right)} \sum_\ell a_\ell q_\ell (1 - p_\ell)$$

$$\Delta E(Y) = \frac{n}{1 - \frac{1}{N_\nu}} \sum_\ell a_\ell q_\ell \left(p_\ell - \frac{1}{N_\nu}\right)$$

$$V(Y) \approx n \sum_\ell a_\ell^2 \frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right)$$

Where $E(Y_{\mathsf{bad}})$ denotes the expected value of an Y_x variable for any bad x. Here, we removed the subscript x of Y_x in $\Delta E(Y)$ and $V(Y)$ since these do not depend on a specific value for x. Let λ be such that $\Delta E(Y) = \lambda\sqrt{V(Y)}$. The probability that the correct Y_x is lower than any wrong Y_x is $\rho = \varphi\left(-\frac{\lambda}{\sqrt{2}}\right)$. That is, the expected number of wrong x's with larger Y_x is

$$r = (N_x - 1)\varphi\left(-\frac{\lambda}{\sqrt{2}}\right) \tag{1}$$

So,

$$n = \lambda^2 \left(1 - \frac{1}{N_\nu}\right)^2 \frac{\sum_\ell a_\ell^2 \frac{q_\ell}{N_x}\left(1 - \frac{q_\ell}{N_x}\right)}{\left(\sum_\ell a_\ell q_\ell \left(p_\ell - \frac{1}{N_\nu}\right)\right)^2}$$

By derivating both terms of the fraction with respect to a_ℓ and equaling them, we obtain that the optimal value is reached for

$$a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\nu}\right)$$

This leads us to

$$E(Y_{\mathsf{bad}}) = \frac{n}{N_y}\left(\bar{\varepsilon}_1^{(N_x)} - \frac{1}{1 - \frac{1}{N_\nu}}(\bar{\varepsilon}_2^{(N_x)})^2\right)$$

$$\Delta E(Y) = \frac{n}{1 - \frac{1}{N_\nu}}(\bar{\varepsilon}_2^{(N_x)})^2$$

$$V(Y) \approx \frac{n}{N_x}(\bar{\varepsilon}_2^{(N_x)})^2$$

$$n = \frac{\lambda^2}{N_x\left(\bar{\varepsilon}_2^{(N_x)}\right)^2}\left(1 - \frac{1}{N_\nu}\right)^2 \tag{2}$$

So we can see where the correction in $\bar{\varepsilon}_2^{(N_x)}$ appears.

The attack works as follows:

set $I = (2,3,13,14)$ and $I_0 = \{0,1,2\}$
initialize the Y_x counters to 0
for $m = 1$ to n **do**
  for $\ell = 1$ to k **do**
    if $g_\ell(h(z^m, \mathsf{IV}^m))$ holds **then**
      compute $\mathsf{v} = f_\ell(h(z^m, \mathsf{IV}^m))$, the suggested $(\bar{K}[2], \bar{K}[3], \bar{K}[13], \bar{K}[14])$
      compute $x = \pi(\mathsf{v})$
      increment Y_x by $a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\mathsf{v}}\right)$
    end if
  end for
end for
output $x = \arg\max_x Y_x$

Clearly, the time complexity is nk. The complexity is measured in terms of number of times the **if** structure is executed. This should have a complexity which is essentially equivalent to executing the phase2 key derivation. The memory complexity has the order of magnitude of N_x. Here is a variant:

set $I = (2,3,13,14)$ and $I_0 = \{0,1,2\}$
initialize a table y_x^μ to 0
for $\ell = 1$ to k **do**
  for all possible μ such that $g_\ell(\mu)$ holds **do**
    compute $x = \pi(f_\ell(\mu))$
    increment y_x^μ by $a_\ell = \frac{N_x}{N_x - q_\ell}\left(p_\ell - \frac{1}{N_\mathsf{v}}\right)$
  end for
end for
initialize the Y_x counters to 0
for $m = 1$ to n **do**
  for all x **do**
    compute $\mu = h(z^m, \mathsf{IV}^m)$
    increment Y_x by y_x^μ
  end for
end for
output $x = \arg\max_x Y_x$

Now, the time complexity is $N_\mu k + N_x n$ and the memory complexity is $N_\mu N_x$. So, let say that the complexity is

$$c = \min(nk, N_\mu k + N_x n) \tag{3}$$

The two complexity curves cross for $n = N_\mu \frac{k}{k - N_x} \approx N_\mu$ when $N_x \ll k$.

For $I = (2,3,13,14)$, we have $N_\mathsf{v} = 2^{32}$, $N_\mu = 2^{48}$, and $N_x = 2^7$. The complexities with and without using conditional biases are summarized in Table 3. As we can see, when ignoring the conditional biases, we need about 30% more packets but the complexity is much lower because k is smaller. So, conditional biases do not seem useful in this case.

3.2 Second Attack

Let $I_0 = \{0,1,2\}$, $I = (15,2,3,14)$, and $x = \mathsf{low1}(\mathsf{TK}[1])$ be the last weak bit. Given IV and $\mathsf{v} = (\bar{K}[2], \bar{K}[3], \bar{K}[14], \bar{K}[15])$, we deduce $x = \pi(\mathsf{v})$ by

$$\pi(\mathsf{v}) = \mathsf{high1}((\bar{K}[3] - \bar{K}[2]) \oplus (\bar{K}[15] - \bar{K}[14]))$$

So, we apply the first attack with this I and $N_x = 2$. Since $15 \in I$ we have more biases. We have r, n, and c from Eq. (1), Eq. (2) and Eq. (3).

For $I = (15,2,3,14)$, we have $N_{\mathsf{v}} = 2^{32}$, $N_\mu = 2^{48}$, and $N_x = 2$. The complexities are summarized in Table 3. Again, conditional biases are not very useful. We can also see that this choice of I leads to a much better attack than the one from Section 3.1 in terms of n but the complexity is slightly higher. This is due to a larger k.

3.3 Merging Attacks

Given two attacks with sets I^1 resp. I^2 for recovering independent x^1 resp. x^2 leading to characteristics Y_x^1 resp. Y_x^2, c^1 resp. c^2, n^1 resp. n^2, λ^1 resp. λ^2, one problem is to merge the sorted lists of x^1 and x^2. One can follow the approach by Junod-Vaudenay [15]. We sort pairs following their likelihood ratio, which is obtained by multiplying the likelihood ratio of both terms. We assume that all Y_x^i are independent, normally distributed with variance $V(Y^i)$, and expected value either $E(Y^i_{\mathsf{bad}})$ or $E(Y^i_{\mathsf{bad}}) + \Delta E(Y^i)$. Given x^i, the ratio for x^i being the correct value based on the observation $Y^i_{x^i}$ is

$$\frac{\Pr[Y^i_{x^i} | x^i \text{ good}]}{\Pr[Y^i_{x^i} | x^i \text{ wrong}]} = \frac{\frac{1}{\sqrt{2\pi V(Y^i)}} e^{-\frac{\left(Y^i_{x^i} - E(Y^i_{\mathsf{bad}}) - \Delta E(Y^i)\right)^2}{2V(Y^i)}}}{\frac{1}{\sqrt{2\pi V(Y^i)}} e^{-\frac{\left(Y^i_{x^i} - E(Y^i_{\mathsf{bad}})\right)^2}{2V(Y^i)}}}$$
$$= e^{Y^i_{x^i} \frac{\Delta E(Y^i)}{V(Y^i)} + \frac{\Delta E(Y^i)}{2V(Y^i)} (\Delta E(Y^i) - E(Y^i_{\mathsf{bad}}))}$$

So, when multiplying some terms of this form for pairs of values, sorting them by decreasing product is equivalent to sorting them by decreasing value of

$$Y_{x^1,x^2} = Y^1_{x^1} \frac{\Delta E(Y^1)}{V(Y^1)} + Y^2_{x^2} \frac{\Delta E(Y^2)}{V(Y^2)}$$

With same assumptions as in [15], we are back in the situation where Y_{x^1,x^2} is normally distributed. We have

$$\Delta E(Y) = V(Y) = \frac{(\Delta E(Y^1))^2}{V(Y^1)} + \frac{(\Delta E(Y^2))^2}{V(Y^2)} = (\lambda^1)^2 + (\lambda^2)^2$$

So, $\lambda = \sqrt{(\lambda^1)^2 + (\lambda^2)^2}$, and the average number of wrong (x^1, x^2) pair with higher score than the good one is $r = (N_x^1 N_x^2 - 1)\varphi(-\frac{\lambda}{\sqrt{2}})$. Overall, we can use

$$n = \frac{\lambda^2}{N_{x^1} \left(\frac{\bar{\varepsilon}_2^{\left(N_{x^1}\right)}(1)}{1 - \frac{1}{N_{\mathsf{v}}^1}} \right)^2 + N_{x^2} \left(\frac{\bar{\varepsilon}_2^{\left(N_{x^2}\right)}(2)}{1 - \frac{1}{N_{\mathsf{v}}^2}} \right)^2}$$

and $c = c^1 + c^2$ by using Eq. (3) for c^1 and c^2. We can use these merging rules to merge the two previous attacks. We obtain the results from Table 3.

Table 3 shows the complexity when merging the previous attacks to recover the 8 weak bits of TK. We compare it with the attack using a merged set I directly. As we can see, merging attacks with small I's is much better.

Table 3. Complexities of several attacks to recover $\log_2 N_x$ bits from TK. We compare them when including conditional biases and without. We provide the number of packets n, the running time complexity c, the expected number r of better wrong values, as well as parameters k, $\varepsilon = \bar{\varepsilon}_2^{(N_x)}$, λ, and N_v. Except when $N_x = 2$ for which it would not make any sense, we target $r = \frac{1}{2}$ (that is, the correct value has the higher score in half of the cases, roughly). We used $I_0 = \{0,1,2\}$.

	reference	I	n	c	r	N_x	k	ε	λ	N_v	N_u	cond. biases
1u	Section 3.1	$(2,3,13,14)$	$2^{40.13}$	$2^{43.13}$	$\frac{1}{2}$	2^7	$2^{3.00}$	$2^{-21.65}$	3.76	2^{32}	N^8	without
1c	Section 3.1	$(2,3,13,14)$	$2^{39.70}$	$2^{51.87}$	$\frac{1}{2}$	2^7	$2^{12.17}$	$2^{-21.44}$	3.76	2^{32}	N^{10}	with
2u	Section 3.2	$(15,2,3,14)$	$2^{36.10}$	$2^{44.91}$	$\frac{1}{4}$	2	$2^{8.81}$	$2^{-18.62}$	0.95	2^{32}	N^{20}	without
2c	Section 3.2	$(15,2,3,14)$	$2^{35.98}$	$2^{54.35}$	$\frac{1}{4}$	2	$2^{18.37}$	$2^{-18.56}$	0.95	2^{32}	N^{22}	with
3u	merge 1u+2u		$2^{39.33}$	$2^{48.17}$	$\frac{1}{2}$	2^8			4.08			without
3c	merge 1c+2c		$2^{39.05}$	$2^{57.43}$	$\frac{1}{2}$	2^8			4.08			with
4u		$(15,2,3,13,14)$	$2^{47.67}$	$2^{58.81}$	$\frac{1}{2}$	2^8	$2^{11.14}$	$2^{-25.81}$	4.08	2^{40}	N^{23}	without
4c		$(15,2,3,13,14)$	$2^{47.36}$	$2^{71.37}$	$\frac{1}{2}$	2^8	$2^{24.01}$	$2^{-25.65}$	4.08	2^{40}	N^{25}	with

We may think that we could get better results by using the entire vector Y instead of Y_x only to compute the likelihood ratio of x. By redoing the computations, we obtain

$$\frac{\Pr[Y|x^i \text{ good}]}{\Pr[Y|x^i \text{ wrong}]} = \frac{\Pr[Y|x^i \text{ good}]}{\frac{1}{N_x^i - 1}\sum_{x \neq x^i} \Pr[Y|x \text{ good}]} = \frac{N_x - 1}{\sum_{x' \neq x} e^{(Y_{x'} - Y_x)\frac{\Delta E(Y)}{V(Y)}}}$$

When x is good and x' is bad, the exponential in the sum is of order $e^{-\lambda}$. When x is bad and x' is good, it has order e^{λ}. When both are bad, it has order $e^{\pm\sqrt{\lambda}}$. So, we have to compare one ratio of order e^{λ} to others of order $\frac{N_x - 1}{e^{\lambda} + (N_x - 2)e^{\pm\sqrt{\lambda}}}$. We know that a wrong ratio is higher than the good one with probability $\varphi(-\lambda/\sqrt{2})$. When multiplying the independent likelihood ratios for x^1 and x^2, if we approximate $\sum_{x'_1 \neq x^1} F(x'_1) \sum_{x'_2 \neq x^2} G(x'_2) \approx \sum_{(x'_1, x'_2) \neq (x^1, x^2)} F(x'_1) G(x'_2)$, we obtain a likelihood ratio of same form based on Y_{x^1, x^2}. This validates the above rule of the thumb for sorting pairs following their Y_{x^1, x^2} score.

4 Attack on WEP

We apply the first attack with $x = v$: we only want to recover key bytes which are the same for all packets. This attack produces a ranking of possible x's in a form of a list $\mathcal{L}$ by decreasing order of likelihood.

We use the following attack:

1: compute the ranking $\mathcal{L}_{15}$ for $I=(15)$ and $I_0=\{0,1,2\}$
2: **for** each $\bar{k}_{15}$ in $\mathcal{L}_{15}$ **do**
3: run recursive attack on input $\bar{k}_{15}$
4: **end for**
5: stop: attack failed

recursive attack with input $(\bar{k}_{15},\bar{k}_3,\ldots,\bar{k}_{i-1})$**:**

6: **if** $i \le i_{\max}$ **then**
7: compute the ranking $\mathcal{L}_i$ for $I=(i)$ and $I_0=\{0,\ldots,i-1,15\}$
8: truncate $\mathcal{L}_i$ to its first ρ_i terms
9: **for** each $\bar{k}_i$ in $\mathcal{L}_i$ **do**
10: run recursive attack on input $(\bar{k}_{15},\bar{k}_3,\ldots,\bar{k}_{i-1},\bar{k}_i)$
11: **end for**
12: **else**
13: **for** each $\bar{k}_{i_{\max}+1},\ldots,\bar{k}_{14}$ **do**
14: test key $(\bar{k}_3,\ldots,\bar{k}_{14},\bar{k}_{15})$ and stop if correct
15: **end for**
16: **end if**

Let $\varepsilon_i = \bar{\varepsilon}_2^{(N_x)}(i|0,\ldots,i-1,15)$ for $i=3,\ldots,i_{\max}$ and $\varepsilon_{15}=\bar{\varepsilon}_2^{(N_x)}(15|0,1,2)$ be the ε used by the attack on $\bar{K}[i]$. Similarly, let $N_x=N_y=N$, and r_i, k_i, λ_i, c_i be their parameters following Eq. (1,2,3). Let R_i be the rank of the correct $\bar{k}_i$ value in L_i. We know that $E(R_i)=r_i$. We can easily see that $V(R_i)=r_i\left(1-\frac{r_i}{N_x-1}\right)$. By using the law of large numbers, the probability that R_i is lower than ρ_i is $u_i=\varphi\left(\frac{\rho_i-r_i}{\sqrt{r_i\left(1-\frac{r_i}{N_x-1}\right)}}\right)$ so the success probability is $\prod_{i=3}^{i_{\max}} u_i$ and the complexity is

$$c = c_{15} + r_{15}\left(c_3 + \rho_3\left(c_4 + \rho_4\left(\cdots c_{i_{\max}} + \rho_{i_{\max}} N^{14-i_{\max}}\cdots\right)\right)\right)$$

To approximate the optimal choice of ρ's, we set $\rho_i = r_i + \alpha\sqrt{r_i\left(1-\frac{r_i}{N_x-1}\right)}$ for some α. The success probability is $\varphi(\alpha)^{i_{\max}-2}$. We can adjust $\alpha=\varphi^{-1}(2^{-\frac{1}{i_{\max}-2}})$ so that this becomes 50% and we obtain c in terms of n. Computation shows that figures are better for $i_{\max}=14$. For this, we have $\alpha\approx 1.588$. We plotted $\log_2 c$ in terms of n on Fig. 3.

Note that this computation assumes real values for the ρ's. Since they must be integer, the real complexity may be slightly higher. For instance, with $n=4000$, the plotted complexity is $2^{24.02}$. With integral values, we can try with $\rho_i=5$ for $i=3,5,6$ and $\rho_i=4$ for $i=4,7,8,\ldots,14$. We obtain $c=2^{24.35}$ and a success rate of 51%.

Note that without the conditional biases, the same analysis with 4000 gives a complexity of 2^{66}. So, these biases make a huge difference in this case.

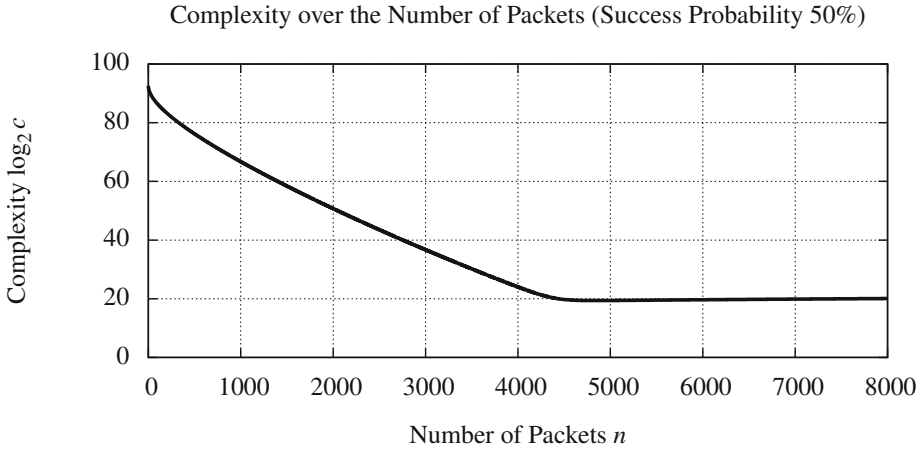


Fig. 3. Logarithmic complexity in terms of data complexity for breaking WEP

5 Attack on WPA

5.1 Distinguishing WPA

The first attack can be turned into a distinguisher as follows. The expected value and variance of the correct Y_x are roughly $E(Y_{\mathsf{bad}})+\lambda\sqrt{V(Y)}$ and $V(Y)$. The random variable Y_x is larger than $T=E(Y_{\mathsf{bad}})+\lambda'\sqrt{V(Y)}$ with probability $\varphi(\lambda-\lambda')$. Now, if we replace the WPA packets by some random sequences, the counters all have expected value $E(Y_{\mathsf{bad}})$ and variance approximately $V(Y)$. The probability that a given counter exceeds T is $\varphi(-\lambda')$. The probability that any counter exceeds this is lower than $N_x\varphi(-\lambda')$. So, the condition $\max_x Y_x > T$ makes a distinguisher of same n and c as in the first attack, and with advantage larger than $\varphi(\lambda-\lambda')-N_x\varphi(-\lambda')$. We find the optimal $\lambda'=\frac{\lambda}{2}+\frac{\ln N_x}{\lambda}$. So, $\mathsf{Adv}\geq\beta$ with

$$\beta=\varphi\left(\frac{\lambda}{2}-\frac{\ln N_x}{\lambda}\right)-N_x\varphi\left(-\frac{\lambda}{2}-\frac{\ln N_x}{\lambda}\right) \tag{4}$$

We use the same values as before and target $\mathsf{Adv}\geq\frac{1}{2}$. We use Eq. (2) for n, Eq. (3) for c, and Eq. (4) for a lower bound β of the advantage. The performances of the distinguishers are summarized on Table 4. Again, the attack based on $I=(15,2,3,14)$ is better in terms of number of packets but not in terms of complexity. It works using 2^{38} packets and complexity 2^{47}. The one based on $I=(2,3,13,14)$ works with 30% more packets (2^{40}) with no conditional biases but with a much better complexity 2^{43}.

5.2 Temporary Key Recovery

The results from [27] lead to an "easy" attack on WPA: guess the 96-bit PPK and the 8 weak bits of TK within an average complexity of 2^{103} until it generates the correct

Table 4. Complexities of several distinguishers for WPA. We compare them when including conditional biases and without. We provide the number of packets n, the running time complexity c, the bound on the advantage β, as well as parameters k, $\varepsilon = \bar{\varepsilon}_2^{(N_x)}$ or ε_2, λ, and N_v. We target $\beta = \frac{1}{2}$. We used $I_0 = \{0,1,2\}$.

	I	n	c	β	N_x	k	ε	λ	N_v	N_μ	cond. biases
1u	$I = (2,3,13,14)$	$2^{39.85}$	$2^{42.85}$	0.5	2^7	$2^{3.00}$	$2^{-21.65}$	3.41	2^{32}	N^8	without
1c	$I = (2,3,13,14)$	$2^{39.42}$	$2^{51.59}$	0.5	2^7	$2^{12.17}$	$2^{-21.44}$	3.41	2^{32}	N^{10}	with
2u	$I = (15,2,3,14)$	$2^{37.94}$	$2^{46.76}$	0.5	2	$2^{8.81}$	$2^{-18.62}$	1.81	2^{32}	N^{20}	without
2c	$I = (15,2,3,14)$	$2^{37.82}$	$2^{56.19}$	0.5	2	$2^{18.37}$	$2^{-18.56}$	1.81	2^{32}	N^{22}	with

keystream. Then, guess the 96-bit PPK of another packet in the same segment (with the weak bits already known). Then, apply the method of [27] to recover TK. We improve this attack by recovering the weak bits of TK separately: we know from Table 3 that we can recover the weak bits of TK by using 2^{38} packets. After having recovered the weak bits, we note that the 96-bit PPK is now enough to recalculate RC4KEY. So, we can do an exhaustive search on PPK for a given packet until we find the correct one generating the packet. This works with complexity 2^{95} on average. We do it twice to recover the PPK of two packets in the same segment. Given these two PPK sharing the same IV32, we recover TK by using the method of [27]. Therefore, we can recover the temporary key TK and decrypt all packets with complexity 2^{96}. The number of packets needed to recover the weak bits is 2^{38}.

6 Conclusion

We deployed a framework to handle pools of biases for RC4 which can be used to break WPA. In the case of the 8 weak bits of TK, we have shown a simple distinguisher and a partial key recovery attack working with 2^{38} packets and practical complexity. This can be used to improve the attack by Moen-Raddum-Hole [27] to mount a full temporary key recovery attack of complexity 2^{96} using 2^{38} packets. So far, this is the best temporal key recovery attack against WPA. In a future work we plan to study further key recovery attacks to recover more pieces of TK with complexity lower than 2^{96}.

We have shown that conditional biases are not very helpful for breaking WPA but they really are against WEP. Indeed, we recover keys with a success rate 50% by using 4 000 packets and a complexity of 2^{26}.

References

1. ANSI/IEEE standard 802.11i, Amendment 6 Wireless LAN Medium Access Control (MAC) and Physical Layer (phy) Specifications, Draft 3. IEEE (2003)
2. IEEE Std 802.11, Standards for Local and Metropolitan Area Networks: Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specifications (1999)
3. IEEE 802.1 WG. 802.1x: Standards for Local and Metropolitan Area Networks: Port-Based Access Control. IEEE (2001)

4. Biham, E., Carmeli, Y.: Efficient Reconstruction of RC4 Keys from Internal States. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 270–288. Springer, Heidelberg (2008)
5. Bittau, A.: Additional Weak IV Classes for the FMS Attack (2003), http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036
6. Chaabouni, R.: Breaking WEP Faster with Statistical Analysis. Semester project. In: EPFL/LASEC (2006)
7. Ferguson, N.: Michael: an Improved MIC for 802.11 WEP. IEEE doc. 802.11-2/020r0 (2002)
8. Fluhrer, S.R., McGrew, D.A.: Statistical Analysis of the Alleged RC4 Keystream Generator. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 19–30. Springer, Heidelberg (2001)
9. Fluhrer, S.R., Mantin, I., Shamir, A.: Weaknesses in the Key Scheduling Algorithm of RC4. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 1–24. Springer, Heidelberg (2001)
10. Golic, J.D.: Iterative Probabilistic Cryptanalysis of RC4 Keystream Generator. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 220–223. Springer, Heidelberg (2000)
11. Golić, J.D.: Linear Statistical Weakness of Alleged RC4 Keystream Generator. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 226–238. Springer, Heidelberg (1997)
12. Housley, R., Whiting, D., Ferguson, N.: Alternate Temporal Key Hash. IEEE doc. 802.11-02/282r2 (2002)
13. Hulton, D.: Practical Exploitation of RC4 Weaknesses in WEP Environments (2001), http://www.dachb0den.com/projects/bsd-airtools/wepexp.txt
14. Jenkins, R.: ISAAC and RC4 (1996), http://burtleburtle.net/bob/rand/isaac.html
15. Junod, P., Vaudenay, S.: Optimal Key Ranking Procedures in a Statistical Cryptanalysis. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 235–246. Springer, Heidelberg (2003)
16. Klein, A.: Attacks on the RC4 Stream Cipher. Design, Codes, and Cryptography 48, 269–286 (2008)
17. Knudsen, L.R., Meier, W., Preneel, B., Rijmen, V., Verdoolaege, S.: Analysis Methods for (Alleged) RC4. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 327–341. Springer, Heidelberg (1998)
18. Korek: Next Generation of WEP Attacks? (2004), http://www.netstumbler.org/showpost.php?p=93942&postcount=%35
19. Korek: Need Security Pointers (2004), http://www.netstumbler.org/showthread.php?postid=89036#pos%t89036
20. Maitra, S., Paul, G.: New Form of Permutation Bias and Secret Key Leakage in Keystream Bytes of RC4. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 253–269. Springer, Heidelberg (2008)
21. Mantin, I.: Analysis of the Stream Cipher RC4 (2001), http://www.wisdom.weizmann.ac.il/~itsik/RC4/rc4.html
22. Mantin, I.: Predicting and Distinguishing Attacks on RC4 Keystream Generator. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 491–506. Springer, Heidelberg (2005)
23. Mantin, I., Shamir, A.: A Practical Attack on Broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 152–164. Springer, Heidelberg (2002)
24. Maximov, A.: Two Linear Distinguishing Attacks on VMPC and RC4A and Weakness. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 342–358. Springer, Heidelberg (2005)
25. Maximov, A., Khovratovich, D.: New State Recovery Attack on RC4. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 297–316. Springer, Heidelberg (2008)
26. Mironov, I.: Not So Random Shuffles of RC4. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 304–319. Springer, Heidelberg (2002)
27. Moen, V., Raddum, H., Hole, K.J.: Weaknesses in the Temporal Key Hash of WPA. Mobile Computing and Communications Review 8, 76–83 (2004)

28. Paul, G., Maitra, S.: Permutation After RC4 Key Scheduling Reveals the Secret. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 360–377. Springer, Heidelberg (2007)
29. Paul, S., Preneel, B.: A New Weakness in the RC4 Keystream Generator and an Approach. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 245–259. Springer, Heidelberg (2004)
30. Paul, G., Rathi, S., Maitra, S.: On Non-Negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key. Design, Codes, and Cryptography 49, 123–134 (2008)
31. Postel, J., Reynolds, J.: A Standard for the Transmission of IP Datagrams over IEEE 802 Networks. RFC 1042 (1988)
32. Roos, A.: A Class of Weak Keys in RC4 Stream Cipher (sci.crypt) (1995), http://groups.google.com/group/sci.crypt.research/msg/078a%a9249d76eacc?dmode=source
33. Sepehrdad, P., Vaudenay, S., Vuagnoux, M.: Discovery and Exploitation of New Biases in RC4. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 74–91. Springer, Heidelberg (2011)
34. Tews, E., Beck, M.: Practical Attacks Against WEP and WPA. In: Proceedings of the Second ACM Conference on Wireless Network Security WISEC 2009, Zurich, Switzerland, pp. 79–86. ACM, New York (2009)
35. Tews, E., Weinmann, R.-P., Pyshkin, A.: Breaking 104 Bit WEP in Less Than 60 Seconds. In: Kim, S., Yung, M., Lee, H.-W. (eds.) WISA 2007. LNCS, vol. 4867, pp. 188–202. Springer, Heidelberg (2008)
36. Tomasevic, V., Bojanic, S., Nieto-Taladriz, O.: Finding an Internal State of RC4 Stream Cipher. Information Sciences: an International Journal 177, 1715–1727 (2007)
37. Vaudenay, S., Vuagnoux, M.: Passive–only key recovery attacks on RC4. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 344–359. Springer, Heidelberg (2007)
38. Wagner, D.: Weak Keys in RC4 (sci.crypt) (1995), http://www.cs.berkeley.edu/~daw/my-posts/my-rc4-weak-keys

A Computation of Biases

Biases were computed using the following formulas:

$$P_{\mathsf{KI}}(i,t) = P_J P_C(i,t) + \frac{1-P_J}{N-1}(1-P_C(i,t))$$

Table 5. Conditional Biases for RC4

If $\bar{g}_i$ holds then $\bar{K}[i] = \bar{f}_i$ with probability p_i. All biases except SVV_db are from [18]. SVV_db is from [33].

row	reference	$\bar{f}$	$\bar{g}$	p
i	A_u15	$2-\sigma_i$	$z_2 = 0, S_t[i] = 0$	Kor_1^0
i	A_s13	$S_t^{-1}[0]-\sigma_i$	$S_t[1] = i, z_1 = i$	Kor_4^1
i	A_u13_1	$S_t^{-1}[z_1]-\sigma_i$	$S_t[1] = i, z_1 = 1-i$	Kor_2^1
i	A_u13_2	$1-\sigma_i$	$S_t[i] = i, S_t[1] = 0, z_1 = i$	Kor_2^0
i	A_u13_3	$1-\sigma_i$	$S_t[i] = i, S_t[1] = 1-i, z_1 = S_t[1]$	Kor_2^0
i	A_s5_1	$S_t^{-1}[z_1]-\sigma_i$	$S_t[1] < i,\ S_t[1]+S_t[S_t[1]] = i,\ z_1 \neq S_t[1], z_1 \neq S_t[S_t[1]], S_t[1] \neq 1$	Kor_3^1
i	A_s5_2	$S_t^{-1}[S_t[1]-S_t[2]]-\sigma_i$	$S_t[1] > i,\ S_t[2]+S_t[1] = i,\ S_t[1] = z_2, S_t^{-1}[S_t[1]-S_t[2]] \neq 1, S_t^{-1}[S_t[1]-S_t[2]] \neq 2$	Kor_3^3
i	A_s5_3	$S_t^{-1}[2-S_t[2]]-\sigma_i$	$S_t[1] > i,\ S_t[2]+S_t[1] = i,\ z_2 = 2-S_t[2], S_t^{-1}[z_2] \neq 1, S_t^{-1}[z_2] \neq 2$	Kor_3^2
i	A_u5_1	$S_t^{-1}[S_t^{-1}[z_1]-i]-\sigma_i$	$S_t[1] = i, z_1 \neq i, S_t^{-1}[z_1] < i, S_t^{-1}[S_t^{-1}[z_1]-i] \neq 1, z_1 \neq 1-i$	Kor_3^2
i	A_u5_2	$1-\sigma_i$	$S_t^{-1}[z_1] = 2, S_t[i] = 1$	Kor_2^0
i	A_u5_3	$1-\sigma_i$	$S_t[1] > -i,\ S_t[i] = i,\ S_t[1] = S_t^{-1}[z_1]-i, S_t^{-1}[z_1] \neq 1$	Kor_3^0
$i>4$	A_u5_4	$S_t^{-1}[z_2]-\sigma_i$	$S_t[1] = 2,\ S_t[4]+2 = i,\ S_t^{-1}[z_2] \neq 1, S_t^{-1}[z_2] \neq 4$	Kor_3^1
i	A_s3	$S_t^{-1}[z_2]-\sigma_i$	$S_t[1] \neq 2, S_t[2] \neq 0, S_t[2]+S_t[1] < i, S_t[2]+S_t[S_t[2]+S_t[1]] = i, S_t^{-1}[z_2] \neq 1, S_t^{-1}[z_2] \neq 2, S_t^{-1}[z_2] \neq S_t[1]+S_t[2]$	Kor_5^1
4	A_4_s13	$S_t^{-1}[0]-\sigma_4$	$S_t[1] = 2, z_2 = 0, S_t[4] \neq 0$	Kor_2^1
4	A_4_u5_1	$S_t^{-1}[N-2]-\sigma_4$	$S_t[1] = 2, z_2 \neq 0, S_t^{-1}[z_2] = 0$	Kor_3^1
4	A_4_u5_2	$S_t^{-1}[N-1]-\sigma_4$	$S_t[1] = 2, z_2 \neq 0, S_t^{-1}[z_2] = 2, K[0]+K[1]+S_0[1] = 2$	Kor_3^1
i	A_neg_1a	$1-\sigma_i$	$S_t[2] = 0, S_t[1] = 2, z_1 = 2$	0
i	A_neg_1b	$2-\sigma_i$	$S_t[2] = 0, S_t[1] = 2, z_1 = 2$	0
i	A_neg_2	$2-\sigma_i$	$S_t[2] = 0, S_t[1] \neq 2, z_2 = 0$	0
i	A_neg_3a	$1-\sigma_i$	$S_t[1] = 1, z_1 = S_t[2]$	0
i	A_neg_3b	$2-\sigma_i$	$S_t[1] = 1, z_1 = S_t[2]$	0
i	A_neg_4a	$-\sigma_i$	$S_t[1] = 0, S_t[0] = 1, z_1 = 1$	0
i	A_neg_4b	$1-\sigma_i$	$S_t[1] = 0, S_t[0] = 1, z_1 = 1$	0
16	SVV_db	$S_t^{-1}[0]-\sigma_i$	$z_i = -16$	$P_{\mathsf{SVV10}}(i,t)$

$$P_{\mathsf{MP}}(i,t) = P_D(i)P_B(i,t)P_0\left(1-\frac{1}{N}\right)+\frac{1}{N}$$
$$P_{008}(i,t) = P_8 P_C(i,t)+\frac{1-P_8}{N-1}\left(1-P_C(i,t)\right)$$
$$P_{009}(i,t) = P_9 P_C(i,t)+\frac{1-P_9}{N-1}\left(1-P_C(i,t)\right)$$
$$\mathsf{Kor}_c^b(i,t) = r_c(t)P_E^b(i,t)+\frac{1-r_c(t)}{N-1}\left(1-P_E^b(i,t)\right)$$
$$P_{\mathsf{SVV10}}(i,t) = P_{\mathsf{db}}P_C(i,t)+\frac{1-P_{\mathsf{db}}}{N-1}\left(1-P_C(i,t)\right)$$

where $P_J=\frac{2}{N}$, $P_0=\left(\frac{N-1}{N}\right)^{N-2}$, $P_8=\frac{1.05}{N}$, $P_9=\frac{1.0338}{N}$, $P_{\mathsf{db}}=0.038488$,

$$P_A^b(i,t) = \left(\tfrac{N-b}{N}\right)^{i-t-1} \qquad r_1(t) = \left(\tfrac{N-1}{N}\right)^{N-t}$$
$$P_B(i,t) = \textstyle\prod_{k=1}^{i-t-1}\frac{N-k}{N} \qquad r_2(t) = \left(\tfrac{N-2}{N}\right)^{N-t-1}$$
$$P_C(i,t) = P_A^1(i,t)P_B(i,t)P_0\left(1-\tfrac{1}{N}\right)+\tfrac{1}{N} \qquad r_3(t) = \left(\tfrac{N-2}{N}\right)\left(\tfrac{N-3}{N}\right)^{N-t-1}$$
$$P_D(i) = \tfrac{(N-i-1)(N-i)}{N^3}\left(\tfrac{N-2}{N}\right)^{N-3+i}\left(\tfrac{N-1}{N}\right)^3 \qquad r_4(t) = \left(\tfrac{N-1}{N}\right)\left(\tfrac{N-2}{N}\right)^{N-t-1}$$
$$P_E^b(i,t) = P_A^b(i,t)P_B(i,t)\left(1-\tfrac{1}{N}\right)+\tfrac{1}{N} \qquad r_5(t) = \left(\tfrac{N-4}{N}\right)^{N-t}$$

These formulas are new. Biases were originally provided with probabilities for $t=-1$. Except for the Korek biases, we have checked that the probabilities match with an error less than 4%. The accuracy of formulas for Korek biases are still unclear but orders of magnitude are correct. They were inspired by [6]. Details on how we have got all formulas are omitted due to lack of space.

Improved Generic Algorithms for Hard Knapsacks

Anja Becker[⋆,1], Jean-Sébastien Coron[3], and Antoine Joux[1,2]

[1] University of Versailles Saint-Quentin-en-Yvelines
[2] DGA
[3] University of Luxembourg

Abstract. At Eurocrypt 2010, Howgrave-Graham and Joux described an algorithm for solving hard knapsacks of density close to 1 in time $\tilde{\mathcal{O}}(2^{0.337n})$ and memory $\tilde{\mathcal{O}}(2^{0.256n})$, thereby improving a 30-year old algorithm by Shamir and Schroeppel. In this paper we extend the Howgrave-Graham–Joux technique to get an algorithm with running time down to $\tilde{\mathcal{O}}(2^{0.291n})$. An implementation shows the practicability of the technique. Another challenge is to reduce the memory requirement. We describe a constant memory algorithm based on cycle finding with running time $\tilde{\mathcal{O}}(2^{0.72n})$; we also show a time-memory tradeoff.

1 Introduction

The Knapsack Problem. Given a list of n positive integers $(a_1, a_2, \ldots, a_n)$ and another positive integer S such that:

$$S = \sum_{i=1}^{n} \epsilon_i \cdot a_i \quad , \tag{1}$$

where $\epsilon_i \in \{0, 1\}$, the knapsack problem consists in recovering the coefficients ϵ_i. The vector $\epsilon = (\epsilon_1, .., \epsilon_n)$ is called the solution of the knapsack problem. It is well known that the decisional version of the knapsack problem is NP-complete [4].

The first cryptosystem based on the knapsack problem was introduced by Merkle and Hellmann [10] in 1978, and subsequently broken by Shamir [14] using lattice reduction. For random knapsack problems the Lagarias-Odlyzko attack [7] can solve knapsacks with density $d < 0.64$, given an oracle solving the shortest vector problem (SVP) in lattices; the density of a knapsack is defined as:

$$d := \frac{n}{\log_2 \max_i a_i} \quad .$$

The Lagarias-Odlyzko attack was further improved by Coster *et al.* [3] to knapsack densities up to $d < 0.94$. Since solving SVP is known to be NP-hard [1], in practice, the shortest vector oracle is replaced by a lattice reduction algorithm such as LLL [8] or BKZ [12].

⋆ The first author was mainly funded by a scholarship of the Gottlieb Daimler- und Karl Benz-Stiftung.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 364–385, 2011.

The Schroeppel-Shamir Algorithm. For a knapsack of density close to 1 lattice reduction algorithms do not seem to apply. Until 2009, the best algorithm for such hard knapsacks was due to Schroeppel and Shamir [13] with time complexity $\tilde{\mathcal{O}}(2^{n/2})$ and memory $\tilde{\mathcal{O}}(2^{n/4})$. This is the same running time as the straightforward meet-in-the-middle algorithm but with a lower memory requirement of $\tilde{\mathcal{O}}(2^{n/4})$ instead of $\tilde{\mathcal{O}}(2^{n/2})$. A drawback is that the Schroeppel-Shamir algorithm requires sophisticated data structure such as balanced trees which can be difficult to implement in practice. A simpler but heuristic variant of Schroeppel-Shamir was described in [5] with the same time and memory complexity; we recall this variant in Sect. 2.1. We also recall how to solve *unbalanced* knapsack problems, where the Hamming weight of the coefficient vector $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ can be much smaller than n.

The Howgrave-Graham–Joux Algorithm. At Eurocrypt 2010, Howgrave-Graham and Joux introduced a more efficient algorithm [5] for hard knapsacks. While in Schroeppel-Shamir's algorithm the knapsack instance is divided into two halves with no overlap, the new algorithm allows for overlaps, which induces more degrees of freedom. This enables to reduce the running time down to $\tilde{\mathcal{O}}(2^{0.337n})$ while keeping the memory requirement reasonably low at $\tilde{\mathcal{O}}(2^{0.256n})$. We recall the Howgrave-Graham–Joux algorithm in Sect. 2.2.

Our Contributions. The main contribution of our paper is to extend the Howgrave-Graham–Joux technique to get a new algorithm with running time down to $\tilde{\mathcal{O}}(2^{0.291n})$. The knapsack instance is divided in two halves with possible overlap, as in the Howgrave-Graham–Joux algorithm, but the set of possible coefficients is extended from $\{0, 1\}$ to $\{-1, 0, +1\}$. This means that a coefficient $\epsilon_i^{(1)} = -1$ in the first half can be compensated with a coefficient $\epsilon_i^{(2)} = +1$ in the second half, the resulting coefficient $\epsilon_i = 0$ of the golden solution being $\epsilon_i = \epsilon_i^{(1)} + \epsilon_i^{(2)} = (-1) + (+1) = 0$. Adding (a few) -1 coefficients brings an additional degree of freedom that enables to again decrease the running time; we describe our new algorithm in Sect. 3. We show the practicality of the technique with an implementation.

Another challenge in solving knapsack problems is to reduce the memory requirement. We first describe a simple constant memory algorithm based on cycle finding with running time $\tilde{\mathcal{O}}(2^{0.75n})$. We show how to improve this algorithm down to $\tilde{\mathcal{O}}(2^{0.72n})$ running time still requiring constant memory, by using the Howgrave-Graham–Joux technique. Eventually, we present a time-memory tradeoff for the Schroeppel-Shamir algorithm downto $\tilde{\mathcal{O}}(2^{n/16})$ memory.

2 Existing Algorithms

2.1 The Schroeppel-Shamir Algorithm

We present the Schroeppel-Shamir algorithm [13] under the simpler heuristic variant described in [5]. We consider a knapsack as in (1) and for simplicity we assume that n is a multiple of 4. We write the knapsack sum S as:

$$S = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4$$

where each σ_i is a knapsack of $n/4$ elements, that is,

$$\sigma_1 = \sum_{i=1}^{n/4} \epsilon_i a_i, \quad \sigma_2 = \sum_{i=n/4+1}^{n/2} \epsilon_i a_i, \quad \sigma_3 = \sum_{i=n/2+1}^{3n/4} \epsilon_i a_i, \quad \sigma_4 = \sum_{i=3n/4+1}^{n} \epsilon_i a_i \ . \quad (2)$$

We guess a middle value σ_M of $n/4$ bits which leads to the equations:

$$\sigma_1 + \sigma_2 = \sigma_M \mod 2^{n/4} \text{ and } \sigma_3 + \sigma_4 = S - \sigma_M \mod 2^{n/4} \ .$$

We solve the two equations separately and merge the result. More precisely, we first construct a sorted list $\{\sigma_2\}$ of all $2^{n/4}$ possible values for σ_2. Then for each possible σ_1, we use the sorted list $\{\sigma_2\}$ to find all σ_2 such that $\sigma_1 + \sigma_2 = \sigma_M \mod 2^{n/4}$. This gives a list $\{\sigma_{12}\}$ of knapsack values $\sigma_{12} = \sigma_1 + \sigma_2$ such that $\sigma_{12} = \sigma_M \mod 2^{n/4}$; the size of the list $\{\sigma_{12}\}$ is heuristically $\tilde{\mathcal{O}}(2^{n/4})$ and it can be built in time $\tilde{\mathcal{O}}(2^{n/4})$. We build the list $\{\sigma_{34}\}$ of knapsack values $\sigma_{34} = \sigma_3 + \sigma_4$ such that $\sigma_{34} = S - \sigma_M \mod 2^{n/4}$ in an analogue way. Eventually, we find a collision between the two lists $\{\sigma_{12}\}$ and $\{S - \sigma_{34}\}$ of two elements σ_{12} and σ_{34}, respectively. For the right guess of σ_M we have found elements such that $\sigma_{12} + \sigma_{34} = S$, thereby solving the knapsack problem.

The time required to build the two lists $\{\sigma_{12}\}$ and $\{\sigma_{34}\}$ is $\tilde{\mathcal{O}}(2^{n/4})$. Then by sorting those two lists the collision can be found in time $\tilde{\mathcal{O}}(2^{n/4})$. Since we have to guess σ_M which is a $n/4$-bit value, the total running time is $\tilde{\mathcal{O}}(2^{n/2})$ and the required memory is $\tilde{\mathcal{O}}(2^{n/4})$.

Unbalanced Case. We say that a knapsack is unbalanced when the Hamming weight of the coefficient vector $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$ is known and equal to ℓ where ℓ significantly differs from $n/2$. As shown in [5], one can adapt the previous algorithm as follows: instead of taking all possible knapsacks of $n/4$ elements we only consider knapsacks of Hamming weight exactly $\ell/4$ (assuming that ℓ is divisible by 4). Note that if the correct solution is not perfectly balanced between the four quarters, then such solution will be missed. This problem is easily solved by permuting the order of the elements in the knapsacks until the Hamming weight of each quarter is equal to $\ell/4$. As explained in [5], the expected number of required repetitions is polynomial in n. Thus, this change does not modify the value of the exponent in the running time.

For $\ell = \tau \cdot n$ the size of the lists $\{\sigma_2\}$ and $\{\sigma_4\}$ becomes $\binom{n/4}{\ell/4} \approx 2^{h(\tau)n/4}$ where:

$$h(x) := -x \cdot \log_2 x - (1-x) \cdot \log_2(1-x) \ .$$

Again, we can guess a middle value σ_M modulo $2^{h(\tau)n/4}$; as previously the two lists $\{\sigma_{12}\}$ and $\{\sigma_{34}\}$ can be built in time $\tilde{\mathcal{O}}(2^{h(\tau)n/4})$ and a collision is found in time $\tilde{\mathcal{O}}(2^{h(\tau)n/4})$. Therefore, the total time complexity is $\tilde{\mathcal{O}}(2^{h(\tau)n/2})$ and the memory complexity is $\tilde{\mathcal{O}}(2^{h(\tau)n/4})$.

2.2 The Howgrave-Graham–Joux Algorithm

We consider the knapsack (1). For simplicity we assume again that n is a multiple of four and additionally that the Hamming weight of the coefficients ϵ_i is equal to $n/2$. To find a solution $x \in \{0,1\}^n$, the basic idea of Howgrave-Graham and Joux [5] is to split the knapsack into two subknapsacks of size n and of Hamming weight $n/4$. In other words, we write S as the sum $\sigma_1 + \sigma_2$ of two subknapsacks with Hamming weight $n/4$ chosen among the n knapsack elements,

$$\underbrace{\sum_{i=1}^{n} a_i y_i}_{\sigma_1} + \underbrace{\sum_{i=1}^{n} a_i z_i}_{\sigma_2} = S \tag{3}$$

where $y_i, z_i \in \{0,1\}$. Clearly, the combination of two solutions $y \in \{0,1\}^n$ and $z \in \{0,1\}^n$ gives a solution to the original knapsack when the two solutions do not overlap. In other words, we represent any x_i by a binary tuple (y_i, z_i), replacing 0 by $(0,0)$ and 1 by $(1,0)$ or $(0,1)$, respectively. As a consequence, a single solution of the original knapsack problem decomposes into many different representations. This is used to reduce the overall running time as described in the following. We choose a modulus M, a random element $R \in \mathbb{Z}_M$ and we only consider decompositions such that:

$$\sigma_1 = \sum_{i=1}^{n} a_i y_i \equiv R \pmod{M} \text{ and } \sigma_2 = \sum_{i=1}^{n} a_i z_i \equiv S - R \pmod{M} .$$

Since both σ_1 and σ_2 are knapsacks of Hamming weight $n/4$ over n elements, the expected number of solutions to each of these two modular subknapsacks is

$$L = \frac{\binom{n}{n/4}}{M} .$$

Assuming that the lists of solutions of the two subknapsacks can be obtained very efficiently (in time $\tilde{\mathcal{O}}(L)$), it remains to paste the partial solutions together to obtain a solution to the original knapsack. We therefore search a collision between the values σ_1 and $S - \sigma_2$, for all y and z in the two lists of solutions. Since the expected number of such collisions is small, this can be done in $\tilde{\mathcal{O}}(L)$. To minimize the overall running time, M is chosen to be as large as possible. More precisely, one chooses M as a number close to the number of decompositions of the original solution into two solutions of the two subknapsacks, i.e. $M \approx 2^{n/2}$. Under these assumptions, the running time would be reduced down to $\tilde{\mathcal{O}}(2^{h(1/4)n}/2^{n/2}) = \tilde{\mathcal{O}}(2^{0.3113})$.

However, there are several technical difficulties with this approach. First, there is an exponentially small number of bad weights $(a_1, .., a_n)$ where the algorithm fails. Second, the assumption that the list of solutions of each subknapsack can be obtained in time $\tilde{\mathcal{O}}(L)$ is quite strong and difficult to achieve. [5] describes a heuristic algorithm, supported by an implementation, and claims that it achieves

the $\tilde{\mathcal{O}}(2^{0.3113n})$ running time. However, May and Meurer recently discovered a mistake in the analysis of this algorithm [9]; they showed that the asymptotic running time of the Howgrave-Graham–Joux algorithm is actually $\tilde{\mathcal{O}}(2^{0.337n})$; see [2] for more details.

3 New Algorithm with Better Time Complexity

3.1 Theoretical Improvement

Our basic idea is to enhance the algorithm of [5] by allowing more representations of the solution of the initial knapsack. Instead of decomposing the original solution into two binary coefficient vectors of weight $n/4$, we consider decompositions that contain 0s, 1s and -1s. More precisely, we choose a parameter α and search for decompositions containing $(1/4+\alpha)n$ 1s and αn -1s. Put differently, we split the 1s of the original solution into pairs $(0,1)$ or $(1,0)$ as before and the 0s into pairs $(0,0)$, $(1,-1)$ or $(-1,1)$. The number of such decompositions is

$$N_D = \binom{n/2}{n/4}\binom{n/2}{\alpha n, \alpha n, (1/2-2\alpha)n} .$$

As in Sect. 2.2, we choose a modulus $M \approx N_D$, a random value R modulo M and search for solutions of the two subknapsacks

$$\sigma_1 = \sum_{i=1}^{n} a_i y_i \equiv R \pmod{M} \text{ and } \sigma_2 = \sum_{i=1}^{n} a_i z_i \equiv S - R \pmod{M} ,$$

where y and z contain $(1/4+\alpha)n$ 1s and αn -1s each. The expected number of solutions to each of these new modular subknapsacks is

$$L = \frac{\binom{n}{(1/4+\alpha)n, \alpha n, (3/4-2\alpha)n}}{M} .$$

Using:

$$\binom{n}{xn, yn, (1-x-y)n} = \tilde{\mathcal{O}}(2^{g(x,y)n})$$

where:

$$g(x,y) := -x\log_2 x - y\log_2 y - (1-x-y)\log_2(1-x-y)$$

we obtain:

$$\log_2 L \approx n \cdot \left(g(1/4+\alpha, \alpha) - \frac{1}{2} - \frac{g(2\alpha, 2\alpha)}{2} \right) .$$

Assuming that creating the lists and searching for collisions can be done in time $\tilde{\mathcal{O}}(L)$ and minimizing on α, we obtain a time complexity $\tilde{\mathcal{O}}(L) = \tilde{\mathcal{O}}(2^{0.151\,n})$ for $\alpha \approx 0.103$.

This analysis shows that adding more representations of the original solution has the potential to give better algorithms. However, there are many obstacles to

achieve such a good algorithm. A first obstacle is that the size of the modulus M should never be larger than the size of the knapsack elements. Indeed, we want the knapsack after reduction modulo M to behave like a random knapsack, which is not the case if M is larger than the original knapsack elements. Thus, we want to ensure $M < 2^n$. Optimizing for α under this condition, we get $\alpha = 0.05677$ and $L \approx 2^{0.173\,n}$.

3.2 The Basic Building Block

Before describing our algorithm, we recall a classical basic building block that we extensively use. This building block performs the following task: given two lists of numbers $\mathbb{L}_a$ and $\mathbb{L}_b$ of respective sizes $|\mathbb{L}_a|$ and $|\mathbb{L}_b|$, together with two integers M and R, the algorithm computes the list $\mathbb{L}_R$ such that:

$$\mathbb{L}_R = \{x + y \mid x \in \mathbb{L}_a, y \in \mathbb{L}_b \text{ s.t. } x + y \equiv R \pmod{M}\} \quad .$$

To solve this problem, we use a classical algorithm [16] whose description is given in pseudo-code by Algorithm 1.

Algorithm 1. Compute list $\mathbb{L}_R$

```
Sort the lists L_a and L_b (by increasing order of the values modulo M);
Let Target ← R;
Let i ← 0 and j ← |L_b| − 1;
while i < |L_a| and j ≥ 0 do
    Let Sum ← (L_a[i] (mod M)) + (L_b[j] (mod M));
    if Sum < Target then Increment i;
    if Sum > Target then Decrement j;
    if Sum = Target then
        Let i_0, i_1 ← i;
        while i_1 < |L_a| and L_a[i_1] ≡ L_a[i_0] (mod M) do Increment i_1;
        Let j_0, j_1 ← j;
        while j_1 ≥ 0 and L_b[j_1] ≡ L_b[j_0] (mod M) do Decrement j_1;
        for i ← i_0 to i_1 − 1 do
            for j ← j_1 + 1 to j_0 do Append L_a[i] + L_b[j] to L_R
        end
        Let i ← i_1 and j ← j_1;
    end
end
Let Target ← R + M;
Let i ← 0 and j ← |L_b| − 1;
Repeat the above loop with the new target;
```

The complexity of Algorithm 1 is $\tilde{\mathcal{O}}(\max(|\mathbb{L}_a|, |\mathbb{L}_b|, |\mathbb{L}_R|))$. Moreover, assuming that the values of the initial lists modulo M are randomly distributed, the expected size of $\mathbb{L}_R$ is $|\mathbb{L}_a| \cdot |\mathbb{L}_b|/M$. However, this cannot be guaranteed in general.

Using a slight variation of Algorithm 1, it is also possible given $\mathbb{L}_a$ and $\mathbb{L}_b$ together with a target integer R to construct the set:

$$\mathbb{L}_R = \{x + y \mid x \in \mathbb{L}_a, y \in \mathbb{L}_b \text{ s.t. } x + y = R\} \quad .$$

The only differences are that we sort the lists by value (not by modular values) and then run the loop with a single target value R (instead of 2).

3.3 Devising a Concrete Algorithm

In order to achieve a concrete algorithm along the lines of the theoretical analysis from Sect. 3.1, we must be able to solve the subknapsacks that arise after decomposing the original knapsack problem in a reasonably efficient manner. The difficulty here is that a direct use of an adapted Schroeppel-Shamir algorithm is too costly.

Instead, we use the idea of decomposing a knapsack into two subknapsacks several times. More precisely, we introduce three levels of decomposition; see Fig. 1 for an illustration. The first decomposition follows the method described in Sect. 3.1, with a different (smaller) choice for the value α denoting the proportion of -1s added on each side. At the second or middle level, we decompose each subknapsack from the first level into two. We also add some new -1s in the decompositions. The number of additional -1s for each of the four subknapsacks at the middle level is controlled by a new parameter β. In the last level, we finally decompose into a total of eight different subknapsacks. At this level, we use a parameter γ to denote the proportion of extra -1s in the subknapsacks.

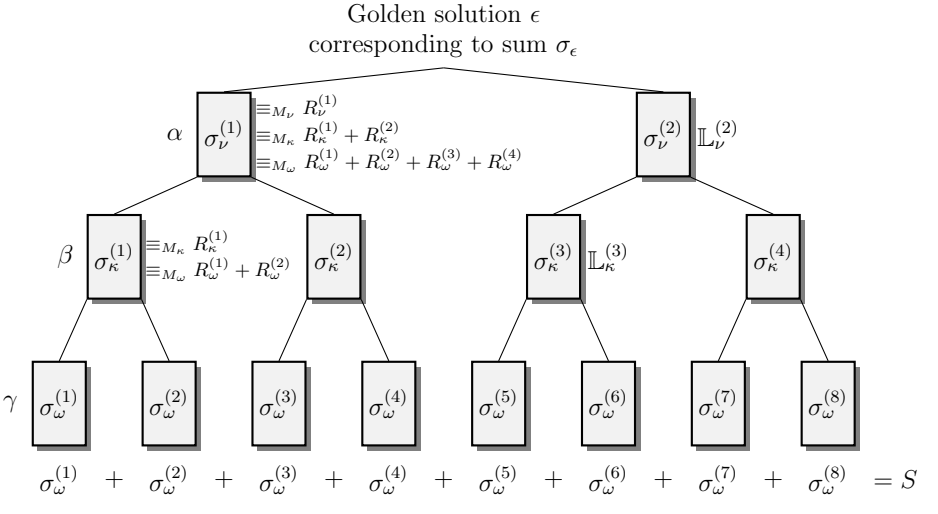


Fig. 1. Iterative decomposition in three steps. $\sigma_\chi^{(j)}$: partial sum, $R_\chi^{(j)}$: target value, M_χ : modulus, α, β and γ : proportion of additional -1s.

Notation. We use a different Greek letter (ϵ, κ, ω or ν) to denote the coefficient vectors of each subknapsack. In the original knapsack, we carry on using the letter ϵ. At the first level of decomposition, we now use $\nu^{(1)}$ and $\nu^{(2)}$ for the coefficient vectors of the two subknapsacks. At the middle level, we choose the notation $\kappa^{(1)}$ to $\kappa^{(4)}$. At the bottom level, we use the letters $\omega^{(1)}$ to $\omega^{(8)}$. We then let $N_\chi(x)$ denote the number of occurrences of $x \in \{-1, 0, 1\}$ in the coefficient vector χ. For a knapsack of n elements, we have:

$$\begin{array}{ll} N_\epsilon(1) = n/2, & N_\epsilon(-1) = 0, \\ N_\nu(1) \approx (1/4 + \alpha)n, & N_\nu(-1) \approx \alpha n, \\ N_\kappa(1) \approx (1/8 + \alpha/2 + \beta)n, & N_\kappa(-1) \approx (\alpha/2 + \beta)n, \\ N_\omega(1) \approx (1/16 + \alpha/4 + \beta/2 + \gamma)n, & N_\omega(-1) \approx (\alpha/4 + \beta/2 + \gamma)n \ . \end{array}$$

We always have $N_\chi(0) = n - N_\chi(1) - N_\chi(-1)$. Since all these numbers need to be rounded to integers for a concrete knapsack instance, we write $\approx$ instead of $=$ above. For each of the coefficient vectors $\chi^{(j)}$ we introduce the corresponding partial sum:

$$\sigma_\chi^{(j)} = \sum_{i=1}^{n} \chi_i^{(j)} a_i \ .$$

To control the size of the lists of solutions that arise at each level of decomposition, we introduce a modulus and target values for each of the subknapsacks. We denote the modulus corresponding to the bottom level by M_ω, we introduce 7 random values $R_\omega^{(j)}$ (for $1 \le j \le 7$) and let $R_\omega^{(8)} = S - \sum_{j=1}^{7} R_\omega^{(j)}$. We solve the eight modular subknapsacks:

$$\sigma_\omega^{(j)} \equiv R_\omega^{(j)} \pmod{M_\omega} \quad \text{for } 1 \le j \le 8 \ .$$

We denote by $\mathbb{L}_\omega^{(j)}$, the list of solutions of each of these subknapsacks.

Basic Principle and Modular Constraints. To build solutions at the middle level κ, we consider sums of two partial solutions from two neighboring lists $\mathbb{L}_\omega^{(2j-1)}$ and $\mathbb{L}_\omega^{(2j)}$ containing solutions of the last level. By construction, we see that:

$$\sigma_\kappa^{(j)} = \sigma_\omega^{(2j-1)} + \sigma_\omega^{(2j)} \equiv R_\omega^{(2j-1)} + R_\omega^{(2j)} \pmod{M_\omega}$$

which means that all these partial sums already have some fixed value modulo M_ω. To prune the size of the lists of solutions at this level, we add an extra constraint modulo M_κ (chosen coprime to M_ω). Thus, we introduce three random values $R_\kappa^{(j)}$ (for $1 \le j \le 3$) and let $R_\kappa^{(4)} = S - \sum_{j=1}^{3} R_\kappa^{(j)}$. The new lists of solutions are denoted by $\mathbb{L}_\kappa^{(j)}$.

For the first level, we proceed similarly, adding partial solutions from $\mathbb{L}_\kappa^{(2j-1)}$ and $\mathbb{L}_\kappa^{(2j)}$. Clearly, the resulting sums already have fixed values modulo M_κ and M_ω. Again, we introduce a modulus M_ν, a random value $R_\nu^{(1)}$ and we let $R_\nu^{(2)} = S - R_\nu^{(1)}$ to reduce the size of the lists.

Finally, the (presumably unique) solution of the original knapsack is found by searching for a collision of the form $\sigma_\nu^{(1)} + \sigma_\nu^{(2)} = S$ with $\sigma_\nu^{(1)} \in \mathbb{L}_\nu^{(1)}$ and $\sigma_\nu^{(2)} \in \mathbb{L}_\nu^{(2)}$. Figure 1 illustrates the technique.

To transform this informal description into a formal algorithm and to analyze its complexity, we need to specify how the lists $\mathbb{L}_\omega^{(j)}$ are constructed. We also explain how to merge solutions from one level to solutions at the next level and specify the choices of the moduli M_ω, M_κ and M_ν in the next paragraph.

Algorithmic Details. The eight lists $\mathbb{L}_\omega^{(j)}$ can be constructed using a straightforward adaptation of the simple birthday paradox algorithm. It suffices to split the n elements into two random subsets of size $n/2$ and to assume that the 1s and -1s are evenly[1] distributed between the two halves. As with the case of binary coefficient vectors, the probability of this event is the inverse of a polynomial in n. Thus by repeating polynomially many times, we recover all of $\mathbb{L}_\omega^{(j)}$ with overwhelming probability. Assuming that the elements in $\mathbb{L}_\omega^{(j)}$ are random modulo M_ω, the expected size of $\mathbb{L}_\omega^{(j)}$ is:

$$L_\omega = \frac{\mathcal{L}_\omega}{M_\omega} = \frac{\binom{n}{N_\omega(1), N_\omega(-1), N_\omega(0)}}{M_\omega} \ ,$$

where $\mathcal{L}_\omega$ is the multinomial coefficient that counts the number of ways to choose $N_\omega(1)$ 1s, $N_\omega(-1)$ -1s and $N_\omega(0)$ 0s among n elements. Since the number of ways to choose $N_\omega(1)/2$ 1s, $N_\omega(-1)/2$ -1s and $N_\omega(0)/2$ 0s among $n/2$ elements is $\approx \mathcal{L}_\omega^{1/2}$ for large n, the running time of the construction of each $\mathbb{L}_\omega^{(j)}$ is $\max(|\mathbb{L}_\omega^{(j)}|, \mathcal{L}_\omega^{1/2})$.

At the middle level, the expected size of $\mathbb{L}_\kappa^{(j)}$ is upper bounded by

$$L_\kappa = \frac{\mathcal{L}_\kappa}{M_\omega \cdot M_\kappa} = \frac{\binom{n}{N_\kappa(1), N_\kappa(-1), N_\kappa(0)}}{M_\omega \cdot M_\kappa} \ .$$

This is only an upper bound on the expected size since the definition of L_κ ignores the fact that we discard solutions that cannot be decomposed with the modular constraints of the lower level.

To construct these lists, we match values from $\mathbb{L}_\omega^{(2j-1)}$ and $\mathbb{L}_\omega^{(2j)}$ modulo M_κ using Algorithm 1 from Sect. 3.2. We let $\mathbb{K}_\kappa^{(j)}$ denote the resulting list. We then remove inconsistent solutions from $\mathbb{K}_\kappa^{(j)}$ in order to produce $\mathbb{L}_\kappa^{(j)}$. We say that a solution is inconsistent when the vector $\omega^{(2j-1)} + \omega^{(2j)}$ contains 2s or -2s and/or does not have the number of 1s, -1s and 0s specified by $N_\kappa(1)$, $N_\kappa(-1)$ and $N_\kappa(0)$. According to Sect. 3.2, the cost of this step is $\max(|\mathbb{L}_\omega^{(2j-1)}|, |\mathbb{L}_\omega^{(2j)}|, |\mathbb{K}_\kappa^{(j)}|)$.

Proceeding in the same way, we give an upper bound on the expected size of $\mathbb{L}_\nu^{(j)}$ by

$$L_\nu = \frac{\mathcal{L}_\nu}{M_\omega \cdot M_\kappa \cdot M_\nu} = \frac{\binom{n}{N_\nu(1), N_\nu(-1), N_\nu(0)}}{M_\omega \cdot M_\kappa \cdot M_\nu} \ .$$

[1] Or almost evenly when the number of 1s and/or -1s are odd.

Using the same notation as above, the cost to construct the two lists $\mathbb{L}_\nu^{(j)}$ is $\max(|\mathbb{L}_\kappa^{(2j-1)}|, |\mathbb{L}_\kappa^{(2j)}|, |\mathbb{K}_\nu^{(j)}|)$.

Finally, the last step is to apply the integer variant of Algorithm 1 to the two integer lists $\mathbb{L}_\nu^{(1)}$ and $\mathbb{L}_\nu^{(2)}$, obtaining a list $\mathbb{K}_0$ of (possibly inconsistent) solutions. The cost of this step is $\max(|\mathbb{L}_\nu^{(1)}|, |\mathbb{L}_\nu^{(2)}|, |\mathbb{K}_0|)$.

To estimate the size of $\mathbb{K}_0$, we count the number of expected solutions in a modular merge modulo the multiple of $M_\omega \cdot M_\kappa \cdot M_\nu$ closest to 2^n. This overestimates the size of $\mathbb{K}_0$ since it is slightly easier to find a knapsack solution modulo this value than a knapsack solution over the integers. This yields an estimate equal to:

$$L_\nu^2 \cdot \frac{M_\nu \cdot M_\kappa \cdot M_\omega}{2^n} \ .$$

If $\mathbb{K}_0$ contains at least one *consistent* solution, we obtain a solution of the initial knapsack problem.

To conclude the description of the algorithm, we need to specify the values of the moduli M_ω, M_κ and M_ν. The key idea at this point is to choose each modulus to ensure that each solution appearing at a given level is represented (on average) by a single decomposition at the previous level. Indeed, if we add a larger modular constraint, we lose solutions from one level to the next and if we choose a smaller constraints, we construct each solution many times which increases the overall cost. Using binomials and multinomials to compute the number of decompositions we obtain the following conditions for the values of the moduli:

$$M_\omega \approx \binom{N_\kappa(1)}{N_\kappa(1)/2} \cdot \binom{N_\kappa(-1)}{N_\kappa(-1)/2} \cdot \binom{N_\kappa(0)}{N_\omega(1)-N_\kappa(1)/2, N_\omega(-1)-N_\kappa(-1)/2, \star} \approx$$
$$2^{(1/8+\alpha+2\beta-2\gamma\log_2\gamma-(7/8-\alpha-2\beta-2\gamma)\log_2(7/8-\alpha-2\beta-2\gamma)+(7/8-\alpha-2\beta)\log_2(7/8-\alpha-2\beta)} \ ,$$

$$M_\kappa \cdot M_\omega \approx \binom{N_\nu(1)}{N_\nu(1)/2} \cdot \binom{N_\nu(-1)}{N_\nu(-1)/2} \cdot \binom{N_\nu(0)}{N_\kappa(1)-N_\nu(1)/2, N_\kappa(-1)-N_\nu(-1)/2, \star} \approx$$
$$2^{(1/4+2\alpha-2\beta\log_2\beta-(3/4-2\alpha-2\beta)\log_2(3/4-2\alpha-2\beta)+(3/4-2\alpha)\log_2(3/4-2\alpha))n} \ ,$$

$$\begin{aligned} M_\nu \cdot M_\kappa \cdot M_\omega &\approx \binom{n/2}{n/4} \cdot \binom{n/2}{N_\nu(-1), N_\nu(-1), \star} \\ &\approx 2^{(1/2-2\alpha\log_2\alpha-(1/2-2\alpha)\log_2(1/2-2\alpha)+(1/2)\log_2(1/2))n} \\ &\approx 2^{(-2\alpha\log_2\alpha-(1/2-2\alpha)\log_2(1/2-2\alpha))n} \ . \end{aligned}$$

The $\star$ symbol in the above multinomials denotes the number of remaining elements (corresponding to 0s) after specifying the number of 1s and -1s introduced to decompose the set of 0s from the lower level.

The overall running time of the algorithm is the maximum of the individual costs to run Algorithm 1 and the construction of the eight lists, which gives:

$$\tilde{\mathcal{O}}(\max(\max_j |\mathbb{L}_\omega^{(j)}|, \max_j \mathcal{L}_\omega^{1/2}, \max_j |\mathbb{K}_\kappa^{(j)}|, \max_j |\mathbb{L}_\kappa^{(j)}|, \max_j |\mathbb{K}_\nu^{(j)}|, \max_j |\mathbb{L}_\nu^{(j)}|, |\mathbb{K}_0|)) \ .$$

Assuming that each list has a size close to its expected value (see Sect. 3.5), the expected running time is:

$$T(\alpha,\beta,\gamma) = \tilde{\mathcal{O}}(\max(L_\omega, \mathcal{L}_\omega^{1/2}, \frac{L_\omega^2}{M_\kappa}, L_\kappa, \frac{L_\kappa^2}{M_\nu}, L_\nu, L_\nu^2 \cdot \frac{M_\nu \cdot M_\kappa \cdot M_\omega}{2^n})) \ .$$

Since none of the $\mathbb{K}_\chi$ lists need to be stored, the amount of memory required is:

$$\tilde{\mathcal{O}}(\max(L_\omega, \mathcal{L}_\omega^{1/2}, L_\kappa, L_\nu)) \ .$$

Finally, there is an additional, very important, parameter to consider, the probability of success p_{succ} taken over the possible random choices of the $R_\chi^{(j)}$ values. This parameter is quite tricky to estimate because it varies depending on the initial knapsack that we are solving. As an illustration, consider the knapsack whose elements are all equal to 0. It is clear that unless all the random $R_\chi^{(j)}$ are chosen equal to 0 then the algorithm cannot succeed. As a consequence, in this case the probability of success is very low. There are many other bad knapsacks; however, for a random knapsack, the expected probability of success is not too small (see Sect. 3.4 for a discussion).

Numerical Results for the Complexity Analysis. Minimizing the expected running time $T(\alpha,\beta,\gamma)$ results in:

$$\alpha = 0.0267, \quad \beta = 0.0168, \quad \gamma = 0.0029 \ .$$

With these values, we obtain:

$$\begin{array}{l} \mathcal{L}_\omega \approx 2^{0.532\,n},\ L_\omega \approx 2^{0.291\,n},\ L_\kappa \approx 2^{0.279\,n},\ L_\nu \approx 2^{0.217\,n} \text{ and} \\ M_\omega \approx 2^{0.241\,n},\ M_\kappa \approx 2^{0.291\,n},\ M_\nu \approx 2^{0.267\,n} \ . \end{array}$$

As a consequence, we find that both the time and memory complexity are equal to $\tilde{\mathcal{O}}(2^{0.291\,n})$. We can also check that the product of the three moduli $M_\omega \cdot M_\kappa \cdot M_\nu$ is smaller than the size of the numbers in the initial knapsack, i.e. 2^n.

However, we remark that γ is so small that for any achievable knapsack size n, the number of -1s added at the last level is 0 in practice. Thus, in order to improve the practical choices of the number of -1s at the higher levels, it is better to adjust the minimization with the added constraint $\gamma = 0$. This leads to the alternative values:

$$\alpha = 0.0194, \quad \beta = 0.0119, \quad \gamma = 0 \ .$$

With these values, we obtain:

$$\begin{array}{l} \mathcal{L}_\omega \approx 2^{0.463\,n},\ L_\omega \approx 2^{0.295\,n},\ L_\kappa \approx 2^{0.284\,n},\ L_\nu \approx 2^{0.234\,n} \text{ and} \\ M_\omega \approx 2^{0.168\,n},\ M_\kappa \approx 2^{0.295\,n},\ M_\nu \approx 2^{0.272\,n} \ . \end{array}$$

We can also remark that by choosing $\alpha = \beta = \gamma = 0$, we recover the time complexity $\tilde{\mathcal{O}}(2^{0.337\,n})$ given by May and Meurer [9] for the algorithm of [5]. However, in our case, the memory complexity is also $\tilde{\mathcal{O}}(2^{0.337\,n})$, which indicates that our algorithm can probably be improved in this respect. In the full version of the paper [2], we also consider the unbalanced case and possible extensions.

3.4 Analysis of the Probability of Success

In order to analyze the probability of success, it is convenient to bear in mind Fig. 1. We are starting from an unknown but fixed golden solution of the knapsack and we wish to decompose it seven times. (At each step we represent the 0s, 1s and -1s of the current coefficient vector by a tuple (i, j) where $i, j \in \{0, 1, -1\}$.) For each of the seven splits, we add a modular constraint modulo a number very close to the total number of decompositions. For example, during the top level split, we are specifying that the sum of the left hand-side after the splitting should be congruent to $R_\nu^{(1)}$ modulo M_ν, to $R_\kappa^{(1)} + R_\kappa^{(2)}$ modulo M_κ and to $R_\omega^{(1)} + R_\omega^{(2)} + R_\omega^{(3)} + R_\omega^{(4)}$ modulo M_ω. Since the three moduli are coprime, this is equivalent to simply specifying a value modulo $M_\nu \cdot M_\kappa \cdot M_\omega$. Each of the decompositions is considered successful if the current golden solution admits at least one way of splitting which satisfies the modular constraint. In this case, we focus on one of the admissible solutions for which we search for a decomposition in the level below. Fixing the solution on the left-hand side also determines the solution of the right-hand side.

Clearly, if each of the seven decompositions succeeds, the initial solution can be found by the algorithm. Assuming independence, the overall probability of success is at least equal[2] to the product of the probability of success of the individual decompositions. If we do not assume independence, we can still say that the overall probability of failure is smaller than the individual probabilities of failure.

Purely Random Heuristic Model. One approach to the analysis of the probability of an individual decomposition succeeding is to assume that for each of the possible decompositions, the resulting modular sum is a random value. We already know that there are knapsacks for which this assumption does not hold, as illustrated by the all-zero example. This is true for a large number of random, however, and is a very useful benchmark for the following analysis. For simplicity, we assume here that the number of possible decompositions is equal to the modulus M for a large set of random knapsacks.

In this case, it is well-known that for large values of M, the proportion of modular values which are not attained after picking M random values is close to $e^{-1} \simeq 0.36$.

Experimental Behavior of Decompositions. In Section 5, we describe an implementation of our algorithm on a 80-bit knapsack. To better understand the behavior of this implementation, it is useful to determine the probability of success of each decomposition. Three levels of decomposition occur. At the top level, a balanced golden solution with 40 zeros and 40 ones needs to be split into two partial solutions with 22 ones and two -1s each. At the middle level, a golden solution with 22 ones and two -1s is to be split into two partial solutions

[2] The probability can be larger, since we ignore multiple correct splits when they occur.

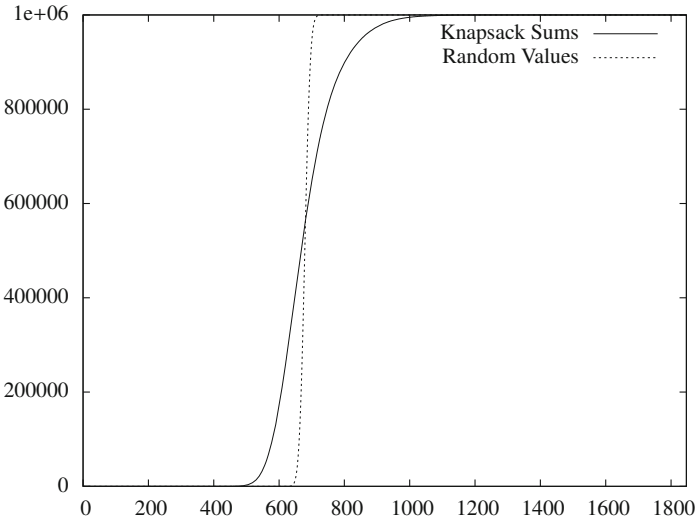


Fig. 2. Cumulative number of knapsacks (in a million) with less than a given number of not obtained values

with 12 ones and two -1s. Finally, at the bottom level, we split 12 ones and two -1s into twice 6 ones and one -1.

At the top level, the number of possible decompositions of a golden solution is larger than $\binom{40}{22}\binom{40}{2,2,36} \approx 2^{56}$. As a consequence, it is not possible to perform experimental statistics of the modular values of such a large set. At the middle level, the number of decompositions is larger than $\binom{22}{11}\binom{2}{1}\binom{46}{1,1,44} \approx 2^{32}$. Thus, it is possible to perform some experiments, but doing a large number of tests to perform a statistical analysis of the modular values is very cumbersome. At the bottom level, the number of decompositions of a golden solution is $\binom{12}{6}\binom{2}{1} = 1848$. This is small enough to perform significant statistics and, in particular, to study the fraction of modular values which are not obtained (depending on a random choice of 14 knapsack elements, 12 1s and two -1s, to be split). The value of the modulus used in this experiment is 1847, the closest prime to 1848.

During our experimental study, we created one million modular subknapsacks from 14 randomly selected values modulo 1847. Among these values 12 elements correspond to additions and 2 to subtractions. From this set we computed (in $\mathbb{Z}_{1847}$) all of the 1848 values that can be obtained by summing 6 out of the 12 addition elements and subtracting one of the subtraction elements. In each experiment, we counted the number of values which were not obtained; the results are presented in Fig. 2. On the vertical axis we display the cumulative number of knapsacks which result in x or less unobtained values. To allow comparison with the purely random model, we display the same curve computed for one million of experiments where 1848 random numbers modulo 1847 are chosen. In particular, we see on this graph that for 99.99% of the random knapsacks we have

constructed the fraction of unobtained value stays below 2/3. This means that experimentally, the probability of success of a decomposition at the bottom level is, at least, 1/3 for a very large fraction of knapsacks. Assuming independence between the probability of success of the seven splits and a similar behavior of three levels[3], we conclude that for 99.93% of random knapsacks an average number of $3^7 = 2187$ repetitions suffices to solve the initial problem.

Distribution of Modular Sums. When considering the decomposition of a given golden solution (at any level), we can construct the set $\mathcal{B}$ of all left-hand sides which can appear. For this set $\mathcal{B}$ we wish to study the distribution of the scalar product $\mathbf{a} \cdot \mathbf{x} = \sum_{i=1}^{n} a_i x_i \pmod M$, for given knapsack weights a_i. Let $P_{a_1,..,a_n}(\mathcal{B}, c)$ denote the probability that a knapsack of elements $\mathbf{a} = (a_1, .., a_n) \in \mathbb{Z}_M^n$ results in the value c modulo M for a uniformly at random chosen solution $(x_1, .., x_n)$ from $\mathcal{B}$,

$$P_{a_1,\cdots,a_n}(\mathcal{B}, c) = \frac{1}{|\mathcal{B}|} \left| \left\{ (x_1, \cdots, x_n) \in \mathcal{B} \text{ such that } \sum_{i=1}^{n} a_i x_i \equiv c \pmod M \right\} \right| .$$

Our main tool to theoretically study the distribution of the scalar products is the following theorem [11, Theorem 3.2]:

Theorem 1. *For any set $\mathcal{B} \subset \mathbb{Z}_M^n$, the identity:*

$$\frac{1}{M^n} \sum_{(a_1,\cdots,a_n)\in\mathbb{Z}_M^n} \sum_{c\in\mathbb{Z}_M} \left(P_{a_1,\cdots,a_n}(\mathcal{B}, c) - \frac{1}{M} \right)^2 = \frac{M-1}{M|\mathcal{B}|}$$

holds.

With this equation we can prove a weak but sufficient result about the proportion of missed values during a decomposition. Let $\Lambda > 0$ be an arbitrary integer. We want to find an upper bound for the fraction f_Λ of "bad" knapsacks modulo M with less than M/Λ obtained values. First, we remark that for a knapsack $(a_1, \cdots, a_n)$ that reaches less than M/Λ values, at least $(\Lambda - 1)M/\Lambda$ values modulo M are obtained 0 times. Since

$$\sum_{c\in\mathbb{Z}_M} P_{a_1,\cdots,a_n}(\mathcal{B}, c) = 1$$

some values c need to be obtained many times. As a consequence, we find that

$$\sum_{c\in\mathbb{Z}_M} \left(P_{a_1,\cdots,a_n}(\mathcal{B}, c) - \frac{1}{M} \right)^2 \geq \frac{\Lambda - 1}{\Lambda} \cdot M \cdot \frac{1}{M^2} + \frac{1}{\Lambda} \cdot M \cdot \frac{(\Lambda - 1)^2}{M^2} = \frac{\Lambda - 1}{M} .$$

[3] The limited number of experiments we have performed for the middle level seem to indicate a comparable behavior. We performed 100 experiments and the number of not obtained values remained in the range 42% – 43.1%.

This implies that the number N_{bad} of "bad" knapsacks satisfies:

$$N_{\text{bad}} \leq M^n \cdot \frac{M-1}{(\Lambda-1)|\mathcal{B}|} \ .$$

With this bound, it is possible to construct a variation of our algorithm with a provable probability of success. Given $\Lambda \geq 10$ as a function of n, we repeat each split for $2\,\Lambda$ random and independently picked values. The probability of failure of such a repeated split is at most $e^{-2} \approx 0.135$, except for a "bad" knapsack. Thus, the global probability of failure on the seven splits is smaller than 95%. By choosing M smaller than $|\mathcal{B}|$ (but close to it), we ensure that the total fraction of bad knapsacks is at most:

$$\frac{7}{\Lambda-1} \ .$$

This fraction becomes arbitrarily small by choosing a large enough value of Λ. Note that the running time is multiplied by $(2\,\Lambda)^3$, since there are three nested levels of decompositions. If a probability of success of 5% is not sufficient, it is possible to increase the probability by repeating the complete algorithm with independent random numbers. A polynomial number of repetition leads to a probability of success exponentially close to 1 (with the exception of the "bad" knapsacks).

3.5 Analysis of the Size of the Lists

Concerning the size of the lists that occur during the algorithm, both the simple heuristic model and the experimental results (see Section 5) predict that the size of the lists are always very close to the theoretical values at the bottom level and smaller (due to the overestimation) at the levels above. It remains to use Theorem 1 to give an upper bound on the size of the various lists.

For the sizes of the lists $\mathbb{L}_\chi$, we can use a direct application of the theorem. The set of concern, $\mathcal{B}$, is the set of all repartitions of 1s, 0s and -1s fulfilling the conditions of $\mathbb{L}_\chi$. The modulus M is the product of all active moduli at the current and preceding levels. That is, for level ω we have $M = M_\omega$; for level κ, $M = M_\omega \cdot M_\kappa$, and for level ν we take $M = M_\omega \cdot M_\kappa \cdot M_\nu$.

Once again, we fix an integer Λ and consider the number F_Λ of knapsacks for which more than $M/(2\,\Lambda)$ values c have a probability that satisfies:

$$P_{a_1,\cdots,a_n}(\mathcal{B}, c) \geq \Lambda/M \ .$$

Due to Theorem 1, we find:

$$\frac{F_\Lambda}{M^n} \cdot \frac{M}{2\,\Lambda} \cdot \frac{(\Lambda-1)^2}{M^2} \leq \frac{M-1}{M|\mathcal{B}|} \leq \frac{1}{|\mathcal{B}|} \ .$$

As a consequence:

$$F_\Lambda \leq \frac{2\,\Lambda}{(\Lambda-1)^2} \cdot \frac{M}{|\mathcal{B}|} \cdot M^n \leq \frac{2\,\Lambda}{(\Lambda-1)^2} \cdot M^n \ .$$

The key point is that for a knapsack which is not one of the F_Λ knapsacks above and for most values of c (all but at most $M/(2\,\Lambda)$), the size of $\mathbb{L}_\chi$ is smaller than Λ times the expected value $|\mathcal{B}|/M$, that is,

$$|\mathbb{L}_\chi| \leq \frac{\Lambda\,|\mathcal{B}|}{M} \quad .$$

To bound the size of the lists $\mathbb{K}_\chi$, we proceed slightly differently. The set $\mathcal{B}$ consists of 1s, 0s and -1s that are allowed in the $\mathbb{L}$ lists and are matched to construct $\mathbb{K}_\chi$. We write $M = M_1 \cdot M_2$, where M_1 is the product of the active moduli for the $\mathbb{L}$ list and M_2 is the modulus that is added when constructing $\mathbb{K}_\chi$. Let $\sigma \pmod M$ denote the target sum as a new modulo constraint for elements in $\mathbb{K}_\chi$. Let $\sigma_L \pmod{M_1}$ and $\sigma_R \pmod{M_1}$ respectively denote the values of the sums in the left-side and right-side lists $\mathbb{L}$. Of course, we have $\sigma_L + \sigma_R \equiv \sigma \pmod{M_1}$. We can write:

$$|\mathbb{K}_\chi| = \sum_{\substack{c \,\in\, \mathbb{Z}_M \\ c \,\equiv\, \sigma_L \pmod{M_1}}} (|\mathcal{B}| \cdot P_{a_1,\cdots,a_n}(\mathcal{B},c)) \cdot (|\mathcal{B}| \cdot P_{a_1,\cdots,a_n}(\mathcal{B},\sigma - c))$$

$$\leq \left[\sum_{\substack{c \,\in\, \mathbb{Z}_M \\ c \,\equiv\, \sigma_L}} (|\mathcal{B}| \cdot P_{a_1,\cdots,a_n}(\mathcal{B},c))^2 \times \sum_{\substack{c \,\in\, \mathbb{Z}_M \\ c \,\equiv\, \sigma_R}} (|\mathcal{B}| \cdot P_{a_1,\cdots,a_n}(\mathcal{B},c))^2 \right]^{1/2} \quad .(4)$$

Thus to estimate the size of the lists $\mathbb{K}_\chi$, we need to find an upper bound for the value of sums of the form:

$$\sum_{\substack{c \,\in\, \mathbb{Z}_M \\ c \,\equiv\, c_1 \pmod{M_1}}} P_{a_1,\cdots,a_n}(\mathcal{B},c)^2 \quad .$$

To do this, it is useful to rewrite the relation from Theorem 1 as:

$$\frac{1}{M^n} \sum_{(a_1,\cdots,a_n) \in \mathbb{Z}_M^n} \sum_{c \in \mathbb{Z}_M} P_{a_1,\cdots,a_n}(\mathcal{B},c)^2 = \frac{M + |\mathcal{B}| - 1}{M|\mathcal{B}|} \quad .$$

Given Λ, we let G_Λ denote the number of knapsacks for which more than $M_1/(8\,\Lambda)$ values c_1 have a sum of squared probabilities that satisfy:

$$\sum_{\substack{c \,\in\, \mathbb{Z}_M \\ c \,\equiv\, c_1 \pmod{M_1}}} P_{a_1,\cdots,a_n}(\mathcal{B},c)^2 \geq \frac{\Lambda^2}{M_1^2 M_2} \quad .$$

We find that

$$\frac{G_\Lambda}{M^n} \cdot \frac{M_1}{8\,\Lambda} \cdot \frac{\Lambda^2}{M_1^2 M_2} \leq \frac{M + |\mathcal{B}| - 1}{M|\mathcal{B}|} \quad ;$$

and as a consequence

$$G_\Lambda \leq \frac{8}{\Lambda} \cdot \frac{M + |\mathcal{B}|}{|\mathcal{B}|} \cdot M^n \ .$$

Moreover, we can check with our concrete algorithm that we always have $|\mathcal{B}| \geq M$ for the construction of the lists $\mathbb{K}_\chi$. Thus we have $G_\Lambda \leq (16/\Lambda)M^n$. For a knapsack which is not one of the G_Λ knapsacks above and for most values[4] of σ_L mod M_1, the size of $\mathbb{K}_\chi$ is smaller than Λ^2 times the expected value $|\mathcal{B}|^2/(M_1^2\, M_2)$, that is,

$$|\mathbb{K}_\chi| \leq \frac{\Lambda^2\, |\mathcal{B}|^2}{M_1^2 \cdot M_2} \ .$$

Note, that this bound includes the case $|\mathbb{K}_0|$.

3.6 Provable Variant of the Concrete Algorithm

Following the ideas presented in Sect. 3.4, we can now describe a variant of our concrete algorithm with provable probabilistic run-time and space requirements. First, fix a large enough value of Λ. We redefine the notion of a "bad" knapsack in this section, by saying that a knapsack is bad if it fails to fulfill one of the three criteria developed in Sect. 3.4 and Sect. 3.5. That is, if there are too many values that yield incorrect splits or lists of type $\mathbb{L}$ or $\mathbb{K}$ which are too large. We find that the total fraction of bad knapsacks is smaller than

$$7\left(\frac{1}{\Lambda - 1} + \frac{2\,\Lambda}{(\Lambda - 1)^2} + \frac{16}{\Lambda}\right) \leq \frac{140}{\Lambda} \quad \text{for } \Lambda \geq 7 \ .$$

By choosing a large enough value for Λ, this fraction can become arbitrarily small.

Once again, we consider a variation of the concrete algorithm where at each level we repeat the choice of random numbers often enough to be successful. For a "good" knapsack there are three ways a decomposition can fail . Firstly, we could choose a random value which does not permit a decomposition of the golden solution; Secondly, we could choose a random value which makes $\mathbb{L}_\chi$ overflow; Thirdly, we could choose a random value which makes $\mathbb{K}_\chi$ overflow. Note that the last two events can be detected, in which case we erase the lists that have been constructed for this random value and turn to the next. For each modulus, the proportion of random values which are incorrect with respect to at least one criteria is smaller than

$$\frac{\Lambda - 1}{\Lambda} + \frac{1}{2\,\Lambda} + \frac{2}{8\,\Lambda} = 1 - \frac{1}{4\,\Lambda} \ .$$

Thus by repeating each split 8Λ times, we make sure that the probability of failure of a given split is at most e^{-2}. Once again, this yields a global probability of success of 5%, which becomes exponentially close to 1 by repeating polynomially many times. Given a real $\varepsilon > 0$, by setting $\Lambda = 2^{\varepsilon\, n}$ we obtain the following theorem:

[4] For all but at most $2M_1/(8\,\Lambda)$ – the factor 2 in the numerator comes from the fact that there are two terms to bound in ((4)).

Theorem 2. *For any real $\varepsilon > 0$ and for a fraction of at least $1 - 140 \cdot 2^{-\varepsilon n}$ of equibalanced knapsacks with density $D < 1$ given by an n-tuple $(a_1, \cdots, a_n)$ and a target value S, if $\epsilon = (\epsilon_1, \cdots, \epsilon_n)$ is a solution of the knapsack then the algorithm described in Sect. 3.3 modified as above finds the solution ϵ sought after in time $\tilde{\mathcal{O}}(2^{(0.291+3\varepsilon)\, n})$.*

We recall that in the theorem, the term *equibalanced* means that the solution ϵ contains exactly the same number of 0s and 1s.

4 Memory Complexity Improvement

In this section we first show a new algorithm of constant memory requirement and running time $\tilde{\mathcal{O}}(2^{3n/4})$. We then show how to decrease its time complexity down to $\tilde{\mathcal{O}}(2^{0.72n})$ using a technique similar to Howgrave-Graham and Joux [5]. Finally, we show a time memory tradeoff for Schroeppel-Shamir's algorithm down to $\tilde{\mathcal{O}}(2^{n/16})$ memory.

4.1 An Algorithm with Running Time $\tilde{\mathcal{O}}(2^{3n/4})$ and Memory $\tilde{\mathcal{O}}(1)$

We describe a simple algorithm that solves the knapsack problem in time $\tilde{\mathcal{O}}(2^{3n/4})$ and constant memory, using a meet-in-the-middle attack. This is done by formulating the meet-in-the-middle attack as a collision search problem (see [15]); then a constant memory cycle-finding algorithm can be used.

We define two functions $f_1, f_2 : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$:

$$f_1(x) = \sum_{i=1}^{n/2} a_i x_i \mod 2^{n/2}, \quad f_2(y) = S - \sum_{i=n/2+1}^{n} a_i y_i \mod 2^{n/2}$$

where x_i denotes the i-th bit of x, and similarly for y_i. If we can find $x, y \in \{0,1\}^{n/2}$ such that $f_1(x) = f_2(y)$, then we get:

$$\sum_{i=1}^{n/2} a_i x_i + \sum_{i=n/2+1}^{n} a_i y_i = S \mod 2^{n/2} \ .$$

This gives a solution of the knapsack problem that is only valid modulo $2^{n/2}$. Since there are heuristically $\tilde{\mathcal{O}}(2^{n/2})$ such solutions holding modulo $2^{n/2}$, and only a single one that holds over $\mathbb{Z}$, a random (x, y) such that $f_1(x) = f_2(y)$ leads to the correct knapsack solution with probability roughly $2^{-n/2}$. Below we show that we can generate such random solution in time $\tilde{\mathcal{O}}(2^{n/4})$ and constant memory. This gives an algorithm with total running time $\tilde{\mathcal{O}}(2^{3n/4})$ and constant memory.

From the two functions f_1, f_2 we define the function $f : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$ where:

$$f(x) = \begin{cases} f_1(x) & \text{if } g(x) = 0 \\ f_2(x) & \text{if } g(x) = 1 \end{cases}$$

where $g : \{0,1\}^{n/2} \to \{0,1\}$ is a random bit function. Then a collision $f(x) = f(y)$ for f gives a desired collision $f_1(x) = f_2(y)$ with probability $1/2$. The function $f : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$ is a random function, therefore using Floyd's cycle finding algorithm [6] we can find a collision for f in time $2^{n/4}$ and constant memory.

However we need to obtain a random collision whereas Floyd's cycle finding algorithm is likely to produce always the same collision. A simple solution is to further randomize the function f; more precisely we apply Floyd' cycle-finding algorithm to $f' : \{0,1\}^{n/2} \to \{0,1\}^{n/2}$ with $f'(x) = P(f(x))$, where P is a random permutation in $\{0,1\}^{n/2}$. Then a new permutation P is used every time a new collision (x, y) is required for f.

4.2 An Algorithm with Running Time $\tilde{\mathcal{O}}(2^{0.72n})$ and Memory $\tilde{\mathcal{O}}(1)$

In this section we show how to slightly decrease the running time down to $\tilde{\mathcal{O}}(2^{0.72n})$, still with constant memory; for this we use the Howgrave-Graham–Joux technique recalled in Sect. 2.1. Again for simplicity we assume that n is a multiple of 4, and that the Hamming weight of the knapsack solution ϵ is exactly $n/2$. As in (3) we write S as the sum $\sigma_1 + \sigma_2$ of two subknapsacks with Hamming weight $n/4$ chosen among the n knapsack elements.

We let W be the set of n-bit strings of Hamming weight $n/4$. We have $\#W = 2^{h(1/4)} \simeq 2^{0.81n}$. We define the two functions $f_1, f_2 : W \to \{0,1\}^{h(1/4)n}$:

$$f_1(y) = \sum_{i=1}^{n} a_i y_i \mod 2^{h(1/4)n}, \quad f_2(z) = S - \sum_{i=1}^{n} a_i z_i \mod 2^{h(1/4)n}$$

where y_i denotes the i-th bit of y, and similarly for z_i. We consider $y, z \in W$ such that:

$$f_1(y) = f_2(z) \ . \tag{5}$$

Since f_1 and f_2 are random functions heuristically there are $2^{h(1/4)n}$ solutions to (5). Moreover given the correct solution ϵ of the knapsack, as in Sect. 2.1 there are $\binom{n/2}{n/4} \simeq 2^{n/2}$ ways of writing this correct solution as $\sigma_1 + \sigma_2 = S$ (see (3)). All these $2^{n/2}$ solutions are solutions of (5). Therefore the probability p that a random solution of (5) leads to the correct knapsack solution is:

$$p = \frac{2^{n/2}}{2^{h(1/4)n}} \simeq 2^{-.31n} \ .$$

The input space of f_1, f_2 has size $2^{h(1/4)n}$. Therefore using the same cycle-finding algorithm as in the previous section, a random solution of (5) can be found in time $\tilde{\mathcal{O}}(2^{h(1/4)n/2})$. The total time complexity is therefore:

$$\begin{aligned} \tilde{\mathcal{O}}(2^{h(1/4)n/2})/p &= \tilde{\mathcal{O}}(2^{h(1/4)n/2}) \cdot 2^{(h(1/4)-1/2)n} \\ &= \tilde{\mathcal{O}}(2^{(3h(1/4)/2-1/2)n}) = \tilde{\mathcal{O}}(2^{.72n}) \ . \end{aligned}$$

Finally, we note that it is possible to further improve this complexity by adding -1s in the decomposition (as in Sect. 3) but the time complexity improvement is almost negligible.

4.3 A Time-Memory Tradeoff on Schroeppel-Shamir Down to $2^{n/16}$ Memory

The original Schroeppel-Shamir algorithm works in time $\tilde{\mathcal{O}}(2^{n/2})$ and memory $\tilde{\mathcal{O}}(2^{n/4})$. In this section we describe a continuous time-memory tradeoff down to $\tilde{\mathcal{O}}(2^{n/16})$ memory. That is we describe a variant of Schroeppel-Shamir with:

$$\text{Running time: } \tilde{\mathcal{O}}(2^{(11/16-\varepsilon)n}), \quad \text{Memory: } \tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$$

for any $0 \leq \epsilon \leq 3/16$. For simplicity we first describe the algorithm with exactly $\tilde{\mathcal{O}}(2^{n/16})$ memory. We write the knapsack as $S = \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4$ as in (2) where each σ_i is a knapsack of $n/4$ elements.

We guess three values R_1, R_2 and R_3 of $3n/16$-bit each and we let R_4 such that $R_1 + R_2 + R_3 + R_4 = S \mod 2^{3n/16}$. We consider the four subknapsack equations

$$\sigma_i = R_i \mod 2^{3n/16} \ .$$

We solve these four equations independently by using the original Schroeppel-Shamir algorithm. Therefore in time $\tilde{\mathcal{O}}(2^{n/8})$ and memory $\tilde{\mathcal{O}}(2^{n/16})$ we obtain four lists $\{\sigma_1\}$, $\{\sigma_2\}$, $\{\sigma_3\}$ and $\{\sigma_4\}$ satisfying the four equations. Eventually to recover the knapsack solution we merge these four lists using the same merging procedure as in the original Schroeppel-Shamir algorithm; since each list has size $\tilde{\mathcal{O}}(2^{n/16})$, the merging procedure runs in time $\tilde{\mathcal{O}}(2^{n/8})$ and memory $\tilde{\mathcal{O}}(2^{n/16})$. Since we have guessed three values of $3n/16$-bit each, the total running time is:

$$\tilde{\mathcal{O}}(2^{3n/16})^3 \cdot \left(\tilde{\mathcal{O}}(2^{n/8}) + \tilde{\mathcal{O}}(2^{n/8})\right) = \tilde{\mathcal{O}}(2^{11n/16})$$

as required, and the memory consumption is $\tilde{\mathcal{O}}(2^{n/16})$.

It is easy to generalize the previous algorithm to memory $\tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$ for any $0 \leq \epsilon < 3/16$. For this we take the R_i's of size $(3/16 - \varepsilon)n$-bit each. We can still build the four lists $\{\sigma_i\}$ in time $\tilde{\mathcal{O}}(2^{n/8})$ using Schroeppel-Shamir, but this time the size of the lists is $\tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$, therefore it requires $\tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$ memory. The merging procedure now runs in time $\tilde{\mathcal{O}}(2^{(1/8+2\varepsilon)n})$, still with memory $\tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$. Therefore the total running time is:

$$\tilde{\mathcal{O}}(2^{(3/16-\varepsilon)n})^3 \cdot \left(\tilde{\mathcal{O}}(2^{n/8}) + \tilde{\mathcal{O}}(2^{(1/8+2\varepsilon)n})\right) = \tilde{\mathcal{O}}(2^{(11/16-\varepsilon)n})$$

as required, for a memory consumption $\tilde{\mathcal{O}}(2^{(1/16+\varepsilon)n})$.

Surprisingly there remains a gap between our variant of Schroeppel-Shamir with $\tilde{\mathcal{O}}(2^{n/16})$ memory and our constant memory algorithm from Sect. 4.1. Namely we were unable to find a variant of Schroeppel-Shamir requiring less than $\tilde{\mathcal{O}}(2^{n/16})$ memory, nor a cycle-based algorithm requiring more than $\tilde{\mathcal{O}}(1)$ memory.

5 Implementation and Experimental Evidence

In order to verify the correctness of the algorithm presented in Sect. 3.3, we have implemented it. We ran our implementation on 50 random knapsacks containing

Table 1. Experimental versus theoretical sizes of the intermediate lists

List type	Min. size	Max. size	Theoretical estimate
$\mathbb{L}_\omega$	12 024 816	12 056 576	$L_\omega = \frac{\binom{80}{6,1,73}}{1\,847} \approx 12\,039\,532$
$\mathbb{K}_\kappa$	61 487 864	61 725 556	$\frac{L_\omega^2}{2\,352\,689} \approx 61\,610\,489$
$\mathbb{L}_\kappa$	12 473 460	20 224 325	$L_\kappa = \frac{\binom{80}{12,2,66}}{1\,847 \cdot 2\,352\,689} \approx 31\,583\,129$
$\mathbb{K}_\nu$	14 409 247	23 453 644	$\frac{L_\kappa^2}{17\,394\,593} \approx 57\,345\,064$
$\mathbb{L}_\nu$	183 447	268 964	$L_\nu = \frac{\binom{80}{22,2,56}}{1\,847 \cdot 2\,352\,689 \cdot 17\,394\,593} \approx 592\,402$
$\mathbb{K}_0$	178	1 090	$\frac{L_\nu^2 \cdot 1\,847 \cdot 2\,352\,689 \cdot 17\,394\,593}{2^{80}} \approx 21\,942$

80 elements on 80 bits. The target sum was constructed in each case as a sum of 40 knapsack elements. For each of these knapsacks, we ran our implementation several times, choosing new random modular constraints for each execution, until a solution was found. We added two -1s at the first level, one -1 at the second and none at the third level. At the same time, we collected some statistics about the behavior of our code.

The total running time to solve the 50 knapsacks was 14 hours and 50 minutes on a Intel® Core™ i7 CPU M 620 at 2.67GHz. The total number of repetitions of the basic algorithm was equal to 280. We observed that a maximum of 16 repetitions (choice of a different random value in level ν) was sufficient to find the solution. Also, 53% of the 50 random knapsacks needed only up to 4 repetitions. On average, each knapsack required 5.6 repetitions.

During the execution of the 280 repetitions of the basic algorithm, we also noted the length of the lists $\mathbb{L}$ and $\mathbb{K}$ (still containing inconsistent solutions) that occurred at each level. The moduli were chosen as primes of size as discussed in Sect. 3.3: $M_\omega = 1\,847$, $M_\kappa = 2\,353\,689$, and $M_\nu = 17\,394\,593$. The experimental and theoretical list sizes are given in Table 1.

We see in Table 1 that the sizes of $\mathbb{L}_\omega$ and $\mathbb{K}_\kappa$ are very close to the predicted values and do not vary a lot. We refer to the full version of our paper [2] for a more detailed discussion; see also [2] for implementation results on 96 bits.

Acknowledgments. We would like to thank Alexander May and Alexander Meurer for pointing out the inconsistency issue in Howgrave-Graham–Joux algorithm. We also thank Dan Bernstein for helpful comments on a preliminary version of this work.

References

1. Ajtai, M.: The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In: STOC 1998, pp. 10–19 (1998)
2. Becker, A., Coron, J.-S., Joux, A.: Improved generic algorithms for hard knapsacks. Eprint archive (2011)
3. Coster, M.J., Joux, A., LaMacchia, B.A., Odlyzko, A.M., Schnorr, C.-P., Stern, J.: Improved low-density subset sum algorithms. Computational Complexity 2, 111–128 (1992)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, New York (1979)
5. Howgrave-Graham, N., Joux, A.: New generic algorithms for hard knapsacks. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 235–256. Springer, Heidelberg (2010)
6. Knuth, D.E.: The Art of Computer Programming, 2nd edn. Seminumerical Algorithms, vol. II. Addison-Wesley, Reading (1981)
7. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. J. ACM 32(1), 229–246 (1985)
8. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 515–534 (1982)
9. May, A., Meurer, A.: Personal communication
10. Merkle, R.C., Hellman, M.E.: Hiding information and signatures in trapdoor knapsacks. IEEE Transactions on Information Theory 24, 525–530 (1978)
11. Nguyen, P.Q., Shparlinski, I.E., Stern, J.: Distribution of modular sums and the security of the server aided exponentiation. In: Progress in Computer Science and Applied Logic, Final Proceedings of Cryptography and Computational Number Theory Workshop, Singapore, vol. 20, pp. 331–224 (2001)
12. Schnorr, C.-P.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. 53, 201–224 (1987)
13. Schroeppel, R., Shamir, A.: A $T = O(2^{n/2}), S = O(2^{n/4})$ algorithm for certain NP-complete problems. SIAM J. Comput. 10(3), 456–464 (1981)
14. Shamir, A.: A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. In: CRYPTO 1982, pp. 279–288 (1982)
15. van Oorschot, P.C., Wiener, M.J.: Improving implementable meet-in-the-middle attacks by orders of magnitude. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 229–236. Springer, Heidelberg (1996)
16. Wagner, D.: A generalized birthday problem. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 288–303. Springer, Heidelberg (2002)

Two-Output Secure Computation with Malicious Adversaries

Abhi Shelat and Chih-Hao Shen

University of Virginia, Charlottesville, VA 22904
{shelat,shench}@virginia.edu

Abstract. We present a method to compile Yao's two-player garbled circuit protocol into one that is secure against malicious adversaries that relies on witness indistinguishability. Our approach can enjoy lower communication and computation overhead than methods based on cut-and-choose [13] and lower overhead than methods based on zero-knowledge proofs [8] (or Σ-protocols [14]). To do so, we develop and analyze new solutions to issues arising with this transformation:

— How to guarantee the generator's input consistency
— How to support different outputs for each player *without* adding extra gates to the circuit of the function f being computed
— How the evaluator can retrieve input keys but avoid selective failure attacks
— Challenging 3/5 of the circuits is near optimal for cut-and-choose (and better than challenging 1/2)

Our protocols require the existence of secure-OT and claw-free functions that have a weak malleability property. We discuss an experimental implementation of our protocol to validate our efficiency claims.

Keywords: Witness indistiguishability, Yao garbled circuits, signature schemes.

1 Introduction

Yao [23] proposed a method that allows two *honest-but-curious* players—a *generator* (denoted by P_1) with secret input x, and an *evaluator* (denoted by P_2) with secret input y—to jointly compute a function $f(x, y)$ such that P_1 receives nothing and P_2 receives $f(x, y)$[1]. In this paper, we propose an approach for transforming Yao's garbled circuit protocol for honest-but-curious players into a protocol that is secure against *malicious* players. Our main goal is to improve the efficiency of this transformation and to do so using more general assumptions.

There are two well-known methods to achieve this transformation: the *commit-and-prove* and *cut-and-choose*. The commit-and-prove method suggested by Goldreich, Micali, and Widgerson [6] only requires the weak general assumption of

[1] A thorough description of this protocol can be found in Lindell and Pinkas [13].

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 386–405, 2011.

zero-knowledge proofs of knowledge. However, this approach requires costly NP-reductions, which have never been implemented. On the other hand, an efficient transformation based on the cut-and-choose method was recently proposed by Lindell and Pinkas [13] and implemented by Pinkas et al. [20]. The general idea in cut-and-choose is for P_1 to prepare multiple copies of the circuit to be evaluated. A randomly selected set of the circuits (called *check-circuits*) are then opened to show if they were constructed correctly. Finally, the unopened circuits (called *evaluation-circuits*) are evaluated by P_2 and the majority of the results is taken as the final output. This approach has only constant round complexity, but the replication incurs both communicational and computational overhead.

The starting point for our work is the cut-and-choose method. *A natural question we aim to study is to understand the fundamental limitations (in terms of efficiency) of the cut-and-choose method.* This method does not require NP-reductions; however, it faces other efficiency problems stemming from the new security problems introduced by evaluating e out of s copies of the circuit. In this paper, we address several of these issues: (1) ensuring input consistency, (2) handling two-output functions, (3) preventing selective failure attacks, and (4) determining the optimal number of circuits to open versus evaluate. Moreover, we identify weak and generic properties that admit efficient solutions to these issues. In several of the cases, using witness indistinguishable protocols suffice. Thus, in the case of input consistency, we are able to use an extremely efficient protocol as long as claw-free functions with a minimal malleability property exist (they do under the standard algebraic assumptions). We will later demonstrate the benefits of our approach by both asymptotic analysis of complexity and experimental results from an implementation. We now give an overview of our contributions.

1.1 Generator's Input Consistency

According to the cut-and-choose method, P_1 needs to send e copies of her garbled input to P_2. Since the circuits are *garbled*, P_1 could cheat by sending different inputs for the e copies of the garbled circuit. For certain functions, there are simple ways for P_1 to extract information about P_2's input (§ 3 of [13]). Therefore, the protocol must ensure that all e copies of P_1's input are *consistent*.

Related work. Let n be P_1's and P_2's input size, and let s be a statistical security parameter for the cut-and-choose method. Mohassel and Franklin [16] proposed the *equality-checker* scheme, which has $O(ns^2)$ computation and communication complexity. Woodruff [22] later suggested an *expander-graph* framework to give a sharper bound to P_1's cheating probability. The asymptotic complexity is $O(ns)$, however, in practice, the constant needed to construct the expander graphs is prohibitively large. Lindell and Pinkas [13] develop an elegant cut-and-choose based construction that enjoys the simulation-based security against malicious players. This approach requires $O(ns^2)$ commitments to be computed and exchanged between the participants. Although these commitments can be implemented using lightweight primitives such as collision-resistant hash functions, communication complexity is still an issue. Jarecki and Shmatikov [8] presented

an approach that is based on commit-and-prove method. Although only a single circuit is constructed, their protocol requires hundreds of heavy cryptographic operations *per gate*, whereas approaches based on the cut-and-choose method require only such expensive operations for the *input gates*. Nielsen and Orlandi [18] proposed an approach with Lego-like garbled gates. Although it is also based on the cut-and-choose method, via an alignment technique only a single copy of P_1's input keys is needed for all the e copies of the garbled circuit. However, similar to Jarecki and Shmatikov's approach, each gate needs several group elements as commitments resulting both computational and communicational overhead. Lindell and Pinkas propose a Diffie-Hellman pseudorandom synthesizer technique in [14]; their approach relies on finding efficient zero-knowledge proofs for specifically chosen complexity assumptions, which is of complexity $O(ns)$.

Our approach to consistency. We solve this problem not by explicitly using zero-knowledge protocols (or Σ-protocols) but by communicating merely $O(ns)$ group elements. Our novel approach is to first observe that witness indistinguishable proofs suffice for consistency, and to then use *claw-free functions*[2] that have a weak malleability property to generate efficient instantiations of such proofs.

Intuitively, P_1's input is encoded using elements from the domain of the claw-free collections which can later be used to prove their consistency among circuits. The elements are hashed into random bit-strings which P_1 uses to construct keys for garbled input gates. The rest of the gates in the circuit use fast symmetric operations as per prior work. A concrete example is to instantiate the claw-free functions under the Discrete Logarithm assumption by letting $f_b(m) = g^b h^m$ for some primes p and q such that $p = 2q + 1$, and distinct group elements g and h of $\mathbb{Z}_p^*$ such that $\langle g \rangle = \langle h \rangle = q$. It is well-known that such a pair of functions have efficient zero-knowledge proofs. An example instantiation of our solution built on this pair of claw-free functions works as follows: P_1 samples $[m_{0,1}, \ldots, m_{0,s}]$ and $[m_{1,1}, \ldots, m_{1,s}]$ from f_0 and f_1's domain $\mathbb{Z}_q$. The range elements $[h^{m_{0,1}}, \ldots, h^{m_{0,s}}]$ and $[gh^{m_{1,1}}, \ldots, gh^{m_{1,s}}]$ are then used to construct garbled circuits in the way that $g^b h^{m_{b,j}}$ is associated with P_1's input bit value b in the j-th garbled circuit. The cut-and-choose method verifies that the majority of the evaluation-circuits are correctly constructed. Let $[j_1, \ldots, j_e]$ be the indices of these evaluation-circuits. At the onset of the evaluation phase, P_1 with input bit x reveals $[g^x h^{m_{x,j_1}}, \ldots, g^x h^{m_{x,j_e}}]$ to P_2 and then proves that these range elements are the commitments of the same bit x. Intuitively, by the identical range distribution property, P_2 with $f_x(m_{x,i})$ at hand has no information about x. Furthermore, after P_1 proves the knowledge of the pre-image of $[f_x(m_{x,j_1}), \ldots, f_x(m_{x,j_e})]$ under the same f_x, by the claw-free property, P_1 proves the consistency of his input keys for all the evaluation-circuits.

Furthermore, in the course of developing our proof, we noticed that witness indistinguishable proofs suffice in place of zero-knowledge proofs. Even more

[2] Loosely speaking, a pair of functions (f_0, f_1) are said to be claw-free if they are (1) easy to evaluate, (2) identically distributed over the same range, and (3) hard to find a *claw*. A claw is a pair of elements, one from f_0's domain and the other from f_1's domain, that are mapped to the same range element.

generally, when the claw-free collection has a very weak malleability property (which holds for all known concrete instantiations), sending a simple function of the witness itself suffices. We will get into more details in §2.1.

It is noteworthy that both the committed-input scheme in [16] and Diffie-Hellman pseudorandom synthesizer technique in [14] are special cases of our approach, and thus, have similar complexity. However, the committed-input scheme is not known to enjoy simulation-based security, and the pseudorandom synthesizer technique requires zero-knowledge proofs that are unnecessary in our case, which means that our approach is faster by a constant factor in practice.

1.2 Two-Output Functions

It is not uncommon that *both* P_1 and P_2 need to receive outputs from a secure computation, that is, the goal function is $f(x, y) = (f_1, f_2)$ such that P_1 with input x gets output f_1, and P_2 with input y gets f_2[3]. In this case, the security requires that *both* the input and output are hidden from the other player. When both players are honest-but-curious, a straightforward solution is to let P_1 choose a random number c as an extra input, convert $f(x, y) = (f_1, f_2)$ into a new function $f^*((x, c), y) = (\lambda, (f_1 \oplus c, f_2))$, run the original Yao protocol for f^*, and instruct P_2 to pass the encrypted output $f_1 \oplus c$ back to P_1, who can then retrieve her real output f_1 with the secret input c chosen in the first place. However, the situation gets complicated when either of the players could potentially be malicious. Note that the two-output protocols we consider are not *fair* since P_2 may always learn its own output and refuse to send P_1's output. However, they can satisfy the notion that if P_1 accepts output, it will be correctly computed.

Related work. One straightforward solution is for the players to run the single-output protocol twice with roles reversed. Care must be taken to ensure that the same inputs are used in both executions. Also, this approach doubles the computation and communication cost. Other simple methods to handle two-output functions also have subtle problems. Suppose, for example, P_1 encrypts all copies of her output and has P_2 send these s random strings (or encryptions) in the last message. In a cut-and-choose framework, however, a cheating P_1 can use these random strings to send back information about the internal state of the computation and thereby violate P_2's privacy. As an example, the cheating P_1 can make one bad circuit in which P_1's output bit is equal to P_2's first input bit. If P_2 sends all copies of P_1's output bit back to P_1, then with noticeable probability, the cheating P_1 can learn P_2's first input bit. The problem remains if instead of sending back all bits, only a randomly chosen output bit is sent. Besides, P_1 should not be convinced by a cheating P_2 with an arbitrary output.

As described in [13], the two-output case can be reduced to the single-output case as follows: (1) P_1 randomly samples $a, b, c \in \{0, 1\}^n$ as extra input; (2) the original function is converted into $f^*((x, a, b, c), y) = (\lambda, (\alpha, \beta, f_2))$ where $\alpha = f_1 \oplus c$ is an encryption of f_1 and $\beta = a \cdot \alpha + b$ is the Message Authentication code (MAC) of α, and (3) P_2 sends (α, β) back to P_1, who can then check the

[3] Here f_1 and f_2 are abbreviations of $f_1(x, y)$ and $f_2(x, y)$ for simplicity purpose.

authenticity of the output $\alpha = f_1 \oplus c$. However, this transformation increases the size of P_1's input from n bits to $4n$ bits. As a result, the complexity of P_1's input consistency check is also increased. A second drawback is that the circuit must also be modified to include extra gates for computing the encryption and MAC function. Although a recent technique [12] can be used to implement XOR gates "for free," the MAC function $a \cdot \alpha + b$ still requires approximately $O(n^2)$ extra gates added to the circuit. Since all s copies of the circuit have to be modified, this results in additional communication of $O(sn^2)$ encrypted gates. Indeed, for simple functions, the size of this overhead exceeds the size of the original circuit.

Kiraz and Schoenmakers [11] present a fair two-party computation protocol in which a similar issue for two-output functions arises. In their approach, P_2 commits to P_1's garbled output. Then P_1 reveals the two output keys for each of her output wires, and P_2 finds one circuit GC_r which agrees with "the majority output for P_1." The index r is then revealed to P_1. However, informing P_1 the index of the majority circuit could possibly leak information about P_2's input. As an anonymous reviewer has brought to our attention an unpublished follow-up work from Kiraz [9], which elaborated this issue (in § 6.6 of [9]) and further fixed the problem without affecting the overall performance. Particularly, in the new solution, the dominant computational overhead is an OR-proof of size $O(s)$, and the dominant communicational overhead is the commitments to P_1 output keys, where the number of such commmitments is of order $O(ns)$. Their techniques favorably compare to our approach, but we do not have experimental data to make accurate comparisons with our implementation.

Our approach to two-output functions. We present a method to evaluate two-output function f without adding non-XOR gates to the original circuit for f.

In order for P_2 to choose one output that agrees with the majority, similar to Kiraz and Schoenmakers' approach in [11], we add extra bits to P_1's input as a one-time pad encryption key by changing the function from $f(x, y) = (f_1, f_2)$ to $f^*((c, x), y) = (\lambda, (f_1 \oplus c, f_2))$, where $x, c, y, f_1, f_2 \in \{0, 1\}^n$. With this extra random input c from P_1, P_2 is able to do the majority function on the evaluation output $f_1 \oplus c$ without knowing P_1's real output f_1. Next, P_2 needs to prove the authenticity of the evaluation output $f_1 \oplus c$ that she has given to P_1. Here, our idea is that P_1's i-th output gate in the j-th garbled circuit is modified to output $0||\sigma_{sk}(0, i, j)$ or $1||\sigma_{sk}(1, i, j)$ instead of 0 or 1, where $\sigma_{sk}(b, i, j)$ is a signature of the message (b, i, j) signed by P_1 under the signing key sk. In other words, the garbled gate outputs P_1's output bit b and a signature of b, bit index i, and circuit index j. Therefore, after the circuit evaluation, P_2 hands $f_1 \oplus c$ to P_1 and proves the knowledge of the signature of each bit under the condition that the j-index for all signatures are the same and valid (among the indices of the evaluation-circuits). Naively, this proof would have been a proof of $O(ns)$ group elements. However, we will show that a witness indistinguishable proof suffices, which reduces the complexity by a constant factor. Furthermore, by using the technique of Camenisch, Chaabouni, and Shelat for efficient set membership proof [4], we are able to reduce the complexity to $O(n + s)$ group elements.

1.3 The Problem of Selective Failure

Another problem with compiling garbled circuits occurs during the Oblivious Transfer (OT) phase, when P_2 retrieves input keys for the garbled circuits. A malicious P_1 can attack the protocol with *selective failure*, where the keys used to construct the garbled circuit might not be the ones used in the OT so that P_2's input can be inferred according to her reaction after OT. For example, a cheating P_1 could use (K_0, K_1) to construct a garbled circuit but use (K_0, K_1^*) instead in the corresponding OT, where $K_1 \neq K_1^*$. As a result, if P_2's input bit is 1, she will get K_1^* after OT and cannot evaluate the garbled circuit properly. In contrast, if her input bit is 0, P_2 will get K_0 from OT and complete the evaluation without complaints. P_1 can therefore infer P_2's input. This issue is identified by both Mohassel and Franklin [16] and Kiraz and Schoenmakers [10].

Related work. Lindell and Pinkas [13] replace each of P_2's input bits with s additional input bits. These s new bits are XOR'ed together, and the result is used as the input to the original circuit. Such an approach makes the probability that P_2 must abort due to selective failure independent of her input. This approach, however, increases the number of input bits for P_2 from n to ns. Woodruff later pointed out that the use of clever coding system can reduce the overhead to $\max(4n, 8s)$. To be sure, Lindell, Pinkas, and Smart [15] implement the method described in [13] and empirically confirm the extra overhead from this step. In particular, a 16-bit comparison circuit that originally needs fifteen 3-to-1 gates and one 2-to-1 gate will be inflated to a circuit of several thousand gates after increasing the number of inputs. Since the number of inputs determines the number of OT operations, an approach that keeps the number of extra inputs small is preferable. In fact, we show that increasing the number of inputs and number of gates in the circuit for this problem is unnecessary.

Independent of our work, Lindell and Pinkas [14] propose to solve this problem by *cut-and-choose OT*. This new solution indeed provides a great improvement over [13] and shares roughly the same complexity with our solution. Furthermore, both the cut-and-choose OT and our solution can be built upon the efficient OT proposed by Naor and Pinkas [17] or Peikert, Vaikuntanathan, and Waters [19]. However, the particular use the latter OT in [14] needs two independently chosen common reference strings, while our solution needs only one.

Our approach to selective failure. Inspired by the idea of *committing Oblivious Transfer* proposed by Kiraz and Schoenmakers [10], we solve the problem of selective failure by having the sender (P_1 in Yao protocol) of the OT *post-facto* prove that she ran the OT correctly by revealing the randomness used in the OT. Normally, this would break the sender-security of the OT. However, in a cut-and-choose framework, the sender is already opening many circuits, so the keys used as inputs for the OT are no longer secret. Thus, the idea is that the sender can prove that he executed the OT correctly for all circuits that are opened by simply sending the random coins used in the OT protocol for those instances. We stress that not every OT can be used here. Intuitively, a committing OT

is the OT with the binding property so that it is hard for a cheating sender to produce random coins different from what she really used.

A critical point with this approach is that in order to simulate a malicious P_2, we need to use a coin-flipping protocol to pick which circuits to open. Consequently, P_1 cannot open the circuits to P_2 until the coin-flipping is over; yet the OT must be done before the coin-flipping in order to guarantee a proper *cut*. So the order of operations of the protocol is critical to security. An efficient committing OT based on Decisional Diffie-Hellman problem is presented in §2.3.

1.4 Optimal Cut-and-Choose Strategy

We find that most cut-and-choose protocols open $s/2$ out of the s copies of the garbled circuit to reduce the probability that P_1 succeeds in cheating. We show that opening $3s/5$-out-of-s is a better choice than $s/2$-out-of-s. In particular, when s circuits are used, our strategy results in security level $2^{-0.32s}$ in contrast to $2^{-s/17}$ from [13] and $2^{-0.31s}$ from [14]. Although the difference with the latter work is only 1% less, we show the optimal parameters for the cut-and-choose method in Appendix A, thereby establishing a close characterization of the limits of the cut-and-choose method.

1.5 Comparison of Communication Complexity

We attempt to compare communication efficiency between protocols that use a mix of light cryptographic primitives (such as commitments instantiated with collision-resistant hash functions) and heavy ones (such as group operations that rely on algebraic assumptions like discrete logarithm). To meaningfully do so, we consider asymptotic security under reasonable assumptions about the growth of various primitives with respect to the security parameter k. We assume that:

1. light cryptographic primitives have size $\Theta(k)$;
2. heavy cryptographic operations that can be instantiated with elliptic curves or bilinear groups take size $\tilde{o}(k^2)$.
3. heavy cryptographic operations that require RSA or prime order groups over $\mathbb{Z}$ take size $\tilde{o}(k^3)$.

The size assumption we make is quite conservative. It is based on the observation that in certain elliptic curve groups, known methods for computing discrete logarithms of size n run in time $L_n(1, 1/2)$. Thus, to achieve security of 2^k, it suffices to use operands of size $\tilde{o}(k^2)$ by which we mean a value that is asymptotically smaller than k^2 by factors of $\log(k)$. The computation bound follows from the running time analysis of point multiplication (or exponentiation in the case of $\mathbb{Z}_p^*$) algorithms. As we discuss below, for reasonable security parameters, however, the hidden constants in this notation make the difference much smaller. Let k be a security parameter for cryptographic operations, let s be a statistical security parameter, and let $|C|$ be the number of gates in the base circuit computing $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$.

– Jarecki and Shmatikov [8]: For each gate, the number of the communicated group elements is at least 100, including the commitments of the garbled values for input wires, the commitments of the doubly-encrypted entries, and the ZK proof for the correctness of the gate. Moreover, for each input or output wires, a ZK proof for conjunction/disjunction is required. Each of the ZK proofs needs constant number of group elements. Finally, this protocol assumes the decisional composite residuosity problem in an RSA group; thus, each group element is of size $\tilde{o}(k^3)$.
– Kiraz [9]: This approach uses an equality-checker framework that requires $O(ns^2)$ commitments for checking P_1's input consistency. They solve the selective failure attack with committing OT as we do. Moreover, to deal with two-output functions, they add n extra bits to P_1's input, commit to all of P_1's output keys, which include $2ns$ commitments and $2ns$ decommitments, and a zero-knowledge OR-proof of size $O(s)$.
– Lindell and Pinkas [13]: Each of the garbled gates requires $4k$ space for four doubly-encrypted entries. Thus, for this approach, the communication analysis is as follows: (1) s copies of the base circuit itself require $s|C|$ gates; (2) each of P_1's n input bits requires s^2 light commitments for the consistency check; (3) P_2's n input bits require $\max(4n, 8s)$ OT's. Also, the MAC-based two-output function computation add additional $O(n^2)$ gates to each of the s copies of the circuit and additional $3n$ bits to P_1's input. Thus, the overall communication cost to handle two-output function is $O(n^2sk + ns^2k)$.

Table 1. Asymptotic Analysis of various two-party secure computation

	COMMUNICATION			
	Base circuit	P_1's input	P_2's input	Two-output
JS [8]	$\|C\| \cdot \tilde{o}(k^3)$	$n \cdot \tilde{o}(k^3)$	n OT's	$n \cdot \tilde{o}(k^3)$
K [9]	$\Theta(\|C\| \cdot sk)$	$\Theta(ns^2k)$	n OT's	$\Theta(nsk) + \Theta(s) \cdot \tilde{o}(k^2)$
LP07 [13]	$\Theta(\|C\| \cdot sk)$	$\Theta(ns^2k)$	$\max(4n, 8s)$ OT's	$\Theta(n^2sk + ns^2k)$
LP10 [14]	$\Theta(\|C\| \cdot sk)$	$\Theta(ns) \cdot \tilde{o}(k^2)$	n OT's	$\Theta(n^2sk + ns^2k)$
Our work	$\Theta(\|C\| \cdot sk)$	$\Theta(ns) \cdot \tilde{o}(k^2)$	n OT's	$\Theta(ns) \cdot \tilde{o}(k^2)$

The recent work of [14] also considers a more efficient way to implement two-party computation based on cut-and-choose OT and specific security assumptions. They report $13sn$ exponentiations and communication of $5sn + 14k + 7n$ group elements. (Note we count bits above to compare commitments versus other primitives.) Concretely, these parameters are similar to our parameters but rely on more specific assumptions, and do not consider two-party outputs.

2 Building Blocks

For clarity purpose, the standard checks that are required for security have been omitted. For example, in many cases, it is necessary to verify that an element that has been sent is indeed a member of the right group. In some cases,

it is implicit that if a player detectably cheats in a sub-protocol, then the other player would immediately abort execution of the entire protocol.

2.1 Consistency Check for the Generator's Input

The cut-and-choose approach to compiling Yao circuits ensures that P_1 submits consistent input values for each copy of the evaluation-circuits. Recall that there are e copies of the circuit which must be evaluated. Thus, for each input wire, P_1 must send e keys corresponding to an input bit 0 or 1. It has been well-documented [16,10,22,13] that in some circumstances, P_1 can gain information about P_2's input if P_1 is able to submit different input values for the e copies of this input wire. The main idea of our solution is inspired by the *claw-free collections*[4] defined as follows:

Definition 1 (Claw-Free Collections in [7]). *A three-tuple of algorithms (G, D, F) is called a* claw-free collection *if the following conditions hold*

1. ***Easy to evaluate:*** *Both the index selecting algorithm G and the domain sampling algorithm D are probabilistic polynomial-time, while the evaluating algorithm F is a deterministic polynomial-time.*
2. ***Identical range distribution:*** *Let $f_I^b(x)$ denote the output of F on input (b, I, x). For any I in the range of G, the random variable $f_I^0(D(0, I))$ and $f_I^1(D(1, I))$ are identically distributed.*
3. ***Hard to form claws:*** *For every non-uniform probabilistic polynomial-time algorithm A, every polynomial $p(\cdot)$, and every sufficiently large n's, it is true that $\Pr[I \leftarrow G(1^n); (x, y) \leftarrow A(I) : f_I^0(x) = f_I^1(y)] < 1/p(n)$.*

With the claw-free collections, our idea works as follows: P_2 first generates I by invoking the index generating algorithm $G(1^k)$, where k is a security parameter. For each of her input bits, P_1 invokes sampling algorithms $D(I, 0)$ and $D(I, 1)$ to pick $[m_{0,1}, \ldots, m_{0,s}]$ and $[m_{1,1}, \ldots, m_{1,s}]$, respectively. P_1 then constructs s copies of garbled circuit with range elements $[f_I^0(m_{0,1}), \ldots, f_I^0(m_{0,s})]$ and $[f_I^1(m_{1,1}), \ldots, f_I^1(m_{1,s})]$ by associating $f_I^b(m_{b,j})$ with P_1's input wire of bit value b in the j-th garbled circuit. Let $[j_1, \ldots, j_e]$ denote the indices of the garbled circuits not checked in the cut-and-choose (evaluation-circuits). During the evaluation, P_1 reveals $[f_I^b(m_{b,j_1}), \ldots, f_I^b(m_{b,j_e})]$ to P_2 and proves in zero-knowledge that P_1 gets $f_I^b(m_{j_1}^b)$ and $f_I^b(m_{j_i}^b)$ via the same function f_I^b, for $2 \leq i \leq e$.

However, in the course of developing our solution, we noticed that witness indistinguishable proofs suffice in place of zero-knowledge proofs. For example, consider the claw-free collection instantiated from the Discrete Logarithm assumption, that is, let $f_I^b(m) = g^b h^m$, where $I = (g, h, p, q)$ includes two primes p and q such that $p = 2q + 1$, and distinct generators g and h of $\mathbb{Z}_p^*$ such that $\langle g \rangle = \langle h \rangle = q$. After revealing $[g^b h^{m_{b,j_1}}, \ldots, g^b h^{m_{b,j_e}}]$ to P_2, it is a natural solution that P_1 proves in zero-knowledge to P_2 the knowledge of $(m_{b,j_i} - m_{b,j_1})$ given common input $g^b h^{m_{b,j_i}} (g^b h^{m_{b,j_1}})^{-1} = h^{m_{b,j_i} - m_{b,j_1}}$, for $2 \leq i \leq e$.

[4] It is well known that claw-free collections exist under either the Discrete Logarithm assumption or Integer Factorization assumption [7].

The key insight here is that it is unnecessary for P_1 to hide $(m_{b,j_i} - m_{b,j_1})$ from P_2 since $[m_{b,j_1}, \ldots, m_{b,j_e}]$ are new random variables introduced by P_1 and b is the only secret needed to be hidden from P_2. Simply sending $(m_{b,j_i} - m_{b,j_1})$ to P_2 will suffice a proof of checking P_1's input consistency without compromising P_1's privacy. In other words, given $[g^b h^{m_{b,j_1}}, \ldots, g^b h^{m_{b,j_e}}, m'_2, \ldots, m'_e]$, if P_2 confirms that $g^b h^{m_{b,j_1}} = g^b h^{m_{b,j_i}} \cdot h^{m'_i}$ for $2 \leq i \leq e$, then either P_1's input is consistent so that $m'_i = m_{b,j_i} - m_{b,j_1}$, or P_1 is able to come up with a claw.

Note that extra work is only done for the input gates—and moreover, only those of P_1. All of the remaining gates in the circuit are generated as usual, that is, they do not incur extra commitments. So, unlike solutions with committed OT such as [8], asymmetric cryptography is only used for the input gates rather than the entire circuit. To generalize the idea, we introduce the following notion.

Definition 2 (Malleable Claw-Free Collections). *A four-tuple of algorithms* (G, D, F, R) *is a* malleable claw-free collection *if the following conditions hold.*

1. ***A subset of claw-free collections:*** (G, D, F) *is a claw-free collection, and the range of* D *and* F *are groups, denoted by* $(\mathbb{G}_1, \star)$ *and* $(\mathbb{G}_2, \diamond)$ *respectively.*
2. ***Uniform domain sampling:*** *For any* I *in the range of* G*, random variable* $D(0, I)$ *and* $D(1, I)$ *are uniform over* $\mathbb{G}_1$*, and denoted by* $D(I)$ *for simplicity.*
3. ***Malleability:*** $R : \mathbb{G}_1 \to \mathbb{G}_2$ *runs in polynomial time, and for* $b \in \{0, 1\}$*, any* I *in the range of* G*, and any* $m_1, m_2 \in \mathbb{G}_1$*,* $f_I^b(m_1 \star m_2) = f_I^b(m_1) \diamond R_I(m_2)$.

Consider the claw-free collection constructed above under the Discrete Logarithm assumption, we know that it can become a malleable claw-free collection simply by letting $\mathbb{G}_1 = \mathbb{Z}_q$, $\mathbb{G}_2 = \mathbb{Z}_p^*$, and $R_I(m) = h^m$ for any $m \in \mathbb{G}_1$.

2.2 Two-Output Functions

To handle two-output functions, we want to satisfy the notion that it might be unfair in the sense that P_2 could abort prematurely after circuit evaluation and she gets her output. However, if P_1 accepts the output given from P_2, our approach guarantees that this output is genuine. Namely, P_2 cannot provide an arbitrary value to be P_1's output. In particular, P_2 cannot learn P_1's output more than those deduced from P_2's own input and output.

Recall that it is a well-accepted solution to convert the garbled circuit computing $f(x, y) = (f_1, f_2)$ into the one computing $g((x, p, a, b), y) = ((\alpha, \beta), f_2)$, where $\alpha = f_1 + p$ as a ciphertext of f_1 and $\beta = a \cdot \alpha + b$ as a MAC for the ciphertext. Since P_2 only gets the ciphertext of P_1's output, she does not learn anything from the ciphertext. Also, given (α, β), P_1 can easily verify the authenticity of her output. However, we are not satisfied with the additional $O(s^2)$ gates computing the MAC (s is the statistical security parameter) to each of the s copies of the garbled circuit, which results in $O(s^3)$ extra garbled gates in total. Indeed, the number of extra gates can easily exceed the size of the original circuit when f is a simple function. Hence, we propose another approach to authenticate P_1's output without the extra gates computing the MAC function.

While our approach also converts the circuit to output the ciphertext of P_1's output, that is, from $f(x, y) = (f_1, f_2)$ to $f^*((c, x), y) = (\lambda, (f_1 \oplus c, f_2))$, we solve

the authentication problem by the use of the public-key signature scheme and its corresponding witness-indistinguishable proof. Each bit value of the output of P_1's output gates is tied together with a signature specifying the value and the location of the bit. On one hand, P_2 can easily verify the signature during the cut-and-choose phase (to confirm that the circuits are correctly constructed). On the other hand, after the evaluation and giving P_1 the evaluation result $(f_1 \oplus c)$, P_2 can show the authenticity of each bit of the result by proving the knowledge of its signature, that is, the signature of the given bit value from the right bit location. Note that a bit location includes a bit index and a circuit index. In other words, a bit location (i, j) indicates P_1's i-th output bit from the j-th garbled circuit. While the bit index is free to reveal (since P_1 and P_2 have to conduct the proof bit by bit anyway), the circuit index needs to be hidden from P_1; otherwise, P_1 can gain information about P_2's input as we discussed above. We stress that it is critical for P_2 to provide a signature from the right location. Since during the cut-and-choose phase, many properly signed signatures are revealed from the check-circuits, if those signatures do not contain location information, they can be used to convince P_1 to accept arbitrary output.

Normally, an OR-proof will suffice the proof that the signature is from one of the evaluation-circuits. Nevertheless, an OR proof of size $O(s)$ for each bit of P_1's n-bit output will result in a zero-knowledge proof of size $O(ns)$. We therefore adopt the technique from [4] in order to reduce the size of the proof to $O(n+s)$. Let $S = \{j_1, \ldots, j_e\}$ be the indices of all the evaluation-circuits. The idea is for P_1 to send a signature of every element in S, denoted by $[\delta(j_1), \ldots, \delta(j_e)]$. By reusing these signatures, P_2 is able to perform each OR proof in constant communication. More specifically, after the evaluation, P_2 chooses one evaluation-circuit, say the j_l-th circuit, the result of which conforms with the majority of all the evaluation-circuits. Let $\mathcal{M} = [\mathrm{M}_1, \ldots, \mathrm{M}_n]$ be P_1's output from the j_l-th circuit. Recall that P_2 has both M_i and the signature to (M_i, i, j_l), denoted by $\sigma(\mathrm{M}_i, i, j)$, due to the way the garbled circuits were constructed. To prove the authenticity of M_i, P_2 sends M_i to P_1, blinds signature $\delta(j_l)$ and $\sigma(\mathrm{M}_i, i, j_l)$, and proves the knowledge of "$\sigma(\mathrm{M}_i, i, j)$ for some $j \in S$." In other words, P_2 needs to prove the knowledge of $\sigma(\mathrm{M}_i, i, j)$ and $\delta(j^*)$ such that $j = j^*$ for $i = 1, \ldots, n$. The complete proof is shown in Protocol 1. Due to the nonforgeability property of signature schemes, P_2 proves the knowledge of the signature and thus the authenticity of $\mathcal{M}$.

One particular implementation of our protocol can use the Boneh-Boyen short signature scheme [2] which is briefly summarized here. The Boneh-Boyen signature scheme requires the *q-SDH (Strong Diffie-Hellman) assumption*[5] and *bilinear maps*[6]. Based on these two objects, the Boneh-Boyen signature scheme includes a three-tuple of efficient algorithms (G, S, V) such that

[5] q-SDF assumption in a group G of prime order p states that given $g, g^x, g^{x^2}, \ldots, g^{x^q}$, it is infeasible to output a pair $(c, g^{1/(x+c)})$ where $c \in \mathbb{Z}_p^*$.

[6] Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of prime order p. A bilinear map is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ with the following properties: (1) for any $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$; (2) for any generator g of $\mathbb{G}_1$, $e(g, g) \neq 1$; and (3) for any $u, v \in \mathbb{G}_1$, it is easy to compute $e(u, v)$.

1. $G(1^k)$ generates key pair (sk, vk) such that $sk = x \in \mathbb{Z}_p^*$ and $vk = (p, g, \mathbb{G}_1, X)$, where $\mathbb{G}_1$ is a group of prime order p, g is a generator of $\mathbb{G}_1$, and $X = g^x$.
2. $S(sk, m)$ signs the message m with the signing key sk by $\sigma(m) = g^{1/(x+m)}$.
3. $V(vk, m, \sigma)$ verifies the signature σ with vk by calculating $e(\sigma, g^m X)$. If the result equals $e(g, g)$, V outputs `valid`; otherwise, V outputs `invalid`.

Protocol 1. Proof of P_1's output authenticity

Common Input:	ciphertext of P_1's output $f_1 \oplus c = [\mathrm{M}_1, \ldots, \mathrm{M}_n]$, the indices of the evaluation-circuits $S = \{j_1, \ldots, j_e\}$ and the public key $(p, \mathbb{G}, g, X, Y)$ of the Boneh-Boyen signature scheme. In particular, $X = g^x$, and $Y = g^y$.
P_1 **Input:**	the corresponding private key (x, y) of the signature scheme.
P_2 **Input:**	the signature vector $[\sigma(b_1, 1, j_l), \ldots, \sigma(b_n, n, j_l)]$ such that $\sigma(b, i, j) = g^{1/(bx+iy+j)}$ and $j_l \in S$.
$P_1 \xrightarrow{Z, \{\delta(j)\}_{j \in S}} P_2$	P_1 picks another generator h of G and a random $z \in \mathbb{Z}_p^*$. Then P_1 sends $[Z, \delta(j_1), \ldots, \delta(j_e)]$ to P_2 such that $Z = h^z$ and $\delta(j) = h^{1/(z+j)}$.
$P_1 \xleftarrow{U_1, \ldots, U_n, V} P_2$	P_2 picks $u_1, \ldots, u_n, v \in \mathbb{Z}_p$ and computes $U_i \leftarrow \sigma(b_i, i, j_l)^{u_i}$ and $V \leftarrow \delta(j_l)^v$. Then $[U_1, \ldots, U_n, V]$ is sent to P_1.
$P_1 \xleftarrow{a_1, \ldots, a_n, b} P_2$	P_2 picks $\alpha, \beta_1, \ldots, \beta_n, \gamma \in \mathbb{Z}_p$ and sends $[a_1, \ldots, a_n, b]$ to P_1, where $a_i \leftarrow e(U_i, g)^{\alpha} e(g, g)^{\beta_i}$ and $b \leftarrow e(V, h)^{\alpha} e(h, h)^{\gamma}$.
$P_1 \xrightarrow{c} P_2$	P_1 picks $c \in \mathbb{Z}_p$ at random and sends it to P_2.
$P_1 \xleftarrow{z_\alpha, \{z_{\beta_i}\}, z_\gamma} P_2$	P_2 sends $z_\alpha \leftarrow \alpha + c \cdot j_l$, $z_{\beta_i} \leftarrow \beta_i - c \cdot u_i$, and $z_\gamma \leftarrow \gamma - c \cdot v$ back to P_1, who checks $a_i \stackrel{?}{=} e(U_i, X^{\mathrm{M}_i} Y^i)^c \cdot e(U_i, g)^{z_\alpha} \cdot e(g, g)^{z_{\beta_i}}$ for $i = 1, \ldots, n$ and $b \stackrel{?}{=} e(V, Z)^c \cdot e(V, h)^{z_\alpha} \cdot e(h, h)^{z_\gamma}$. P_1 aborts if any of the checks fails.

2.3 Committing Oblivious Transfer

The oblivious transfer (OT) primitive, introduced by Rabin [21], and extended by Even, Goldreich, and Lempel [5] and Brassard, Crépeau and Robert [3] works as follows: there is a sender with messages $[m_1, \ldots, m_n]$ and a receiver with a selection value $\sigma \in \{1, \ldots, n\}$. The receiver wishes to retrieve m_σ from the sender in such a way that (1) the sender does not "learn" anything about the receiver's choice σ and (2) the receiver "learns" only m_σ and nothing about any other message m_i for $i \neq \sigma$. Kiraz and Schoenmakers [10] introduced another notion of OT called *committing OT* in which the receiver also receives a perfectly-hiding and computationally-binding commitment to the sender's input messages, and the sender receives as output the values to open the commitment. Indeed, Kiraz and Schoenmakers introduced this notion specifically for use in a Yao circuit evaluation context. We adopt the idea behind their construction.

Formally, a one-out-of-two committing oblivious transfer OT_1^2 is a pair of interactive probabilistic polynomial-time algorithms sender and receiver. During the protocol, the sender runs with input messages $((m_0, r_0), (m_1, r_1))$, while the receiver runs with input the index $\sigma \in \{0,1\}$ of the message it wishes to receive. At the end of the protocol, the receiver outputs the retrieved message m'_σ and two commitments $\mathsf{com}_{\text{H}}(m_0; r_0), \mathsf{com}_{\text{H}}(m_1; r_1)$, and the sender outputs the openings (r_0, r_1) to these commitments. Correctness requires that $m'_\sigma = m_\sigma$ for all messages m_0, m_1, for all selections $\sigma \in \{0,1\}$ and for all coin tosses of the algorithms. Here, we use the standard notion of simulation security.

Theorem 1. *[19] If the Decisional Diffie-Hellman assumption holds in group G, there exists a protocol that securely computes the committing OT_1^2.*

Protocol 2 constructively proves Theorem 1. This protocol is a simple modification of the OT protocols designed by Peikert, Vaikuntanathan, and Waters [19] and later Lindell and Pinkas [14]. We simply add a ZK proof of knowledge in intermediate steps. Intuitively, the receiver-security is achieved due to the Decisional Diffie-Hellman assumption and the fact that the ZK proof of knowledge is independent of the receiver's input. On the other hand, the sender security comes from the uniform distributions of $X_{i,j}$ and $Y_{i,j}$ over G given that $r_{i,j}$ and $s_{i,j}$ are uniformly chosen and that the ZK proof has an ideal-world simulator for the verifier (or the receiver in the OT). As described in [15], it is possible to batch the oblivious transfer operations so that all n input keys (one for each bit) to s copies of the garbled circuit are transferred in one execution.

Protocol 2. Oblivious transfer for retrieving P_2's input keys [14]

Common:	A statistical security parameter s, a group G of prime order p, and G's generator g_0
P_1 **Input:**	Two s-tuples $[K_{0,1}, \ldots, K_{0,s}]$ and $[K_{1,1}, \ldots, K_{1,s}]$.
P_2 **Input:**	$\sigma \in \{0,1\}$
P_1 **Output:**	Commitment openings $\{K_{i,j}, r_{i,j}, s_{i,j}\}_{i \in \{0,1\}, 1 \le j \le s}$
P_2 **Output:**	$[K_{\sigma,1}, \ldots, K_{\sigma,s}]$ and $\{\mathsf{com}_{\text{H}}(K_{i,j}; r_{i,j}, s_{i,j})\}_{i \in \{0,1\}, 1 \le j \le s}$
$P_1 \xleftarrow{h_0, g_1, h_1} P_2$	P_2 picks $y, a \in \mathbb{Z}_p$ and sends $(g_1, h_0, h_1) \leftarrow (g_0^y, g_0^a, g_1^{a+1})$ to P_1.
$P_1 \xleftrightarrow{\text{ZK PoK}} P_2$	P_2 proves that (h_0, g_1, h_1) satisfies $(h_0 = g_0^a) \wedge (\frac{h_1}{g_1} = g_1^a)$.
$P_1 \xleftarrow{g, h} P_2$	P_2 picks $r \in \mathbb{Z}_p$ and sends $g \leftarrow g_\sigma^r$ and $h \leftarrow h_\sigma^r$ to P_1.
$P_1 \xrightarrow{\{X_{i,j}, Y_{i,j}\}} P_2$	For $i \in \{0,1\}, 1 \le j \le s$, P_1 picks $r_{i,j}, s_{i,j} \in \mathbb{Z}_p$ and sends $X_{i,j}$ and $Y_{i,j}$ to P_1, where $X_{i,j} = g_i^{r_{i,j}} h_i^{s_{i,j}}$ and $Y_{i,j} = g^{r_{i,j}} h^{s_{i,j}} \cdot K_{i,j}$. P_2 gets $\mathsf{com}_{\text{H}}(K_{i,j}; r_{i,j}, s_{i,j}) = (X_{i,j}, Y_{i,j})$ and computes key $K_{\sigma,j} \leftarrow Y_{\sigma,j} \cdot X_{\sigma,j}^{-r}$.

3 Main Protocol

Here we put all the pieces together to form the complete protocol. Note that $\mathsf{com}_{\mathrm{H}}(K;t)$ denotes a perfectly-hiding commitment to K with opening t, and $\mathsf{com}_{\mathrm{B}}(K;t)$ denotes a perfectly-binding commitment to K with opening t.

Common input: a security parameters k, a statistical security parameter s, a malleable claw-free collection $(G_{\mathrm{CLW}}, D_{\mathrm{CLW}}, F_{\mathrm{CLW}}, R_{\mathrm{CLW}})$, a signature scheme $(G_{\mathrm{SIG}}, S_{\mathrm{SIG}}, V_{\mathrm{SIG}})$, a two-universal hash function $H : \{0,1\}^* \to \{0,1\}^k$, and the description of a boolean circuit C computing $f(x,y) = (f_1, f_2)$, where $|x| = 2n$ (including the extra n-bit random input) and $|y| = |f_1| = |f_2| = n$.

Private input: P_1 has the original input $x_1 \dots x_n$ and the extra random input $x = x_{n+1} \dots x_{2n}$, while P_2 has input $y = y_1 y_2 \dots y_n$.
Private output: P_1 receives output $f_1(x,y)$, while P_2 receives output $f_2(x,y)$.

1. P_2 runs the index selecting algorithm $I \leftarrow G_{\mathrm{CLW}}(1^k)$ and sends I to P_1.
2. **Committing OT for P_2's input:** For every $1 \le i \le n$ and every $1 \le j \le s$, P_1 picks a random pair of k-bit strings $(K^0_{i,j}, K^1_{i,j})$, which is associated with P_2's i-th input wire in the j-th circuit. Both parties then conduct n instances of committing OT in parallel. In the i-th instance,
   (a) P_1 uses input $([K^0_{i,1}, \dots, K^0_{i,s}], [K^1_{i,1}, \dots, K^1_{i,s}])$, whereas P_2 uses input y_i.
   (b) P_1 gets the openings $([t^0_{i,1}, \dots, t^0_{i,s}], [t^1_{i,1}, \dots, t^1_{i,s}])$ to both commitment vectors, whereas P_2 gets the vector of her choice $[K^{y_i}_{i,1}, \dots, K^{y_i}_{i,s}]$ and the commitments to both vectors, ie., $[\mathsf{com}_{\mathrm{H}}(K^0_{i,1}; t^0_{i,1}), \dots, \mathsf{com}_{\mathrm{H}}(K^0_{i,s}; t^0_{i,s})]$ and $[\mathsf{com}_{\mathrm{H}}(K^1_{i,1}; t^1_{i,1}), \dots, \mathsf{com}_{\mathrm{H}}(K^1_{i,s}; t^1_{i,s})]$.
3. **Garbled circuit construction:** P_1 runs the key generating algorithm $G_{\mathrm{SIG}}(1^k)$ to generate a signature key pair (sk_1, pk_1) and the domain sampling algorithm $D_{\mathrm{CLW}}(I)$ to generate domain element $m^b_{i,j}$, for $b \in \{0,1\}$, $1 \le i \le 2n$, $1 \le j \le s$. Next, P_1 constructs s independent copies of garbled version of C, denoted by $GC_1, \dots, GC_s$. In addition to Yao's construction, circuit GC_j also satisfies the following:
   (a) $J^b_{i,j}$ is associated with value b to P_1's i-th input wire, where $J^b_{i,j}$ is extracted from group element $F_{\mathrm{CLW}}(b, I, m^b_{i,j})$, ie., $J^b_{i,j} = H(F_{\mathrm{CLW}}(b, I, m^b_{i,j}))$.
   (b) $K^b_{i,j}$ chosen in Step 2 is associated with value b to P_2's i-th input wire.
   (c) $b || S_{\mathrm{SIG}}(sk_1, (b,i,j))$ is associated with bit value b to P_1's i-th output wire.
4. For $b \in \{0,1\}, 1 \le i \le 2n, 1 \le j \le s$, P_1 sends circuits $GC_1, \dots, GC_s$ and the commitments to $F_{\mathrm{CLW}}(b, I, m^b_{i,j})$, denoted by $\mathsf{com}_{\mathrm{B}}(F_{\mathrm{CLW}}(b, I, m^b_{i,j}); r^b_{i,j})$ to P_2.
5. **Cut-and-choose:** P_1 and P_2 conduct the coin flipping protocol to generate a random tape, by which they agree on a set of check-circuits. Let T be the resulting set, that is, $T \subset \{1, \dots, s\}$ and $|T| = 3s/5$. For every $j \in T$, P_1 sends to P_2 P_1s of garbled circuit GC_j, including $[K^b_{1,j}, \dots, K^b_{n,j}]$, $[t^b_{1,j}, \dots, t^b_{n,j}]$, $[m^b_{1,j}, \dots, m^b_{2n,j}]$, $[r^b_{1,j}, \dots, r^b_{2n,j}]$, for $b \in \{0,1\}$, and the random keys associated with each wire of GC_j. P_2 check the following:
   (a) The commitment from Step 2 is revealed to $K^b_{i,j}$ with $t^b_{i,j}$.

(b) The commitment from Step 4 is revealed to $F_{\text{CLW}}(b, I, m^b_{i,j})$ with $r^b_{i,j}$.
(c) GC_j is a garbled version of C^* that is correctly built. In particular,

- $H(F_{\text{CLW}}(b, I, m^b_{i,j}))$ is associated with value b to P_1's i-th input wire;
- $K^b_{i,j}$ is associated with bit value b to P_2's i-th input wire;
- $V_{\text{SIG}}(pk_1, (b, i, j), \sigma(b, i, j)) = \mathsf{valid}$, where $\sigma(b, i, j)$ is the signature comes along with bit value b from P_1's i-th output wire;
- the truth table of each boolean gate is correctly converted to the doubly-encrypted entries of the corresponding garbled gate.

If any of the above checks fails, P_2 aborts.

6. **Consistency check for P_1's inputs:** Let $e = 2s/5$ and $\{j_1, \ldots, j_e\}$ be the indices of evaluation-circuits. P_1 then decommits to her input keys for the evaluation-circuits by sending $([r^{x_1}_{1,j_1}, \ldots, r^{x_{2n}}_{2n,j_1}], \ldots, [r^{x_1}_{1,j_e}, \ldots, r^{x_{2n}}_{2n,j_e}])$ to P_2. Let $[M_{1,j_1}, \ldots, M_{2n,j_1}], \ldots, [M_{1,j_e}, \ldots, M_{2n,j_e}]$ be the resulting decommitments. Next, P_1 proves the consistency of her i-th input bit by sending $[m^{x_i}_{i,j_2} \star (m^{x_i}_{i,j_1})^{-1}, \ldots, m^{x_i}_{i,j_e} \star (m^{x_i}_{i,j_1})^{-1}]$ to P_2, who then checks if

$$M_{i,j_l} = M_{i,j_1} \diamond R_{\text{CLW}}(I, m^{x_i}_{i,j_l} \star (m^{x_i}_{i,j_1})^{-1}), \text{ for } l = 2, \ldots, e.$$

P_2 aborts if any of the checks fails. Otherwise, let $J^{x_i}_{i,j_l} = H(M_{i,j_l})$.

7. **Circuit evaluation:** For every $j \in \{j_1, \ldots, j_e\}$, P_2 now has key vectors $[J^{x_1}_{1,j}, \ldots, J^{x_{2n}}_{2n,j}]$ (from Step 6) representing P_1's input x and $[K^{y_1}_{1,j}, \ldots, K^{y_n}_{n,j}]$ (from Step 2) representing P_2's input y. So P_2 is able to do the evaluation on circuit GC_j and get P_1's output $[\mathrm{M}_{1,j}||\sigma(\mathrm{M}_{1,j}), \ldots, \mathrm{M}_{n,j}||\sigma(\mathrm{M}_{n,j})]$ and P_2's output $[\mathrm{N}_{1,j}, \ldots, \mathrm{N}_{n,j}]$, where $\mathrm{M}_{i,j}, \mathrm{N}_{i,j} \in \{0, 1\}$. Let $\mathcal{M}_j = [\mathrm{M}_{1,j}, \ldots, \mathrm{M}_{n,j}]$ and $\mathcal{N}_j = [\mathrm{N}_{1,j}, \ldots, \mathrm{N}_{n,j}]$ be the n-bit outputs for P_1 and P_2, respectively. P_2 then chooses index j_l such that $\mathcal{M}_{j_l}$ and $\mathcal{N}_{j_l}$ appear more than $e/2$ times in vectors $[\mathcal{M}_{j_1}, \ldots, \mathcal{M}_{j_e}]$ and $[\mathcal{N}_{j_1}, \ldots, \mathcal{N}_{j_e}]$, respectively. P_2 sends $\mathcal{M}_{j_l}$ to P_1 and takes $\mathcal{N}_{j_l}$ as her final output. If no such j_l exists, P_2 aborts.

8. **Verification to P_1's output:** To convince P_1 the authenticity of $\mathcal{M}_{j_l}$ without revealing j_l, P_1 generates another signature key pair (sk_2, pk_2). Then P_1 signs the indices of all the evaluation-circuits and sends the results to P_2. In particular, P_1 sends to P_2 the public key pk_2 and a signature vector $[\delta(j_1), \ldots, \delta(j_e)]$, where $\delta(j) = S_{\text{SIG}}(sk_2, j)$. The signature is verified by P_2 by checking $V_{\text{SIG}}(pk_2, j, \delta(j)) = \mathsf{valid}$, for every $j \in \{j_1, \ldots, j_e\}$. Next, P_2 proves to P_1 in witness-indistinguishable sense the knowledge of $\sigma(\mathrm{M}_{i,j_l}, i, j)$ (a signature signed with sk_1) and $\delta(j^*)$ (a signature signed with sk_2) such that j and j^* are equivalent, for $1 \le i \le n$. P_1 aborts if the proof is not valid; otherwise, P_1 takes $\mathcal{M}_{j_l} \oplus (x_{n+1}, \ldots, x_{2n})$ as her final output.

Theorem 2. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ be any function. Given a secure committing oblivious transfer protocol, a perfectly-hiding commitment scheme, a perfectly-binding commitment scheme, a malleable claw-free family, and a pseudo-random function family, the Main protocol securely computes f.*

We have omitted the standard simulation-based definition of "securely computes f" for space. Roughly, this definition requires a simulator for the corrupted

evaluator, and a simulator for the corrupted generator that is able to generate transcripts given only oracle access to either the evaluator or generator (respectively) that are indistinguishable from the transcripts produced in real interactions between the corrupted generator and honest evaluator or honest generator and corrupted evaluator. (A simulator for when both parties are corrupted is also required but trivial.) The proof of Theorem 2 is omitted for space.

4 Experimental Results

We produced an implementation of our protocol to demonstrate its practical benefits. Our implementation takes the boolean circuit generated by Fairplay compiler as input. The encryption function used to construct garbled gates is defined as $\mathsf{Enc}_{J,K}(m) = (m \oplus \text{SHA-256}(J) \oplus \text{SHA-256}(K))_{1\ldots k}$, where $|J| = |K| = |m| = k$, and $S_{1\ldots k}$ denotes the least significant k bits of S. Here SHA-256 is modeled as a pseudorandom function. The choice of SHA-256 is to make a fair comparison as it is used in [20].

Following Pinkas et. al [20], we set the security level to 2^{-40} and the security parameter k (key length) to 128-bit. In the first experiment, P_1 and P_2 hold a 32-bit input $x = (x_{31}x_{30}\ldots x_0)_2$ and $y = (y_{31}y_{30}\ldots y_0)_2$, respectively. They want to compute $f(x, y) = (f_1, f_2)$ such that after the secure computation, P_1 receives $f_1 = \sum_{i=0}^{31} x_i \oplus y_i$, and P_2 receives f_2 as the result of comparison between x and y. The 6 gates of overhead we incur in the first experiment relate to our method for two-output functions. In the second experiment, P_2 has a 128-bit message block while P_1 has a 128-bit encryption key. They want to securely compute the AES encryption, and only P_2 gets the ciphertext.

We ran our experiments on two machines: `slower` and `fast`, where `slower` runs OS X 10.5 with Intel Core 2 Duo 2.8 GHz and 2GB RAM, and `fast` runs CentOS with Intel Xeon Quad Core E5506 2.13 GHz and 8GB RAM. `slower` is not as powerful as the machine used in [20] (Intel Core 2 Duo 3.0 GHz, 4GB RAM), and `fast` is the next closest machine that we have.

Table 2 reports the *best* numbers from [20]. We note that [20] applies the Garbled Row Reduction technique so that even non-XOR gates can save 25% of

Table 2. The performance comparison with [20]

	# Gates			Time (s)			Totals	
	Base	Overhead	Non-XOR	Precomp	OT	Calc	Time (s)	KBytes
(f_1, f_2)	531	2,250	278	117	16	39	172	140,265
Ours (on `slower`)	531	6	237	35	15	21	**71**	**5,513**
Ours (on `fast`)	531	6	237	27	11	15	53	5,513
$(\lambda, \text{AES}_x(y))$	33,880	12,080	11,490	483	34	361	878	406,010
Ours (on `slower`)	33,880	0	11,286	138	58	69	**265**	**190,122**
Ours (on `fast`)	33,880	0	11,286	98	44	50	192	190,122

Table 3. The running time (in seconds) of two experiments on machine `slower`

	$f(x,y) = (f_1, f_2)$			$f(x,y) = (\lambda, \mathrm{AES}_x(y))$		
	P_1	P_2	Sum (s)	P_1	P_2	Sum (s)
Precomp Time	35.4	0.0	35.4	137.7	0.0	137.7
OT Time	7.9	6.7	14.6	31.9	26.3	58.2
Cut-and-Choose	0.0	14.7	14.7	0.0	44.4	44.4
Input Check	0.0	3.0	3.0	0.0	10.0	10.0
Eval Time	0.0	3.4	3.4	0.0	14.1	14.1
Two-output	0.1	0.0	0.1	0.0	0.0	0.0
Total (s)	43.4	27.8	71.2	169.6	94.8	264.4

COMM. FOR EACH STAGE (KBYTES)		
Circuit construction	2,945	53.42%
Oblivious transfer	675	12.25%
Cut-and-choose	1,813	32.89%
P_1's input consistency	76	1.38%
P_1's output validity	3	0.01%
Total communication	5,513	100.00%

(a)

SEMI-HONEST ADVERSARIES		
	This work	[20]
No. of gates	531	531
Comm. (KBytes)	23	22
MALICIOUS ADVERSARIES		
No. of gates	537	2,781
Comm. (KBytes)	5,513	167,276

(b)

Fig. 3. (a) Communication cost for Experiment 1 by stages for our solution given statistical security parameter $s = 125$ and security parameter $k = 128$. (b) The circuit size and communication cost comparison with [20] (which also ensures the cheating probability is limited below 2^{-40}).

the communication overhead. A future version of our protocol can also reap this 25% reduction since the technique is compatible with our protocol.

Our implementation involves a program for P_1 and one for P_2. For the purpose of timing, we wrote another program that encapsulates both of these programs and feeds the output of one as the input of the other and vice versa. Timing routines are added around each major step of the protocol and tabulated in Table 3. This timing method eliminates any overhead due to network transmission, which we cannot reliably compare. The reported values are the averages from 5 runs.

We implemented our solution with the PBC (Pairing Based Cryptography) library [1] for testing. The components of our protocol, including the claw-free collections, the generator's input consistency check, and the generator's output validity check, are built on top of the elliptic curve $y^2 = x^3 + 3$ over the field $\mathbb{F}_q$ for some 80-bit prime q. We have made systems-level modifications to the random

COMM. FOR EACH STAGE (KBYTES)		
Circuit construction	99,408	52.29%
Oblivious transfer	2,699	1.42%
Cut-and-choose	87,585	46.16%
P_1's input consistency	256	0.13%
P_1's output validity	0	0.00%
Total communication	190,122	100.00%

(a)

SEMI-HONEST ADVERSARIES		
	This work	[20]
No. of gates	33,880	33,880
Comm. (KBytes)	795	503
MALICIOUS ADVERSARIES		
No. of gates	33,880	45,960
Comm. (KBytes)	190,122	406,010

(b)

Fig. 4. (a) Communication cost for Experiment 2 by stages for our solution given statistical security parameter $s = 125$ and security parameter $k = 128$.

bit sampling function of the PBC library (essentially to cache file handles and eliminate unnecessary systems calls).

In Table 4, we list the results of the MAC-based two-output function handling and ours. The MAC approach introduces extra 16,384 (128^2) non-XOR gates to the AES circuit, whereas the original AES circuit has only 11,286 non-XOR gates. Since the number of non-XOR gates is almost doubled in the MAC-based approach, their circuit construction and evaluation need time about twice as much as ours. Moreover, the MAC-based approach has twice as many input bits as ours so that the time for P_1's input consistency has doubled.

Table 4. Computation time (in seconds) of $f(x, y) = (\mathrm{AES}_x(y), \lambda)$ running on machine `slower` under different two-output handling methods

	MAC two-output approach			Our two-output approach		
	P_1	P_2	Subtotal	P_1	P_2	Subtotal
Precomp Time	498.9	0.0	498.9	294.1	0.0	294.1
OT Time	32.0	26.3	58.3	31.9	26.2	58.1
Cut-and-Choose	0.0	158.6	158.6	0.0	185.3	185.3
Input Check	0.0	40.4	40.4	0.0	19.8	19.8
Eval Time	0.0	50.6	50.6	0.0	24.4	24.4
Two-output	0.0	0.0	0.0	0.7	0.6	1.3
Total	530.9	275.9	806.8	326.7	256.3	583.0

References

1. Pairing-Based Cryptography Library (2006), `http://crypto.stanford.edu/pbc/`
2. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. Journal of Cryptology 21, 149–177 (2008)

3. Brassard, G., Crépeau, C., Robert, J.M.: All-or-Nothing Disclosure of Secrets. In: Odlyzko, A. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 234–238. Springer, Heidelberg (1987)
4. Camenisch, J., Chaabouni, R., Shelat, A.: Efficient Protocols for Set Membership and Range Proofs. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 234–252. Springer, Heidelberg (2008)
5. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. Communications of ACM 28, 637–647 (1985)
6. Goldreich, O., Micali, S., Wigderson, A.: How to Play ANY Mental Game. In: 19th Annual ACM Symposium on Theory of Computing, pp. 218–229. ACM, New York (1987)
7. Goldreich, O., Kahan, A.: How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology 9, 167–189 (1996)
8. Jarecki, S., Shmatikov, V.: Efficient Two-Party Secure Computation on Committed Inputs. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 97–114. Springer, Heidelberg (2007)
9. Kiraz, M.: Secure and Fair Two-Party Computation. Ph.D. thesis, Technische Universiteit Eindhoven (2008)
10. Kiraz, M., Schoenmakers, B.: A Protocol Issue for The Malicious Case of Yao's Garbled Circuit Construction. In: 27th Symposium on Information Theory in the Benelux, pp. 283–290 (2006)
11. Kiraz, M., Schoenmakers, B.: An Efficient Protocol for Fair Secure Two-Party Computation. In: Malkin, T. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 88–105. Springer, Heidelberg (2008)
12. Kolesnikov, V., Schneider, T.: Improved Garbled Circuit: Free XOR Gates and Applications. In: Aceto, L., Damgård, I., Goldberg, L., Halldórsson, M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 486–498. Springer, Heidelberg (2008)
13. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
14. Lindell, Y., Pinkas, B.: Secure Two-Party Computation Via Cut-and-Choose Oblivious Transfer. Crypto ePrint Archive (2010), http://eprint.iacr.org/2010/284
15. Lindell, Y., Pinkas, B., Smart, N.: Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
16. Mohassel, P., Franklin, M.: Efficiency Tradeoffs for Malicious Two-Party Computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)
17. Naor, M., Pinkas, B.: Oblivious transfer with adaptive queries. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, p. 791. Springer, Heidelberg (1999)
18. Nielsen, J., Orlandi, C.: LEGO for Two-Party Secure Computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
19. 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)
20. Pinkas, B., Schneider, T., Smart, N., Williams, S.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)

21. Rabin, M.: How to Exchange Secrets by Oblivious Transfer. Tech. Rep. TR-81, Harvard Aiken Computation Laboratory (1981)
22. Woodruff, D.: Revisiting the Efficiency of Malicious Two-Party Computation. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 79–96. Springer, Heidelberg (2007)
23. Yao, A.: Protocols for Secure Computations. In: 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE Computer Society, Los Alamitos (1982)

A Optimal Choice in Cut-and-Choose Strategy

According to the cut-and-choose strategy, P_2 chooses e copies of the garbled circuits and asks P_1 to open the rest $(s-e)$. After the verification, P_2 evaluates the rest e copies of the circuits and takes the majority output as her output. A natural question is: *Under the assumption that P_1's inputs are consistent, how many circuits does P_2 evaluate in order to minimize the probability for P_1's best cheating strategy to succeed?*

The assumption is valid due to the consistency check on P_1's input. Given that s and e are fixed and known to P_1, let b be the number of bad circuits created by P_1. A circuit is *bad* if either the circuit is wrongly constructed or P_2's inputs are selectively failed via OT. The goal is to find e and b such that the probability that P_1 cheats without getting caught $\binom{s-b}{s-e}/\binom{s}{s-e}$ is minimized.

We first claim that P_1's best cheating strategy is to produce $b = \lfloor e/2 \rfloor + 1$ bad circuits. Indeed, if $b \leq \lfloor e/2 \rfloor$, P_2's output will not get affected since the faulty outputs will be overwhelmed by majority good ones. Also, the more bad circuits, the more likely that P_1 will get caught since $\binom{s-(b-1)}{s-e} > \binom{s-b}{s-e}$. So the best strategy for P_1 to succeed in cheating is to construct as few bad circuits as possible while the majority of evaluation circuits are bad, which justifies the choice of b.

Our next goal is to find the e that minimizes $\Pr(e) = \binom{s-\lfloor \frac{e}{2} \rfloor - 1}{s-e}/\binom{s}{s-e}$. To get rid of the troublesome floor function, we will consider the case when e is even and odd separately. When $e = 2k$ for some $k \in \mathbb{N}$ such that $k \leq \frac{s}{2}$, let $\Pr_{\text{even}}(k) = \binom{s-k-1}{s-2k}/\binom{s}{s-2k}$. Observer that $\frac{\Pr_{\text{even}}(k+1)}{\Pr_{\text{even}}(k)} = \frac{(2k+1)(2k+2)}{(s-k-1)k}$. It is not hard to solve the quadratic inequality and come to the result that

$$\frac{\Pr_{\text{even}}(k+1)}{\Pr_{\text{even}}(k)} \leq 1 \text{ when } 0 < k \leq \frac{1}{10}\left(s - 7 + \sqrt{(s-7)^2 - 40}\right) \stackrel{\text{def}}{=} \alpha.$$

In other words, $\Pr_{\text{even}}(k) \geq \Pr_{\text{even}}(k+1)$ when $0 < k \leq \alpha$; and $\Pr_{\text{even}}(k) < \Pr_{\text{even}}(k+1)$ when $\alpha < k \leq \frac{s}{2}$. Therefore, $\Pr_{\text{even}}$ is minimal when $k = \lceil \alpha \rceil$. Similarly, when $e = 2k+1$, the probability $\Pr_{\text{odd}}(k) = \binom{s-k-1}{s-2k-1}/\binom{s}{s-2k-1}$ is minimal when $k = \lceil \beta \rceil$, where $\beta = \frac{1}{5}(s-7)$. In summary,

$$\begin{cases} \Pr_{\text{even}}(e) \text{ is minimal when } e = 2\lceil \alpha \rceil; \\ \Pr_{\text{odd}}(e) \text{ is minimal when } e = 2\lceil \beta \rceil + 1, \end{cases}$$

and $\Pr(e)$'s minimum is one of them.

Efficient Non-interactive Secure Computation

Yuval Ishai[1,⋆], Eyal Kushilevitz[1,⋆⋆], Rafail Ostrovsky[2,⋆⋆⋆], Manoj Prabhakaran[3,†], and Amit Sahai[2,‡]

[1] Dept. of Computer Science, Technion, Haifa, Israel
{yuvali,eyalk}@cs.technion.il
[2] University of California, Los Angeles
{rafail,sahai}@cs.ucla.edu
[3] University of Illinois, Urbana-Champaign
mmp@cs.uiuc.edu

Abstract. Suppose that a receiver R wishes to publish an encryption of her secret input x so that every sender S, holding an input y, can reveal $f(x, y)$ to R by sending her a single message. This should be done while simultaneously protecting the secrecy of y against a corrupted R and preventing a corrupted S from having an unfair influence on the output of R beyond what is allowed by f.

When the parties are semi-honest, practical solutions can be based on Yao's garbled circuit technique. However, for the general problem when the parties, or even S alone, may be malicious, all known polynomial-time solutions are highly inefficient. This is due in part to the fact that known solutions make a *non-black-box* use of cryptographic primitives, e.g., for providing non-interactive zero-knowledge proofs of statements involving cryptographic computations on secrets.

Motivated by the above question, we consider the problem of secure two-party computation in a model that allows only parallel calls to an ideal oblivious transfer (OT) oracle with no additional interaction. We obtain the following results.

- **Feasibility.** We present the first general protocols in this model which only make a *black-box* use of a pseudorandom generator (PRG). All previous OT-based protocols either make a non-black-box use of cryptographic primitives or require multiple rounds of interaction.
- **Efficiency.** We also consider the question of minimizing the asymptotic number of PRG calls made by such protocols. We show that $\mathrm{polylog}(\kappa)$ calls are sufficient for each gate in a (large) boolean circuit computing f, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error of a malicious sender. Furthermore, the number of PRG calls per gate can be made *constant* by settling for a relaxed notion of security which allows a malicious S to arbitrarily correlate the event that R detects cheating

⋆ Work done in part while visiting UCLA. Supported by ERC Starting Grant 259426, ISF grant 1361/10, and BSF grant 2008411.

⋆⋆ Work done in part while visiting UCLA. Supported by ISF grant 1361/10 and BSF grant 2008411.

⋆⋆⋆ Research supported in part from NSF grants 0830803 and 0916574, BSF grant 2008411, and grants from Okawa Foundation, IBM, and Lockheed-Martin Corporation.

† Supported by NSF grant CNS 07-47027.

‡ Research supported in part from NSF grants 0916574, 0830803, 0716389, and 0627781, BSF grant 2008411, a Google Research Award, a Xerox Foundation Grant, an equipment grant from Intel, and an Okawa Foundation Research Grant.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 406–425, 2011.

with the input of R. This improves over the state of the art also for *interactive* constant-round black-box protocols, which required $\Omega(\kappa)$ PRG calls per gate, even with similar relaxations of the notion of security.

Combining the above results with 2-message (parallel) OT protocols in the CRS model, we get the first solutions to the initial motivating question which only make a black-box use of standard cryptographic primitives.

1 Introduction

This work is motivated by the following variant of the problem of computing on encrypted data [42,43]. Suppose that a receiver R wishes to publish a semantically secure encryption of her secret input x so that any sender S, holding an input y, can reveal $f(x, y)$ to R by sending her a single message. (The message can be seen as an encryption of $f(x, y)$ that the receiver can decrypt). We want this process to protect the secrecy of y against a corrupted R and, at the same time, prevent a corrupted S from having an unfair influence on the output of R beyond what is allowed by f. We refer to this flavor of computing on encrypted data as *non-interactive secure computation* (NISC).

As a concrete motivating scenario for NISC, consider a receiver Roberta who wishes to publish an encrypted version of her personal profile x on her public web page towards finding a suitable partner for dating. A solution to our problem would allow an arbitrary sender Sam, with personal profile y, to send an email message to Roberta which reveals his verifiable contact information only if the two profiles match. (The matching criteria can either be determined by a public algorithm which is embedded into f, or alternatively specified in Roberta's secret profile). In order to protect the secrecy of Roberta's profile x, its encryption should be semantically secure. In order to protect the secrecy of Sam's profile y, he should be ensured that no information is revealed to Roberta other than what is implied by the output of f. Finally, to help protect Roberta against eager senders who try to force a match, she should be ensured that every strategy of such a sender corresponds to some valid profile y.

Standard techniques for secure computation and computing on encrypted data perform quite well when the parties are guaranteed to be semi-honest. For instance, practical NISC protocols in this setting can be obtained by combining Yao's garbled circuit technique [44,31] and any two-message oblivious transfer (OT) protocol [7,14]. Low-communication (but at this point less practical) solutions can be obtained using homomorphic encryption for general functions [13] or for restricted classes of functions [30,6,22,33].

For some of the above protocols, protecting S against a malicious R can come at a relatively low cost. In protocols based on Yao's construction this can be done (in the CRS model) by using efficient 2-message UC-secure OT protocols [39] (see also [11]). However, known techniques for protecting R against a malicious S either involve additional rounds of interaction [32] or are highly inefficient. For instance, this is the case if S is required to prove, using non-interactive zero-knowledge (NIZK), that he constructed a valid garbled circuit [7,17]). Such proofs seem very costly even with the best known NIZK techniques. Moreover, even from a qualitative point of view, such NIZK-based solutions leave much to be desired in that they inherently require to make

a *non-black-box* use of the underlying cryptographic primitives. For instance, while the semi-honest version of Yao's protocol can be implemented by making a black-box use of (parallel) 2-message OT and a pseudorandom generator (PRG), this is not the case for a NIZK-based variant which is secure against malicious senders.

The above state of affairs motivates the study of NISC protocols which only make a *black-box* use of standard cryptographic primitives. We further simplify the problem by allowing S and R to engage in *parallel* invocations of an ideal OT oracle, but without allowing any additional interaction[1]. We refer to such a protocol as a NISC protocol in the *OT-hybrid* model, or NISC/OT protocol for short. More formally, a NISC/OT protocol for $f(x, y)$ is a protocol which UC-securely realizes f using only parallel calls to an ideal OT oracle.

Our main motivation for considering this a good model for NISC is the aforementioned existence of efficient UC-secure 2-message implementations of OT in the CRS model. Indeed, using the UC composition theorem [8], NISC/OT protocols can be combined with such 2-message OT protocols to yield NISC protocols in the CRS model that have the same communication pattern as in the initial motivating example.

Additional advantages of OT-based protocols include the *generality* advantage of being able to realize OT in a variety of models and under a variety of standard assumptions, as well as efficiency advantages such as the possibility of precomputing the necessary OTs [4,2] and the possibility to amortize the cost of this precomputation [3,36,18]. See [24] for further discussion.

We turn to the feasibility question of minimizing the assumptions on which NISC/OT protocols can be based. In the OT-hybrid model, any polynomial-time computable functionality can be efficiently realized *unconditionally* [16,28,24]. However, it is wide open whether the same is true for constant-round protocols. (This question is related to the corresponding question in the honest-majority MPC setting [5], which in turn is related to other wide open questions [19]). Given the lack of progress on this front, a second best alternative is to base general NISC/OT protocols on any one-way function, or equivalently a PRG. As noted above, Yao's protocol provides such a solution in the semi-honest model. Moreover, it is shown in [24] (see Appendix B of [23]) how to get a similar protocol in the malicious NISC/OT model; however, this protocol inherently makes a non-black-box use of the PRG. This motivates the following main question:

> *Are there NISC/OT protocols for general functions which only make a black-box use of a PRG?*

A second goal of this work is to push the asymptotic efficiency limits of constant-round black-box protocols by minimizing the number of calls to the underlying cryptographic primitive. Existing constant-round black-box protocols in the OT-hybrid model (such as [34,32] and their variants) require $\Omega(\kappa)$ calls to a PRG (or symmetric encryption) for each gate in the circuit, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error for a malicious sender[2]. This should be compared to the

[1] It is also useful to allow a message from the sender to the receiver which is independent of the receiver's OT choices; such a message can be realized in the pure parallel-OT hybrid model at the cost of one additional OT.

[2] The "LEGO protocol" [38] reduces this overhead by a factor of $\log |C|$, where $|C|$ is the size of the circuit, at the expense of employing a homomorphic commitment primitive.

best protocols in the semi-honest model [44,31] which require only $O(1)$ PRG calls per gate.

1.1 Our Results

We obtain the following main results.

- **Feasibility.** We present the first general NISC/OT protocols which only make a black-box use of a PRG. All previous protocols in the OT-hybrid model either make a non-black-box use of cryptographic primitives [24] or require multiple rounds of interaction (cf. [32]).
- **Efficiency.** We also consider the question of minimizing the asymptotic number of PRG calls made by such protocols. We show that $\mathrm{polylog}(\kappa)$ calls are sufficient for each gate in a (large) boolean circuit computing f, where κ is a statistical security parameter guaranteeing at most $2^{-\kappa}$ simulation error of a malicious sender[3]. Furthermore, the number of PRG calls per gate can be made *constant* by settling for a relaxed notion of security which allows a malicious S to arbitrarily correlate the event that R detects cheating with the input of R.
  This improves over the state of the art also for *interactive* constant-round black-box protocols, which required $\Omega(\kappa)$ PRG calls per gate, even with similar relaxations of the notion of security.

Combining the above results with 2-message (parallel) OT protocols in the CRS model, we get the first solutions to the initial motivating question which only make a black-box use of standard cryptographic primitives.

On re-using public keys. A standard security caveat that applies to many non-interactive protocols in the public key model (cf. [29,27,12,9]) is that re-using the same receiver's public key for multiple sender messages may be problematic if the sender can learn the receiver's output on these messages. Indeed, the standard (UC-)security guarantee of our protocols only applies when an independent receiver message is used in each session. While the receiver's output does not reveal additional information about the receiver's input (other than what is allowed by f), it *may* reveal information about the secret randomness embedded in the public key, which may in turn compromise the receiver's security when leaking multiple outputs without refreshing the public key. Our protocols are indeed susceptible to this type of attacks.

We stress that re-using the same public key for multiple sender messages is always safe (in the sense of providing the natural "real-ideal" security guarantee) if the receiver refreshes the public key after revealing an output or using it in another protocol. This seems to be a very mild requirement in many practical scenarios in which sender messages are infrequent or can be aggregated before taking any action.

Similarly to [9], we can provide t-time reusable public keys (for which up to t outputs can be revealed before the key needs to be refreshed) at a much better cost than publishing t independent public keys. We note, however, that (non-black-box) NIZK-based NISC protocols are not susceptible at all to this type of attacks, and leave the

[3] The simulation error of the receiver is close to the distinguishing advantage of the PRG (as in Yao's original protocol) and can be made $2^{-\Omega(\kappa)}$ by choosing a PRG with similar strength.

possibility of obtaining a similar result using black-box constructions as an interesting open question.

On asymptotic vs. practical efficiency. As is usual in theoretical work in cryptography, we focus on optimizing asymptotic efficiency and do not try to optimize or even analyze the underlying hidden constants. Moreover, in doing so we focus on the typical case where the circuit size is much bigger than the input size which in turn is much bigger than the security parameter, and sometimes ignore low-order *additive* terms that depend on the smaller quantities. These optimization goals may conflict with practical efficiency. The question of optimizing NISC protocols towards practical implementations is left for future work.

1.2 Overview of Techniques

At a high level, our NISC/OT protocols are obtained using the following modular steps:

1. Statistically secure NISC/OT protocols for $\mathbf{NC}^0$ functions. Here we can rely on a previous protocol from [24] (see Appendix B of [23]). We also present an asymptotic efficiency improvement by applying "MPC in the head" techniques in the spirit of [20]. This is presented in Section 3.
2. Computationally secure NISC protocols for *general* functions in the $\mathbf{NC}^0$-hybrid model (allowing the parties a single call to an ideal $\mathbf{NC}^0$ functionality). Here we combine a particular implementation of Yao's garbled circuit construction with the use of unconditional one-time MACs to guarantee that a malicious sender can either deliver a correct output to the receiver or the receiver detects cheating and aborts. However, these protocols allow a malicious sender to correlate the event of the receiver aborting with the receiver's input. We present two variants of the protocol: the first (Section 5) allows arbitrary correlations with the receiver's inputs, and is the most efficient protocol we obtain in this work. The second variant (Section 6) is slightly less efficient but allows only correlations that can be expressed as disjunctions of circuit wires and their negations.
3. Finally, we present (in Section 7) an efficient general reduction of full security to security with the latter type of "correlated abort". The idea is to transform the original circuit into a new, randomized, circuit in which disjunctions of wires or their negations provide essentially no information about the input. A special case of this transformation is implicit in [25]. We reduce the general case to honest-majority MPC in the semi-honest model and instantiate it using a recent efficient protocol from [10].

We also present (in Section 4) a direct ad-hoc construction of NISC protocols in the $\mathbf{NC}^0$-hybrid model, which is asymptotically less efficient but is somewhat simpler than that obtained by combining steps 2 and 3 above.

2 Preliminaries

Below we define a non-interactive secure computation scheme (NISC). NISC may involve a trusted setup, an ideal implementation of some (non-reactive) functionality $\mathcal{H}$. We shall refer to such an NISC scheme as NISC/$\mathcal{H}$.

An NISC/$\mathcal{H}$ scheme for a function $f : X \times Y \to Z$ is a 2-party protocol between Receiver and Sender, of the following format:

- Receiver gets an input $x \in X$ and Sender gets an input $y \in Y$.
- The two parties invoke an instance of $\mathcal{H}$ with inputs of their choice, and Receiver obtains outputs from $\mathcal{H}$.
- Sender may send an additional message to Receiver.
- Receiver carries out a local computation and outputs $f(x, y)$ or an error message.

The correctness and secrecy requirements of an NISC scheme can be specified in terms of UC security. We shall denote the security parameter by κ and require that for a protocol to be considered secure, the simulation error be $2^{-\Omega(\kappa)}$. An NISC/$\mathcal{H}$ scheme for f is required to be a UC-secure realization of the following functionality $\mathcal{F}_f$ in the $\mathcal{H}$-hybrid model.

- $\mathcal{F}_f$ accepts $x \in X$ from Receiver and $y \in Y$ from Sender, and outputs $f(x, y)$ to Receiver and an empty output to Sender. If y is a special input error, then the output to Receiver is error.

In particular, we will be interested in NISC/OT schemes, where OT stands for a functionality that provides (parallel) access to multiple instances of $\binom{2}{1}$ Oblivious Transfer. In this case, the additional message from the sender to the receiver can be implemented using a single additional OT call.

We define a relaxed notion of security which is useful in the context of NISC, but may also be of broader interest.

Security with input-dependent abort. Given an SFE functionality $\mathcal{F}$, we define a functionality $\mathcal{F}^\dagger$ which behaves as follows: first $\mathcal{F}^\dagger$ accepts a description of a predicate ϕ from the adversary (e.g., in the form of a PPT algorithm); after receiving inputs from all the parties, $\mathcal{F}^\dagger$ computes the outputs to the parties as $\mathcal{F}$ does; but before delivering the outputs to the parties, $\mathcal{F}^\dagger$ runs ϕ with all the inputs; if ϕ outputs abort, then $\mathcal{F}^\dagger$ replaces the output to the honest parties by the message abort. Otherwise $\mathcal{F}^\dagger$ delivers the output from $\mathcal{F}$ to all the parties.

Though we defined security with input-dependent abort as a general security notion, we shall exclusively focus on 2-party functionalities $\mathcal{F}_f$ as defined above.

Security with wire-disjunction triggered abort. For a 2-party SFE functionality $\mathcal{F}$ as above, outputting $f(x, y)$ to only Receiver, we define a functionality $\mathcal{F}^\ddagger$ which is a restriction of $\mathcal{F}^\dagger$ in which the algorithm ϕ for determining aborting is restricted to be of the following form: ϕ includes a set W of pairs of the form (w, b), where w is a wire in a fixed circuit C for computing the function f, and $b \in \{0, 1\}$; $\phi(x, y) =$ abort if and only if there exists a pair $(w, b) \in W$ such that when C is evaluated on (x, y), the wire w takes the value b. We will also consider the stronger notion of *input-disjunction triggered abort* where the disjunction can only involve input wires.

Protocol making a black-box use of a PRG. We are interested in NISC/OT schemes that do not rely on any cryptographic assumption other than the security of a PRG.

Further, the scheme should be able to use any PRG provided to it, in a black-box fashion. Formally, we consider *fully black-box reductions* [41] from NISC/OT to PRG.

Towards a more concrete measure of efficiency, we require NISC/OT protocols to be $2^{-\Omega(\kappa)}$ secure and measure complexity as a function of κ and the circuit size of f. Security against corrupted senders will be statistical. To achieve the above goal against a computationally bounded corrupted receiver, we need to use a PRG for which the advantage of any PPT adversary in distinguishing the output of the PRG from a random string (of the appropriate length) is $2^{-\Omega(\kappa)}$. To this end a PRG can have a longer *computational security parameter*, k, that defines the length of its seed (k is a function of κ, but for simplicity we denote it as a separate parameter). The PRGs considered below have input and output length $\Theta(k)$.

Efficiency: Communication and Cryptographic Overheads. The best known NISC/OT scheme secure against *passive* corruption is provided by Yao's garbled circuit construction (see below) and forms the benchmark for efficiency for us. There are three aspects in which a NISC/$\mathcal{H}$ scheme can be more expensive compared to the garbled circuit (over OT) scheme:

- *The complexity of* $\mathcal{H}$. For instance, if $\mathcal{H}$ is the parallel OT functionality OT, then the number of instances of $\binom{2}{1}$ OTs and the total length of the OT strings provide natural measures of complexity of $\mathcal{H}$. (Note that a NISC/$\mathcal{H}$ scheme invokes a single instance of $\mathcal{H}$). If $\mathcal{H}$ is a more involved functionality, we shall be interested in complexity measures related to securely realizing $\mathcal{H}$.
- *Communication overhead.* We shall include in this any communication directly between the two parties and any communication between the parties and $\mathcal{H}$. We define the *communication overhead* of a NISC scheme as the ratio of total communication in the NISC scheme and the total communication in Yao's garbled circuit (over OT) scheme.
- *Cryptographic overhead.* Yao's garbled circuit scheme makes a black-box use of PRG. To evaluate a function that is computed by a circuit C, it uses $\Theta(|C|)$ calls to a PRG (with input and output lengths $\Theta(k)$). The ratio between the number of such calls made by a NISC scheme and Yao's garbled circuit scheme can be defined as the *cryptographic overhead* of the NISC scheme.

Garbled Circuit. There are two main variants of Yao's garbled circuit construction in the literature, one following the original construction of Yao [44,31] and one following the presentation in [37,1]. The former allows a negligible probability of error while evaluating the circuit (since a "wrong" key may decrypt a ciphertext encrypted using different key, without being detected), whereas the latter includes pointers with the keys indicating which ciphertexts they are intended for. In this work, we follow the latter presentation (with a few simple changes). Below we describe the version of garbled circuit we shall use.

Consistent with our use of garbled circuits, we shall refer to two parties Receiver (with input x) and Sender (with input y) who wish to evaluate a function represented as a boolean circuit C, with binary gates. Given such a circuit C and input y for Sender, the garbled circuit Y_C for C is constructed as follows. For each (non-output) wire w in C, two k-bit strings K_w^0 and K_w^1 are randomly chosen, as well as a random bit r_w. The random bit r_w is used to mask the values on the wires during evaluation of the

garbled circuit (if w is an output wire, or an input wire belonging to Receiver, then we set $r_w = 0$). The garbled circuit Y_C consists of the following:

- For each gate g (with input wires u, v and output wire w), for each $a, b \in \{0,1\}$, an encryption
$$\text{Encr}^{g,a,b}_{K_u^{a\oplus r_u}, K_v^{b\oplus r_v}}(K_w^{c\oplus r_w}, c)$$
where $c = \tilde{F}_g(a,b) := F_g(a \oplus r_u, b \oplus r_v) \oplus r_w$ where u, v are the input wires to gate g and w its output wire,[4] and Encr is a symmetric-key encryption scheme with a mild one-time semantic security guarantee (see below).
- For each input-wire w for which the value is already fixed (i.e., w is an input wire that belongs to Sender), the pair $(K_w^{y_w}, y_w \oplus r_w)$, where y_w is the value of that input wire.

Note that if gate g has input wires u, v and output wire w, then for any values (α, β), given (K_u^x, a) and (K_v^y, b) where $a = \alpha \oplus r_u$ and $b = \beta \oplus r_v$, one can obtain (K_w^z, c), where $z = F_g(\alpha, \beta)$ and $c = z \oplus r_w$. Hence, given Y_C and $(K_w^{x_w}, x_w)$ for each input-wire w belonging to Receiver (note that for such w, $x_w \oplus r_w = x_w$), one can evaluate $C(x, y)$ (note that an output wire w has $r_w = 0$).

The privacy guarantee of a garbled circuit is that, given x and $f(x, y)$, it is possible to construct a simulation $(\tilde{Y_C}, \tilde{K}_1, \ldots, \tilde{K}_{|x|})$ which is computationally indistinguishable (with at most a distinguishing advantage $2^{-\Omega(\kappa)}$) from $(Y_C, K_1, \ldots, K_{|x|})$ where Y_C is the garbled circuit constructed for Sender's input y, and K_i are keys of the form $K_w^{x_w}$ where w are the input wires for Receiver.

For the above security guarantee to hold, encryption Encr only needs to be *one-time* semantically secure (even if one of the two keys is revealed) [1]. But it is convenient for us to restrict ourselves to a concrete instantiation of an encryption scheme[5]. A particular instance of an encryption scheme, that satisfies the requirements for a garbled circuit, can be defined in terms of black-box access to a pseudorandom generator or, more conveniently, in terms of a pseudorandom function. For each key K in $\{0,1\}^k$, let the pseudorandom function initialized with the key K be denoted by $R_K : [|C|] \to \{0,1\}^{k'}$ where $[|C|]$ is the set of indices for the gates in the circuit C and k' is the maximum length of the messages to be encrypted ($k + 1$ above, but longer in the variants used in some of our schemes)[6]. Then the encryption Encr is defined as follows:

$$\text{Encr}^t_{K_1,K_2}(m) = m \oplus R_{K_1}(t) \oplus R_{K_2}(t), \tag{1}$$

where $t = (g, a, b)$ is a "tag" that is used once with any key.

[4] In fact, g may have more than one output wire; in such case, they are all assigned the same keys K_w^0 and K_w^1.

[5] The construction and analysis in [1] admits any one-time semantically secure symmetric-key encryption scheme. Our construction here is somewhat more streamlined since we use a specific encryption scheme.

[6] Note that the domain of the inputs for PRF is small, and hence a PRG with an appropriately long output can be used to directly implement the PRF. In fact, the PRF will be invoked on only as many inputs as the maximum fan-out of the circuit, and needs to be defined only for the values on which it will be invoked. Nevertheless we shall use the PRF notation for convenience and conceptual clarity.

3 A Statistical NISC/OT Protocol for $\mathbf{NC}^0$

A central building block of our protocols for general functions f is a statistically secure protocol for "simple" functions g in which each output bit depends on just a small number of input bits. We say that $g(x, y)$ is d-local if each of its output bits depends on at most d input bits. Towards an asymptotic measure of efficiency, we will (implicitly) consider an infinite family of finite functionalities $g_n : \{0,1\}^{\alpha_n} \times \{0,1\}^{\beta_n} \to \{0,1\}^{\gamma_n}$. We say that such a family is in $\mathbf{NC}^0$ if there exists an absolute constant d such that each g_n is d-local. The precise locality d of the $\mathbf{NC}^0$ functions we will employ is small, and will be hidden in the asymptotic analysis.

Our general protocols for evaluating a function $f(x, y)$ will typically require the evaluation of $\mathbf{NC}^0$ functions $g(a, b)$ where the receiver's input a is short (comparable in length to x) but the sender's input b and the output are long (comparable to the circuit size of f). We would therefore like the number of OT invocations to be comparable to the receiver's input length α, and the total communication complexity (in the OT-hybrid model) to be as close as possible to the output length γ.

The semi-honest model. In the semi-honest model, there are several known techniques for obtaining *perfectly* secure protocols that meet these requirements (cf. [21] and references therein): in such protocols the number of OTs is exactly α and the total communication complexity is $O(\gamma)$ (with a hidden multiplicative constant depending at most exponentially on d). Our goal is to get similar efficiency in the malicious model without introducing additional interaction.

Previous results. A statistically secure NISC/OT protocol for $\mathbf{NC}^0$ functions in the malicious model is implicit in [28]. (Via known reductions, this can be extended to functions in low complexity classes such as $\mathbf{NC}^1$ with a polynomial complexity overhead). A more efficient protocol was given in [24] (see Appendix B of [23]). The protocol from [24] can also provide computational security for general functions, but this requires a *non-black-box* use of a pseudorandom generator. From here on we focus on the case of $\mathbf{NC}^0$ functions.

The protocol of [24] is based on a reduction to *multi-party* computation (MPC) in the *semi-honest* model, in the spirit of the MPC-based zero-knowledge protocols of [20]. Instantiated with standard MPC protocols, and settling for a relaxed notion of security, discussed in Section 3.2 below, its communication complexity is $\Theta(\gamma \cdot \kappa)$, where γ is the output length of f and κ is a statistical security parameter guaranteeing simulation error of $2^{-\kappa}$. (Here and in the following we will assume that $\gamma \gg \alpha, \kappa$ and ignore low order terms in the efficiency analysis for simplicity).

3.1 Overview of New Protocol

We present a different approach for NISC/OT that reduces the multiplicative overhead from $\Theta(\kappa)$ to $\mathrm{polylog}(\kappa)$. Our general approach employs perfectly secure MPC protocols for the *malicious* model. The efficiency improvement will be obtained by plugging in the recent perfectly secure protocol from [10].

Given an $\mathbf{NC}^0$ function $g(a, b)$, where $g : \{0,1\}^{\alpha} \times \{0,1\}^{\beta} \to \{0,1\}^{\gamma}$, our construction has a similar high level structure to that of [24,23]:

1. Start with a perfectly secure NISC/OT protocol π for g in the *semi-honest* model in which the receiver uses its original α input bits a as the sequence of OT choices. Several such protocols with a constant overhead can be found in the literature (see [21] and references therein).
2. Use the sender's algorithm in π to define a "certified OT" functionality COT, which is similar to parallel OT except that it verifies that the α pairs of strings (together with an additional witness) provided by the sender satisfy a given global consistency predicate. If this verification fails, a special error message $\perp$ is delivered to the receiver.

   Concretely, we will employ a COT functionality in which the sender's witness includes its randomness and its input b, and the predicate verifies that the α pairs of strings are as prescribed by the protocol. (For efficiency reasons, it may be useful to include in the witness the values of intermediate wires in the sender's computation. This standard technique can be used to transform an arbitrary predicate into one in $\mathbf{NC}^0$).
3. Take a *perfectly secure* MPC protocol Π_{COT} for a multi-party functionality corresponding to COT, and use it to obtain a statistically secure two-party NISC/OT protocol π_{COT} for COT. This is the main novel contribution of the current section, which will be described in detail below.
4. Use π_{COT} for obtaining an NISC/OT protocol π_g for g with security in the malicious model. This can be done in a straightforward way by using COT to emulate π while ensuring that the sender does not deviate from its prescribed algorithm. Note that the protocol makes a non-black-box use of π, and thus in our black-box setting we cannot apply it to protocols π which make use of cryptographic primitives.

3.2 Relaxing Security

A (standard) technical subtlety that we need to address is that our direct implementation of π_{COT} will not realize the functionality COT under the standard notion of security, but rather under a relaxed notion of security that we refer to as security with "input-value disjunction (IVD) abort". This is similar to the notion of security with wire-value disjunction (WVD) abort from Section 6, except that here the disjunctive predicate applies only to input values. That is, the ideal functionality is augmented by allowing a malicious sender to specify a disjunctive predicate in the receiver's input bits (such as $x_2 \vee \bar{x}_4 \vee x_7$) which makes the functionality deliver $\perp$ if the receiver's input satisfies the predicate. (Otherwise the output of the original functionality is delivered).

A standard method for upgrading security with IVD-abort into full security is by letting the receiver "secret-share" its input (cf. [28, 32]). Concretely, the receiver encodes x into a longer input x' in a way that ensures that every disjunctive predicate in x' is either satisfied with overwhelming probability, or alternatively is completely independent of x. The original functionality g is then augmented to a functionality h that first decodes the original input and then computes g. (To prevent cheating by a malicious receiver, the decoding function should output a valid input x for any string x').

One can now apply any protocol π_h for h which is secure with IVD-abort in order to obtain a fully secure protocol for the original functionality g. We note that the functionality h will not be in $\mathbf{NC}^0$; thus, the overhead for realizing it unconditionally (even in the semi-honest model) will be too big for our purposes. Instead, we apply the security

boosting reduction only at higher level protocols which offer computational security and rely on Yao's garbled circuit construction. For such protocols, we only pay an *additive* price comparable to the circuit *size* of the decoder, which we can make linear in the input length.

We finally suggest a concrete method to encode x into x' as above. A simple method suggested in [28,32] is to let x' be an additive sharing of x into $\kappa + 1$ shares (over $\mathcal{F}_2^\alpha$). This has the disadvantage of increasing the length of x by a factor of κ, which we would like to avoid. Better alternatives were suggested in the literature (see, e.g., [40]) but these still increase the input length by a constant factor and significantly increase the circuit size. Instead, we suggest the following encoding method. Let $G : \{0,1\}^\delta \to \{0,1\}^\alpha$ be a κ-wise independent generator. That is, for a random r, the bits of $G(r)$ are κ-wise independent. Then the encoding is defined by $Enc(x) = (r_1, \ldots, r_{\kappa+1}, x \oplus G(r_1 \oplus \cdots \oplus r_{\kappa+1}))$ where the r_i are uniformly random strings of length δ. The corresponding decoder is defined by $Dec(r_1, \ldots, r_{\kappa+1}, z) = z \oplus G(r_1 \oplus \cdots \oplus r_{\kappa+1})$.

The following lemma is straightforward.

Lemma 1. *For every disjunctive predicate $P(x')$, the following holds: (1) If P involves at most κ literals, then $\Pr[P(Enc(x)) = 1]$ is completely independent of x. (2) Otherwise, $\Pr[P(Enc(x)) = 1] \geq 1 - 2^{-\kappa}$.*

We note that efficient implementations of G can be based on expander graphs [35]. In particular, for any constant $0 < c < 1$ there is an $\mathbf{NC}^0$ implementation of G (with circuit size $O(\alpha)$) where $\delta = \alpha^c + poly(\kappa)$. Thus, in the typical case where $\alpha \gg \kappa$, the encoding size is $\alpha + o(\alpha)$.

The following corollary shows that, from an asymptotic point of view, boosting security with IVD-abort into full security comes essentially for free both in terms of the circuit size and the receiver's input length.

Corollary 1. *Let $f(x, y)$ be a functionality with circuit size s and receiver input size $\alpha = |x|$. Then, there exists a functionality $h(x', y)$ and a linear-time computable encoding function Enc such that:*

- *A fully secure protocol π_f for f can be obtained from any protocol π_h for h which is secure with IVD-abort by letting the parties in π_f run π_h with inputs $x' = Enc(x)$ and y.*
- *The circuit size of h is $s + O(\alpha) + \mathrm{poly}(\kappa)$.*
- *The receiver's input length in h is $\alpha + o(\alpha) + \mathrm{poly}(\kappa)$.*

3.3 Realizing COT via Robust MPC

It remains to describe an efficient protocol π_{COT} for COT which is secure with IVD-abort. In this section, we reduce this task to perfectly robust MPC in the presence of an honest majority.

We consider an MPC network which involves a sender S, n servers P_i, and 2α receivers $R_{i,b}$, $1 \leq i \leq \alpha$, $b \in \{0,1\}$; for simplicity, we assume that receivers do not send, but only receive messages in the protocol. (We will later set $n = O(\kappa\alpha)$). All parties are connected via secure point-to-point channels as well as a common broadcast medium. Define the following multi-party version of COT: the sender's input consists

of α pairs of strings $(y_{i,0}, y_{i,1})$ and a witness w. The other players have no input. The output of receiver $R_{i,b}$ is $\perp$ if $P(\{y_{i,b}\}, w) = 0$, and otherwise it is $y_{i,b}$.

Now, assume we are given an MPC protocol Π_{COT} that realizes this multiparty COT functionality and provides the following security guarantees. The adversary may attack up to $t = \Omega(n)$ of the servers, as well as any number of the other players (sender and receivers). For such an adversary, the protocol provides perfect correctness and, moreover, if the adversary is semi-honest we are also guaranteed *privacy*. Such a protocol, with the desired complexity, appears in [10]. We now use Π_{COT} to construct a COT protocol π_{COT} as follows.

1. Sender runs the MPC protocol Π_{COT} "in his head" (a-la [20]), where its input (α pairs of strings $(y_{i,0}, y_{i,1})$ and a witness w) serve as inputs for the sender S of Π_{COT}. It creates strings $V_1, \ldots, V_n$ with the views of the n servers in this run, as well as $V_{1,0}, V_{1,1}, \ldots, V_{\alpha,0}, V_{\alpha,1}$ with the views of the 2α receivers.
2. Let u be an integer such that $1/u \in [t/2n, t/4n]$. The sender and the receiver apply one parallel call to an OT in which the receiver selects, for each $i \in [n]$, a view V_{i,b_i} (where the n selection bits $b_i \in \{0,1\}$ are the COT-receiver input) as well as each of the n server views with probability $1/u$[7].
3. Receiver checks for inconsistencies among the views that it read (namely, for each pair of views V_A, V_B, corresponding to players A, B, all messages from A to B reported in V_B should be consistent with what an honest A computes in Π_{COT} based on V_A). If any such inconsistency is found or if any of the α selected receiver views has a $\perp$ output, then the receiver outputs $\perp$; otherwise, the receiver outputs the output of the α selected receivers.

To analyze the protocol above, first note that if both Sender and Receiver are honest then the output of protocol π_{COT} is always correct (in particular, because so is Π_{COT}).

Next, consider the case of a dishonest Receiver (and honest Sender). Since, we use ideal OTs the Receiver can only choose it's selection bits which will yield exactly one of $V_{i,0}, V_{i,1}$ (for each $i \in [n]$) and each of the n server views with probability $1/u$. By the choice of u, the expected number of server views that the receiver will obtain, denoted ℓ, is $n/u \leq t/2$ and, moreover, only with a negligible probability $\ell > t$. Whenever $\ell \leq t$, the privacy property of Π_{COT} assures that from (semi-honest) views of ℓ servers and any number of receivers, no additional information (other than what the output of those receivers contain) is learned about the input of the Sender.

Finally, consider the case of a dishonest Sender (and honest Receiver). The COT simulator, given the Sender's view (in particular, the views of the MPC players), constructs the inconsistency graph G, whose nodes are the MPC players and an edge between nodes A, B whenever the corresponding views are inconsistent. In addition, G' is the sub-graph induced by the n nodes corresponding to the servers. The simulator starts by

[7] This is based on [24] which can be done non-interactively in our model. The observation is that it is known that $\binom{u}{1}$-OT non-interactively reduces to $u-1$ instances of $\binom{2}{1}$-OT. Now, given $\binom{u}{1}$-OT, a string can be transferred with probability $1/u$ simply by letting the sender put the string in a random location i of a u-entry array, and send to the receiver (independently of the receiver's selection) an additional message with i. Also note, that with our choice of parameters $u = O(1)$.

running a polynomial-time 2-approximation (deterministic) algorithm for finding minimal vertex-cover in the graph G'; i.e, the algorithm outputs a vertex cover B whose size is not more than twice the size of a minimal vertex-cover B^*. Consider two case, according to the size of B.

Case 1: $|B| > t$. In this case the simulator outputs $\perp$ with probability 1; we argue that in the real COT protocol, the receiver outputs $\perp$ with probability negligibly less than 1. This is because $|B^*| \geq |B|/2 > t/2$ and so there must be a matching in G' of size larger than $t/4$ (the size of a minimal vertex-cover of a graph is at most twice the size of a maximal matching). This, together with the choice $t = \Omega(n)$, implies that the probability that the ℓ servers picked by the Receiver do not contain an edge of G' is $2^{-\Theta(n)}$. In all other cases, the Receiver outputs $\perp$. (A similar argument was made in [20]; for more details, see there).

Case 2: $|B| \leq t$. In this case, the COT simulator passes the views of the MPC sender and of all servers in B to the MPC simulator. The MPC simulator extracts an effective sender input (i.e., α pairs of strings and a witness w). If this input does not satisfy the predicate P then output $\perp$ (by the perfect correctness of Π_{COT}, on such input π_{COT} always outputs $\perp$ as well). It remains to deal with the case where the predicate does hold. For this, the COT simulator picks each server with probability $1/u$ (as does the honest receiver in π_{COT}) and if there is any inconsistency among the set T of selected views then the receiver outputs $\perp$; otherwise, the simulator also compares the view of each of the 2α receivers with each of the servers in T. It prepares a disjunctive predicate, P_d, consisting of the literals corresponding to receivers which have at least one such inconsistency (i.e., the predicate is satisfied exactly if the Receiver will select any of the problematic views; in both cases this leads to a $\perp$ output). It sends to the functionality the input extracted by the simulator along with the predicate P_d.

To conclude, let us summarize the complexity of our construction and compare it with the one in [23, Appendix B] (essentially the two constructions are incomparable with advantages depending on the spectrum of parameters).

Theorem 1. *The above protocol is a secure protocol* with IVD abort *for computing any* $\mathbf{NC}^0$ *function* $g(a, b)$*, where* $g : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\gamma$*. Its communication complexity is polylog*$(\kappa) \cdot \gamma +$ poly(α, κ)*. (Recall that* $n = O(\kappa\alpha)$*). The number of OT calls is* $O(\alpha\kappa)$*.*

Theorem 2. *[23] There exists a secure protocol* with IVD abort *for computing any* $\mathbf{NC}^0$ *function* $g(a, b)$*, where* $g : \{0,1\}^\alpha \times \{0,1\}^\beta \to \{0,1\}^\gamma$ *whose communication complexity is* $O(\kappa\gamma)$ *and number of OT calls is* $O(\alpha + \kappa)$*.*

4 A Direct Protocol for NISC/NC^0

Our first construction follows a cut-and-choose approach in the spirit of previous constant-round protocols making black-box access to cryptographic primitives [34,32]. The price we pay for this relatively simple solution is $O(\kappa)$ cryptographic and communication overheads. In particular, we show the following.

Theorem 3. *For any function* $f : X \times Y \to Z$ *that has a polynomial sized circuit* C *with* n *input wires for the first input, there exists an* $\mathbf{NC}^0$ *functionality* $\mathcal{H}_C$ *with*

$O(\kappa k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C$ that makes a black-box use of a PRG, invoking the PRG $O(\kappa|C|)$ times, and with $O(\kappa k|C|)$ total communication. (Recall that κ is a statistical security parameter and k is a computational one).

We shall defer the proof of this theorem to Section 8, where a more general result is presented (see Theorem 6).

5 A Lean NISC/NC^0 Protocol with Input-Dependent Abort

In this section, we present a NISC scheme for $\mathcal{F}_C^\dagger$, which allows input-dependent abort. This scheme is very efficient: the communication overhead over the garbled circuit scheme is (a small) constant and the cryptographic overhead is just 1 (allowing the PRGs to output a slightly longer string). We shall present the scheme first as a NISC/$\mathcal{H}_C$ scheme, for an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$, and then apply the result of Section 3 to obtain an NISC/OT scheme.

Theorem 4. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$ with $O(\kappa|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^\dagger$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

PROOF SKETCH: The details of the proof appears in the full version of this paper. At a high-level, this scheme allows Receiver to verify that each pointer bit uncovered in the garbled circuit is correct as follows: each pointer bit is tagged using a MAC (with a key provided by Receiver). However since this bit should be kept secret until the corresponding entry in the garbled circuit is decrypted, a share of the tag is kept encrypted with the pointer bit, and the other share is provided to Receiver. Sender, who does not know the MAC key, can create the one share that he must encrypt, and an $\mathbf{NC}^0$ functionality takes the MAC key from Receiver, computes the MAC tag and hands over the other share to Receiver. Input dependent abort is obtained since, intuitively, the sender can only use wrong MACs in some entries which will make the Receiver abort in case those entries are decrypted. □

6 NISC/NC^0 with Wire-Disjunction Triggered Abort

We extend the scheme in Section 5, to achieve the stronger security guarantee of security with wire-disjunction triggered abort. Similar to the previous scheme, this scheme ensures (partial) correctness of the garbled circuit via an $\mathbf{NC}^0$ functionality which provides the share of a MAC to Receiver. However, the MAC is not just on a single pointer bit, but also on the key stored in the garbled circuit entry. This scheme has some features of the scheme in Section 4 in that Sender provides a table of purported outputs from a PRF, some of which will be verified by Receiver during decoding. However, this construction avoids the $O(\kappa)$ overhead, at the expense of settling for security with *wire-disjunction triggered abort*.

This construction involves a statistically secure, one-time MAC for k bit messages. It will be important for us to implement this MAC scheme using $\mathbf{NC}^0$ circuits. This can be done following [21], if the message is first encoded appropriately. Since the encoding itself is not an $\mathbf{NC}^0$ functionality, we require Sender to provide the encoding, along with values of all the wires in a circuit that computes the encoding. Then an $\mathbf{NC}^0$ circuit can verify this encoding, and in parallel create the MAC tag.

In the full version we prove the following theorem.

Theorem 5. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an $\mathbf{NC}^0$ functionality $\mathcal{H}_C$ with $O(k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, such that there is an NISC/$\mathcal{H}_C$ scheme for $\mathcal{F}_C^{\ddagger}$ that makes a black-box use of a PRG, invoking the PRG $O(|C|)$ times, and with $O(k|C|)$ total communication.*

Compared to Theorem 4, this construction is asymptotically less efficient, since the output of $\mathcal{H}_C$ is longer ($O(k|C|)$ instead of $O(\kappa|C|)$, as $\mathcal{H}_C$ will now be required to deliver the entire garbled srcuit to Receiver).

7 From Security with WDT-Abort to Full Security

In this section, we discuss general methods for converting any NISC scheme satisfying security with *wire disjunction triggered* (WDT) abort into an NISC with full security, based on semi-honest secure MPC protocols. Our transformation formalizes and generalizes such a transformation that was given in the work of [25,26] (and our intuition below follows their intuition) in the context of constructing stateful hardware circuits that remain private even when an adversary can tamper with the values on wires. We note that the construction of [25] also had to deal with multiple other issues that do not concern us, which added complexity to their solution. Outlined below, our solution can be seen as a simplification of their construction.

The benefit of the transformation outlined in this section over the simple majority-based approach discussed earlier is the potential for greater efficiency. We will first formalize the encoding notion that we use to deal with WDT attacks, then we present an outline of our general transformation, and then show how to invoke this transformation using known semi-honest MPC protocols from the literature to obtain higher levels of efficiency.

Our transformation is captured by means of a new encoding, that we define below. The details of realizing this transformation are presented in the full version.

Definition 1. ***(WDT-resilient encoding)*** *A randomized circuit family C' together with an efficient randomized encoding algorithm family Enc and an efficient deterministic decoding algorithm family Dec is a* WDT-resilient encoding *of a circuit C that takes two inputs if the following properties hold*[8]*:*

[8] The entire tuple (C', Enc, Dec) is parameterized by a statistical security parameter 1^κ, which is omitted here for simplicity of notation. Note also that this definition defines the notion of a WDT-resilient encoding. In applications, we will require that there is an efficient deterministic procedure that takes as input C and 1^κ and outputs a tuple (C', Enc, Dec) from such a family.

(Correctness). For all (x, y) in the domain of C, we have that

$$\Pr[Dec(C'(Enc(x), y)) = C(x, y)] = 1$$

(Malicious Receiver Security). There exists a randomized efficient machine $RecSim$ such that for every x' in the range of Enc (but not necessarily in the image of Enc), there exists x in the domain of Enc such that for every y such that (x, y) is in the domain of C, the output distribution of $RecSim(x', C(x, y))$ is identical to the distribution $C'(x', y)$.

(WDT-Malicious Sender Security). For any set S of wires in C' or their negations, let $Disj_S[C'(Enc(x), y)]$ to be the event that the disjunction of the values specified by S, when the input of C' is $(Enc(x), y)$, is satisfied. The probability space is over the random gates of C' and the randomness used by Enc.
For any such S and for all x_1, x_2, and y such that (x_1, y) and (x_2, y) are in the domain of C, we have:

$$|\Pr[Disj_S[C'(Enc(x_1), y)]] - \Pr[Disj_S[C'(Enc(x_2), y)]]| = 2^{-\Omega(\kappa)}.$$

8 Public-Code NISC

So far we only considered NISC schemes which rely on an OT oracle that gets inputs from both the sender and the receiver. As discussed in the introduction, this can be combined with a 2-message implementation of OT to get a protocol which does not require any active action from the receiver except publishing an encryption of her input.

In this section we discuss this variant of NISC, called Public-Code NISC or PC-NISC for short. In more detail, this flavor of NISC allows Receiver to publish an encoding of her input x, and later let one or more Senders compute on the encoding of x using their private inputs y, and send it back to her; she can decode this message and recover the value $f(x, y)$ (and nothing more). There could be a setup like a common reference string (CRS), or correlated random variables.

Formally, a PC-NISC scheme for a function $f : X \times Y \to Z$ consists of the following four PPT algorithms.

- Setup: takes only the security parameter as an input and outputs a pair of strings (σ^R, σ^S). These strings are meant to be given to the two parties (Receiver and Sender, respectively).
- Encode: takes an input $x \in X$, and a setup string σ^R, and outputs a string c encoding x (or possibly an error message, if σ^R appears malformed).
- Compute: takes an encoding c, an input $y \in Y$ and a setup string σ^S and outputs an "encoded output" (or an error message if c appears malformed).
- Decode: takes an encoded output and a setup string σ^R, and outputs $z \in Z$ (or an error message if the encoded output appears malformed).

Ideally, in a PC-NISC scheme, a single published encoding can be used by Receiver to carry out multiple computations. To define the security of a PC-NISC scheme, below we define the functionality $\mathcal{F}_f^{(T)}$, which allows T invocations before letting a corrupt Sender manipulate the outcome.

- $\mathcal{F}_f^{(T)}$ accepts an input x from Receiver.
- Then in each round, it accepts an input y from Sender. and outputs $f(x, y)$ to Receiver (and an empty output to Sender). If y is a special command error, the output to Receiver is error.
- There is a bound T on the number of inputs $\mathcal{F}_f^{(T)}$ accepts from corrupt Senders before correctness is compromised. More formally, a corrupt Sender is allowed to include with its input a command (cheat, ψ) where ψ is an arbitrary PPT algorithm, and after T such rounds, in each subsequent such round, $\mathcal{F}_f^{(T)}$ outputs $\psi(x)$ to Receiver.

Now, given a PC-NISC scheme Σ consider the 2-party protocol Π_Σ (in a $\mathcal{F}_{\Sigma.\mathsf{Setup}}$-hybrid model, which simply makes a fresh pair (σ^R, σ^S) available to the two parties) in which Receiver, on input x, sends $c := \Sigma.\mathsf{Encode}(x, \sigma^R)$ to Sender; on receiving an input y reactively from the environment, Sender sends $u = \Sigma.\mathsf{Compute}(c, y, \sigma^S)$ to Receiver, and Receiver outputs $\Sigma.\mathsf{Decode}(u)$. We say that Σ is a secure PC-NISC scheme if the protocol $\Pi_\Sigma^{\mathcal{F}_{\Sigma.\mathsf{Setup}}}$ is a UC secure realization of the functionality $\mathcal{F}_f^{(T)}$.

We shall be interested in NISC schemes for $\mathcal{F}_f^{(T)}$, where $T = \Omega(\kappa)$.

Defining PC-NISC/$\mathcal{H}$. The goal of PC-NISC was to avoid the live availability of Receiver, when Sender is executing the scheme. However it is still possible to consider such a scheme in an $\mathcal{H}$-hybrid model, if the functionality $\mathcal{H}$ itself allows Receiver to send an input, and subsequently have multiple rounds of independent interactions with Sender, delivering a separate output to Receiver in each round. We shall use this convention as an intermediate step in achieving PC-NISC/OT and PC-NISC/CRS schemes, which can be specified in the plain model (i.e., without reference to a hybrid-model) in terms of the Setup algorithm. In PC-NISC/CRS, Setup sets $\sigma^R = \sigma^S$ to be a randomly generated string, according to some probability distribution that will be specified by the scheme.

In PC-NISC/OTvar, Setup outputs several instances of correlated random variables: in each instance, Receiver gets two random bits (a_0, a_1) and Sender gets random bits (b_0, b_1) such that $a_0 b_0 = a_1 \oplus b_1$[9]. They can be readily used in a PC-NISC scheme Σ_0 for evaluating the OT function, in which Receiver has a choice bit c, Sender has two inputs x_0 and x_1, and Receiver obtains x_c. Hence a NISC/OT scheme for a function f can be easily turned into a PC-NISC/OTvar scheme for f *if the number of sessions to be supported* $T = 1$: the Encode and Compute algorithms will incorporate $\Sigma_0.\mathsf{Encode}$ and $\Sigma_0.\mathsf{Compute}$; further, Compute will include the message sent by Sender in the NISC/OT scheme; Decode involves first applying $\Sigma_0.\mathsf{Decode}$ to obtain the outcome of OT, before carrying out the local computation of the NISC/OT scheme.

The main challenge in constructing a PC-NISC scheme, beyond that already present in constructing NISC schemes, is to be able to support a larger number of computations for the same input encoding.

First, we observe that the NISC/OT scheme for $\mathbf{NC}^0$ functionalities from Section 3 can be extended into a PC-NISC/OTvar supporting T adding a $poly(\kappa, T)$ amount to communication and cryptographic complexities. This is done by increasing the number of servers in the underlying MPC used in this scheme.

[9] There are several equivalent formulations of such a pair of correlated random variables.

In the full version we prove the feasibility result below, analogous to – indeed extending – Theorem 3.

Theorem 6. *For any function $f : X \times Y \to Z$ that has a polynomial sized circuit C with n input wires for the first input, there exists an* $\mathbf{NC}^0$ *functionality $\mathcal{H}_C^{(T)}$ with $O(\kappa k|C|)$-bit long output and $n + O(\kappa)$-bit input from Receiver, supporting T computations, such that there is a NISC/$\mathcal{H}_C^{(T)}$ scheme for $\mathcal{F}_f^{(T)}$ that makes a black-box use of a PRG, invoking the PRG $O((\kappa + T)|C|)$ times, and with $O((\kappa + T)k|C|)$ total communication.*

Note that the above NISC scheme is already for $\mathcal{F}_f^{(T)}$, and can be translated to a PC-NISC scheme for f supporting T executions, as described earlier. Thus, given this scheme, we can combine it with a PC-NISC/OTvar for $\mathcal{H}_C^{(T)}$ (also described above) to obtain a PC-NISC/OTvar for $\mathcal{F}_f^{(T)}$. A proof of Theorem 6 is given in the full version.

References

1. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. In: IEEE Conference on Computational Complexity, pp. 260–274. IEEE Computer Society, Los Alamitos (2005)
2. Beaver, D.: Precomputing Oblivious Transfer. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 97–109. Springer, Heidelberg (1995)
3. Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: Proc. 28th STOC, pp. 479–488. ACM, New York (1996)
4. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 589–590. Springer, Heidelberg (1990)
5. Beaver, D., Micali, S., Rogaway, P.: The round complexity of secure protocols (extended abstract). In: STOC, pp. 503–513. ACM, New York (1990)
6. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
7. Cachin, C., Camenisch, J., Kilian, J., Müller, J.: One-Round Secure Computation and Secure Autonomous Mobile Agents. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 512–523. Springer, Heidelberg (2000)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016 (2001), Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001 (2001)
9. Chung, K.-M., Kalai, Y., Vadhan, S.P.: Improved delegation of computation using fully homomorphic encryption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 483–501. Springer, Heidelberg (2010)
10. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly Secure Multiparty Computation and the Computational Overhead of Cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010)
11. Damgård, I., Nielsen, J.B., Orlandi, C.: Essentially Optimal Universally Composable Oblivious Transfer. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 318–335. Springer, Heidelberg (2009)

12. Gennaro, R., Gentry, C., Parno, B.: Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
13. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178. ACM, New York (2009)
14. Gentry, C., Halevi, S., Vaikuntanathan, V.: *i*-hop homomorphic encryption and rerandomizable yao circuits. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 155–172. Springer, Heidelberg (2010)
15. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press, Cambridge (2004)
16. Goldreich, O., Micali, S., Wigderson, A.: How to play ANY mental game. In: ACM (ed.) Proc.19th STOC, pp. 218–229. ACM, New York (1987), See [15, ch. 7] for more details
17. Horvitz, O., Katz, J.: Universally-Composable Two-Party Computation in Two Rounds. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 111–129. Springer, Heidelberg (2007)
18. Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending Oblivious Transfers Efficiently. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 145–161. Springer, Heidelberg (2003)
19. Ishai, Y., Kushilevitz, E.: On the Hardness of Information-Theoretic Multiparty Computation. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 439–455. Springer, Heidelberg (2004)
20. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: STOC, pp. 21–30. ACM, New York (2007)
21. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Cryptography with constant computational overhead. In: STOC, pp. 433–442. ACM, New York (2008)
22. Ishai, Y., Paskin, A.: Evaluating Branching Programs on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
23. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer - efficiently, Preliminary full version on http://www.cs.uiuc.edu/~mmp/
24. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008)
25. 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)
26. 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)
27. Kalai, Y.T., Raz, R.: Succinct non-interactive zero-knowledge proofs with preprocessing for logsnp. In: FOCS, pp. 355–366. IEEE, Los Alamitos (2006)
28. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC, pp. 20–31. ACM, New York (1988)
29. Kilian, J., Micali, S., Ostrovsky, R.: Minimum resource zero-knowledge proofs (extended abstract). In: FOCS, pp. 474–479. IEEE, Los Alamitos (1989)
30. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373. IEEE, Los Alamitos (1997)
31. Lindell, Y., Pinkas, B.: A proof of yao's protocol for secure two-party computation. Electronic Colloquium on Computational Complexity (ECCC) (063) (2004)
32. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)

33. Melchor, C.A., Gaborit, P., Herranz, J.: Additively homomorphic encryption with d-operand multiplications. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 138–154. Springer, Heidelberg (2010)
34. Mohassel, P., Franklin, M.K.: Efficiency tradeoffs for malicious two-party computation. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 458–473. Springer, Heidelberg (2006)
35. Mossel, E., Shpilka, A., Trevisan, L.: On epsilon-biased generators in nc^0. Random Struct. Algorithms 29(1), 56–81 (2006)
36. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA, pp. 448–457 (2001)
37. Naor, M., Pinkas, B., Sumner, R.: Privacy preserving auctions and mechanism design. In: ACM Conference on Electronic Commerce, pp. 129–139 (1999)
38. Nielsen, J.B., Orlandi, C.: LEGO for Two-Party Secure Computation. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 368–386. Springer, Heidelberg (2009)
39. 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)
40. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
41. 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)
42. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphisms. In: Foundations of Secure Computation (Workshop, Georgia Inst. Tech., Atlanta, Ga., 1977), pp. 169–179. Academic, New York (1978)
43. Sander, T., Young, A., Yung, M.: Non-interactive cryptocomputing for NC^1. In: FOCS, pp. 554–567 (1999)
44. Yao, A.C.-C.: How to generate and exchange secrets. In: Proc. 27th FOCS, pp. 162–167. IEEE, Los Alamitos (1986)

Towards a Game Theoretic View of Secure Computation

Gilad Asharov[1,⋆], Ran Canetti[2,⋆⋆], and Carmit Hazay[3]

[1] Department of Computer Science, Bar-Ilan University, Israel
asharog@cs.biu.ac.il
[2] Department of Computer Science, Tel-Aviv University, Israel
canetti@tau.ac.il
[3] Department of Computer Science, Aarhus University, Denmark
carmit@cs.au.dk

Abstract. We demonstrate how Game Theoretic concepts and formalism can be used to capture cryptographic notions of security. In the restricted but indicative case of two-party protocols in the face of malicious fail-stop faults, we first show how the traditional notions of secrecy and correctness of protocols can be captured as properties of Nash equilibria in games for rational players. Next, we concentrate on fairness. Here we demonstrate a Game Theoretic notion and two different cryptographic notions that turn out to all be equivalent. In addition, we provide a simulation based notion that implies the previous three. All four notions are weaker than existing cryptographic notions of fairness. In particular, we show that they can be met in some natural setting where existing notions of fairness are provably impossible to achieve.

1 Introduction

Both Game Theory and the discipline of cryptographic protocols are dedicated to understanding the intricacies of collaborative interactions among parties with conflicting interests. Furthermore, the focal point of both disciplines is the same, and is algorithmic at nature: designing and analyzing algorithms for parties in such collaborative situations. However, the two disciplines developed very different sets of goals and formalisms. Cryptography focuses on designing algorithms that allow those who follow them to interact in a way that guarantees some basic concrete properties, such as secrecy, correctness or fairness, in face of adversarial, malicious behavior. Game Theory is more open-ended, concerning itself with understanding algorithmic behaviors of "rational" parties with well-defined goals in a given situation, and on designing rules of interaction that will "naturally" lead to behaviors with desirable properties.

Still, in spite of these differences, some very fruitful cross fertilization between the two disciplines has taken place (see e.g. [26,8]). One very natural

⋆ Supported by the European Research Council as part of the ERC project LAST.
⋆⋆ Supported by the Check Point Institute for Information Security, BSF, ISF, and Marie Curie grants.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 426–445, 2011.

direction is to use cryptographic techniques to solve traditional Game Theoretic problems. In particular, the works of Dodis et al. [7], Ismalkov et al. [25,24], Abraham et al. [1] and Halpern and Pass [23] take this path and demonstrate how a mutli-party protocol using cryptographic techniques can be used to replace a trusted correlation device or a mediator in mechanism design.

Another line of research is to extend the traditional Game Theoretic formalisms to capture, within the context of Game Theory, cryptographic concerns and ideas that take into account the fact that protocol participants are computationally bounded, and that computational resources are costly [7,23,21].

Yet another line of work is aimed at using Game Theoretic concepts and approach to amend traditional cryptographic goals such as secure and fair computation. A focal point in this direction has been the concept of rational fair exchange of secrets (also known as *rational secret sharing*) [22,19,29,27,28,30,9,3]. Here the goal is to design a protocol for exchanging secrets in a way that "rational players" will be "interested" in following the protocol, where it is assumed that players are interested in learning the secret inputs of the other players while preventing others from learning their own secrets. In fact, it is assumed that the participants have specific preferences and some quantitative prior knowledge on these preferences of the participants is known to the protocol designer. Furthermore, such prior knowledge turns out to be essential in order to get around basic impossibility results [3,6].

These ingenious works demonstrate the benefit in having a joint theory of protocols for collaborative but competing parties; but at the same time they underline the basic incompatibility in the two formalisms. For instance, the (primarily Game Theoretic) formalisms used in the works on rational secret sharing do not seem to naturally capture basic cryptographic concepts, such as semantic security of the secrets. Instead, these works opt for more simplistic notions that are not always compatible with traditional cryptographic formalisms. In particular, existing modeling (that is used both by constructions and by impossibility results) treats the secret as an atomic unit and consider only the case where the parties either learnt or did not learn the secret *entirely.* Unlike traditional cryptographic modeling, the option where *partial information* about the secret is leaked through the execution is disregarded.

This work. We relate the two *formalisms.* In particular we show how Game Theoretic formalism and concepts can be used to capture traditional cryptographic security properties of protocols. We concentrate on the setting of two-party protocols and fail-stop adversaries. While this setting is admittedly limited, it does incorporate the core aspects of secrecy, correctness and fairness in face of malicious (i.e., not necessarily "rational") aborts.

In this setting, we first show Game Theoretic notions of secrecy and correctness that are *equivalent,* respectively, to the standard cryptographic notions of secret and correct evaluation of deterministic functions in the fail-stop setting (see e.g [12]). We then turn to capturing fairness. Here the situation turns out to be more intricate. We formulate a natural Game Theoretic notion of fairness, and observe that it is *strictly weaker* than existing cryptographic notions of fair two-party function evaluation. We then formulate new cryptographic

notions of fairness that are *equivalent* to this Game Theoretic notion, and a simulation-based notion of fairness that implies the above three. Furthermore, we show that these new notions can indeed be realized in some potentially meaningful settings where traditional cryptographic notions are provably unrealizable.

The results in more detail. The basic idea proceeds as follows. We translate a given protocol into a set of games, in such a way that the protocol satisfies the cryptographic property in question *if and only if* a certain pair of strategies (derived from the protocol) are in a (computational) Nash equilibrium in each one of the games. This allows the cryptographic question to be posed (and answered) in Game Theoretic language. More precisely, given a protocol, we consider the (extensive form with incomplete information) game where in each step the relevant party can decide to either continue running the protocol as prescribed, or alteratively abort the execution. We then ask whether the pair of strategies that instruct the players to continue the protocol to completion is in a (computational) Nash equilibrium. Each cryptographic property is then captured by an appropriate set of utilities and input distributions (namely, distributions over the types). In particular:

Secrecy. A given protocol is secret (as in, e.g. [12]) if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the following set of utilities and distributions over the types. For each pair of values in the domain, we define a distribution that chooses an input for one party at random from the pair. The party gets low payoff if the two values lead to the same output value and yet the other party managed to guess which of the two inputs was used. It is stressed that this is the first time where a traditional cryptographic notion of secrecy (in the style of [16]) is captured in Game Theoretic terms. In particular, the works on rational secret sharing do not provide this level of secrecy for the secret. (Indeed, the solution approaches taken there need the secret to be taken from a large domain.)

Correctness. A protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities where the parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output.

Fairness. Here we make the bulk of our contributions. We first recall the basic setting: Two parties interact by exchanging messages in order to valuate a function f on their inputs. The only allowed deviation from the protocol is abortion, in which event both parties learn that the protocol was aborted. Consequently, a protocol in this model should specify, in addition to the next message to be sent, also a prediction of the output value in case the execution is aborted. (Although the setting makes sense for any function, it may be helpful to keep in mind the *fair exchange* function, where the output of each party is the input of the other.)

Current notions of fairness for two party protocols in this model (e.g., [20,18]) require there to be a point in the computation where both parties move from a state of no knowledge of the output to a full knowledge of it. This is a strong notion, which is impossible to realize in many situations. Instead, we would like

to investigate more relaxed notions of fairness, which allow parties to gradually learn partial information on their desired outputs - but do so in a way that is "fair". Indeed, such an approach seems reasonable both from a game theoretic point of view (as a zero-sum game) and from a cryptographic point of view via the paradigm of gradual release (see e.g. [4,15,11,20,3] and the references within).

A first thing to note about such a notion of fairness is that it is sensitive to the potential prior knowledge that the parties may have on each other's inputs. Indeed, a "gradual release" protocol that is "fair" without prior knowledge may become "unfair" in a situation where one of the parties has far more knowledge about the possible values of the inputs of the second party than vice versa.

We thus explicitly model in our security notions the knowledge that each party has on the input of the other party. That is, we let each party has, in addition to its own input, some additional information on the input of the other party. Furthermore, to simplify matters and put the two parties on equal footing, we assume that the information that the parties have on the input of the other consists of two possible values for that input. That is, each party receives three values: its own input, and two possible values for the input of the other party. Indeed, such information naturally captures situations where the domain of possible inputs is small (say, binary). The formalism can also be naturally extended to deal with domains of small size which is larger than two.

We first sketch our Game Theoretic notion. We consider the following set of distributions over inputs (types): Say that a quadruple of elements (a_0, a_1, b_0, b_1) in the domain of function f is *valid* if for all $i \in \{0, 1\}$, $f(a_0, b_i) \neq f(a_1, b_i)$ and $f(a_i, b_0) \neq f(a_i, b_1)$. For each valid quadruple of values in the domain, we define a distribution that chooses an input for one party at random from the first two values, and an input for other party at random from the other two values. The utility function for a party is the following: When the party aborts the protocol, each party predicts its output. If the party predicts correctly and the other one does not, then it gets payoff +1. If it predicts incorrectly and the other party predicts correctly then it gets payoff -1. Else, it gets payoff 0. We say that a protocol is Game Theoretically Fair if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the above utility, applied to both parties, and any distribution from the above family.

We then consider three different cryptographic notions of fairness and investigate their relationships with the above Game Theoretic notion.

- First, we formulate a simple "game based" notion of fairness that limits the gain of an arbitrary (i.e., not necessarily "rational") fail-stop adversary in a game that closely mimics the above Game Theoretic interaction. The main difference between the notions is that in the cryptographic setting the adversary is arbitrary, rather than rational. Still, we show that the notions are *equivalent*.
- Next, we show that this notion in fact corresponds to the natural concept of *gradual release*. That is, say that a protocol satisfies the *gradual release* property if at any round the probability of any party to predict its output increases only by a negligible amount. We show that a protocol is fair (as in

the above notions) *if and only if* it satisfies the gradual release property. We note that the notion of gradual release is in essence the basis of the classic protocols of Beaver, Goldwasser and Levin [4,15]. It has also been a key tool in the work of Asharov and Lindell [3]. Due to lack of space we do not present the definition of gradual release here; see full version [2] for a formal description.

– Then, we formulate an ideal-model based notion of fairness that allows for gradual release of secrets. In this notion the ideal functionality accepts a "sampling algorithm" M from the ideal-model adversary. The functionality then obtains the inputs from the parties and runs M on these inputs, and obtains from M the outputs that should be given to the two parties. The functionality then makes the respective outputs available to the two parties. (Ie, once the outputs are available, the parties can access them at any time.) The correctness and fairness guarantees of this interaction clearly depend on the properties of M. We thus require that M be both "fair" and "correct", in the sense that both parties get correct output with roughly equal (and substantial) probability. We then show that the new simulation based definition implies the gradual release notion. (We note that the converse does not necessarily hold with respect to secure computation in the fail-stop model, even disregarding fairness).

A positive result. Finally, we consider the realizability of this notion. Here, we first assert that the impossibility results of Cleve and Asharov and Lindell [6,3] hold even with respect to the new notions, as long as *both parties are required to receive an output.* We then observe that our notion is meaningful even in the case where the parties are not guaranteed to always learn the correct output when played honestly. Surprisingly, in cases where the parties learn the correct output with probability one half or smaller (i.e., correctness holds with probability between 0 and 1/2), our simulation-based notion of fairness is in fact *achievable* with no set-up or trusted third parties. We demonstrate a family of two-party protocols, parameterized by this correctness probability, that realize the new notion of fairness. For instance, for the case that correctness is guaranteed with probability one half, we design a fair protocol where with probability one half both parties obtain the correct output, and with probability one half both parties obtain an incorrect value. An alternative protocol makes sure that each party obtains a correct output with probability one half, and at each execution exactly one party obtains the correct output value. These scenarios were not known to be achievable before (not even by [18]), and may prove to be useful.

On the definitional choices. One question that comes to mind when considering our modeling is why use plain Nash equilibria to exhibit correspondence between cryptographic notions and Game Theoretic ones. Why not use, for instance, stronger notions such as Dominant Strategy, Survival Under Iterated Deletions, or Subgame Perfect equilibria. It turns out that in our setting of two party computation with fail-stop faults, Nash equilibria do seem to be the concept that most naturally corresponds to cryptographic secure protocols. In particular, in

the fail-stop case any Nash equilibrium is sub-game perfect, or in other words empty threats do not hold (see more discussion on this point in the next section).

Future work. One interesting challenge is extending the results of this work to the Byzantine case. For one, in the Byzantine case there are multiple cryptographic notions of security, including various variants of simulation based notions. Capturing these notions using Game Theoretic tools might shed light on the differences between these cryptographic notions. In particular, it seems that here the Game Theoretic formalism will have to be extended to capture arbitrary polynomial time strategies at each decision point.In particular, it seems likely that more sophisticated Game Theoretic solution concepts such as sub-game perfect equilibria and computational relaxations thereof [21,31] will be needed.

Another challenge is to extend the notions of fairness presented here to address also situations where the parties have more general, asymmetric a priori knowledge on each other's inputs, and to find solutions that use minimal trust assumptions on the system. Dealing with the multi-party case is another interesting challenge.

Organization. Section 2 presents cryptographic and Game Theoretic "solution concepts". Section 4 presents results regarding fairness: (i) the game theoretic notion, (ii) the equivalent cryptographic definition, (iii) a new simulation based definition and, (iv) the study of the fairness definition. The gradual release property, its relation to fairness and some more details are found in the full version [2].

2 The Model and Solution Concepts

We review some basic definitions that capture the way we model protocols (or, equivalently, strategies), as well as the solution concepts we will consider — both the cryptographic and the game theoretic ones. While most of these definitions are known, some are new to this work.

2.1 Cryptographic Definitions

We review some standard cryptographic definitions of security for protocols.

The Fail-Stop setting. The setting that we consider in this paper is that of two-party interaction in the presence of *fail-stop* faults. In this setting both parties follow the protocol specification exactly, with the exception that any one of the parties may, at any time during the computation, decide to stop, or *abort* the computation. Specifically, it means that fail-stop adversaries do not change their initial input for the execution, yet, they may arbitrarily decide on their output.

Cryptographic Security. We present game based definitions that capture the notions of privacy and correctness. We restrict attention to deterministic functions. By definition [12], the view of the ith party ($i \in \{0,1\}$) during an execution of π on (x_0, x_1) is denoted $\mathsf{view}_{\pi,i}(x_0, x_1, n)$ and equals $(x_i, r^i, m_1^i, ..., m_t^i)$, where r^i equals the contents of the ith party's *internal* random tape, and m_j^i represents the jth message that it received.

Definition 1 (Privacy). *Let f and π be as above. We say that π* privately computes *f if the following holds:*

1. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party P_0*

$$\left\{\mathsf{view}_{\pi,\mathcal{A}(z),0}(x_0,x_1,n)\right\}_{x_0,x_1,x_1',y,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathsf{view}_{\pi,\mathcal{A}(z),0}(x_0,x_1',n)\right\}_{x_0,x_1,x_1',z\in\{0,1\}^*,n\in\mathbb{N}}$$

   where $|x_0|=|x_1|=|x_1'|$ and $f(x_0,x_1)=f(x_0,x_1')$.

2. *For every non-uniform* PPT *adversary $\mathcal{A}$ that controls party P_1*

$$\left\{\mathsf{view}_{\pi,\mathcal{A}(z),1}(x_0,x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \left\{\mathsf{view}_{\pi,\mathcal{A}(z),1}(x_0',x_1,n)\right\}_{x_0,x_0',x_1,z\in\{0,1\}^*,n\in\mathbb{N}}$$

   where $|x_0|=|x_0'|=|x_1|$ and $f(x_0,x_1)=f(x_0',x_1)$.

Definition 2 (Correctness). *Let f and π be as above. We say that π* correctly computes *f if for all sufficiently large inputs x_0 and x_1 such that $|x_0|=|x_1|=n$, we have that $\Pr[\mathsf{output}_{\pi,i}\in\{\perp\circ\{0,1\}^*, f(x_0,x_1)\}]\geq 1-\mu(n)$, where $\mathsf{output}_{\pi,i}\neq\perp$ denotes the output returned by P_i upon the completion of π whenever the strategy of the parties is* `continue`*, and μ is a negligible function.*

2.2 Game Theoretic Definitions

We review the relevant concepts from Game Theory, and the extensions needed to put these concepts on equal footing as the cryptographic concepts. Traditionally, a 2-player *(normal form, full information)* game $\Gamma=(\{A_0,A_1\},\{u_0,u_1\})$ is determined by specifying, for each player P_i, a set A_i of possible actions and a utility function $u_i: A_0\times A_1\mapsto R$. Letting $A\stackrel{\text{def}}{=}A_0\times A_1$, we refer to a tuple of actions $a=(a_0,a_1)\in A$ as an outcome. The utility function u_i of party P_i expresses this player's preferences over outcomes: P_i prefers outcome a to outcome a' if and only if $u_i(a)>u_i(a')$. A *strategy* σ_i for P_i is a distribution on actions in A_i. Given a strategy vector $\sigma=\sigma_0,\sigma_1$, we let $u_i(\sigma)$ be the expected utility of P_i given that all the parties play according to σ. We continue with a definition of Nash equilibria:

Definition 3 (Nash equilibria for normal form, complete information games). *Let $\Gamma=(\{A_0,A_1\},\{u_0,u_1\})$ be as above, and let $\sigma=\sigma_0,\sigma_1$ be a pair of strategies as above. Then σ is in a* Nash equilibrium *if for all i and any strategy σ_i' it holds that $u_i(\sigma_0'',\sigma_1'')\leq u_i(\sigma)$, where $\sigma_i''=\sigma_i'$ and $\sigma_{1-i}''=\sigma_{1-i}$.*

The above formalism is also naturally extended to the case of *extensive form* games, where the parties take turns when taking actions. Another natural extension is to games with *incomplete information.* Here each player has an additional piece of information, called *type,* that is known only to itself. That is, the strategy σ_i now takes as input an additional value x_i. To extend the notion of Nash equilibria to deal with this case, it is assumed that an a priori distribution on the inputs (types) is known and fixed.

Definition 4 (Nash equilibria for extensive form, incomplete information games). *Let $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ be as above, and let D be a distribution over $(\{0,1\}^*)^2$. Also, let $\sigma = \sigma_0, \sigma_1$ be a pair of extensive-form strategies as described above. Then σ is in a* Nash equilibrium for D *if for all i and any strategy σ_i' it holds that $u_i(x_0, x_1, \sigma_0''(x_0), \sigma_1''(x_1)) \leq u_i(x_0, x_1, \sigma_0(x_0), \sigma_1(x_1))$, where (x_0, x_1) is taken from distribution D, $\sigma_i(x)$ denotes the strategy of P_i with type x, $\sigma_i'' = \sigma_i'$ and $\sigma_{1-i}'' = \sigma_{1-i}$.*

Extensions for the cryptographic model. We review the (by now standard) extensions of the above notions to the case of computationally bounded players. See e.g. [7,26] for more details. The first step is to model a strategy as an (interactive) probabilistic Turing machine that algorithmically generates the next move given the type and a sequence of moves so far. Next, in order to capture computationally bounded behavior (both by the acting party and, more importantly, by the other party), we move to an asymptotic treatment. That is, we consider an infinite sequence of games. The third and last step is to relax the notion of "greater or equal to" to "not significantly less than". This is intended to compensate for the small inevitable imperfections of cryptographic constructs. That is, we have:

Definition 5 (Computational Nash equilibria for extensive form, incomplete inf. games). *Let $\Gamma = (\{A_0, A_1\}, \{u_0, u_1\})$ be as above, and let $D = \{D_n\}_{n \in \mathbf{N}}$ be a family of distributions over $(\{0,1\}^*)^2$. Let $\sigma = \sigma_0, \sigma_1$ be a pair of* PPT *extensive-form strategies as described above. Then σ is in a* Nash equilibrium for D *if for all sufficiently large n's, all i and any* PPT *strategy σ_i' it holds that $u_i(n, x_0, x_1, \sigma_0''(n, x_0), \sigma_1''(n, x_1)) \leq u_i(n, x_0, x_1, \sigma_0(n, x_0), \sigma_1(n, x_1)) + \mu(n)$, where (x_0, x_1) is taken from distribution D_n, $\sigma_i(x, n)$ denotes the strategy of P_i with type x, $\sigma_i'' = \sigma_i'$ and $\sigma_{1-i}'' = \sigma_{1-1}$, and μ is a negligible function.*

Our setting. We consider the following setting: At each step, the relevant party can make a binary decision: Either *abort* the computation, in which case the other party is notified that an abort action has been taken, or else *continue running the protocol π scrupulously.* The traditional Game Theoretic modeling of games involving such "exogenous" random choices that are not controllable by the players involves, introduces additional players (e.g., "Nature") to the game. In our case, however, the situation is somewhat different, since the random choices may be secret, and in addition each player also has access to local state that is preserved throughout the interaction and may affect the choices. Specifically, an action may specify a (potentially randomized) algorithm and a configuration. The outcome of taking this action is that an output of running the said algorithm from the said configuration, is appended to the history of the execution, and the new configuration of the algorithm is added to the local history of the player. More formally:

Definition 6. *Let $\pi = (P_0, P_1)$ be a two-party protocol (i.e., a pair of Interactive Turing Machines). Then, the* local history *of P_i (for $i \in \{0,1\}$), during*

an execution of π *on input* (x_0, x_1) *and* internal *random tape* r^i*, is denoted by* $\mathsf{history}_{\pi,i}(x_0, x_1, n)$ *and equals* $(x_i, r^i, m_1^i, \ldots, m_t^i)$*, where* m_j^i *represents its* j*th message. The* history *of* π *during this execution is captured by* $(m_1^0, m_1^1), ..., (m_t^0, m_t^1)$ *and is denoted by* $\mathsf{history}_\pi$*. The configuration of* π *at some point during the interaction consists of the local configurations of* P_0, P_1*.*

Fail-stop games. We consider games of the form $\Gamma_{\pi,u} = (\{A_0, A_1\}, \{u_0, u_1\})$, where $A_0 = A_1 = \{\texttt{continue,abort}\}$. The decision is taken before the sending of each message. That is, first the program π_i is run from its current configuration, generating an outgoing message. Next, the party makes a strategic decision whether to continue or to abort. A `continue` action by player i means that the outgoing message generated by π_i is added to the history, and the new configuration is added to the local history. An `abort` action means that a special `abort` symbol is added to the configurations of both parties and then both π_0 and π_1 are run to completion, generating local outputs, and the game ends. We call such games *fail-stop games.*

The utility functions in fail-stop games may depend on all the histories: the joint one, as well as the local histories of both players. In the following sections, it will be convenient to define utility functions that consider a special field of the local history, called the *local output* of a player P_i. We denote this field by $\mathsf{output}_{\pi,i}$. Denote by $\sigma^{\texttt{continue}}$ the strategy that always returns `continue`. The basic Game Theoretic property of protocols that we will be investigating is whether the pair of strategies $(\sigma^{\texttt{continue}}, \sigma^{\texttt{continue}})$ is in a (computational) Nash equilibrium in fail-stop games, with respect to a given set of utilities and input distributions. That is:

Definition 7 (Nash protocols). *Let* $\mathcal{D}$ *be a set of distribution ensembles over pairs of strings, and let* $\mathcal{U}$ *be a set of extensive-form binary utility functions. A two-party protocol* π *is called* Nash Protocol with respect to $\mathcal{U}, \mathcal{D}$ *if, for any* $u \in \mathcal{U}$ *and* $D \in \mathcal{D}$*, the pair of strategies* $\sigma = (\sigma^{\texttt{continue}}, \sigma^{\texttt{continue}})$ *is in a computational Nash equilibrium for the fail-stop game* $\Gamma_{\pi,u}$ *and distribution ensemble* D*.*

On subgame perfect equilibria and related solution concepts. An attractive solution concept for extensive form games (namely, interactive protocols) is subgame perfect equilibria, which allow for analytical treatment which is not encumbered by "empty threats". Furthermore, some variants of this notion that are better suited to our computational setting have been recently proposed (see [21,31]). However, we note that in our limited case of fail-stop games any Nash equilibrium is subgame perfect. Indeed, once one of the parties aborts the computation there is no chance for the other party to "retaliate", hence empty threats are meaningless. (Recall that the output generation algorithms are not strategic, only the decision whether to abort is.)

3 Privacy and Correctness in Game Theoretic View

In this section we capture the traditional cryptographic privacy and correctness properties of protocols using Game Theoretic notions. We restrict attention to

the fail-stop setting and deterministic functions with a single output. (Fairness aside, private computation of functions with two distinct outputs can be reduced to this simpler case; see [12] for more details.)

Privacy in Game Theoretic view. Our starting point is the notion of private computation. A protocol is private if no (fail-stop) PPT adversary is able to distinguish any two executions where the adversary's inputs and outputs are the same, even when the honest party uses different inputs in the two executions. Our goal, then, is to define a set of utility functions that preserve this property for Nash protocols. We therefore restrict ourselves to input distributions over triples of inputs, where the input given to one of the parties is fixed, whereas the input of the other party is uniformly chosen from the remaining pair. This restriction captures the strength of cryptographic (semantic) security: *even* if a party knows that the input of the other party can only be one out of two possible values, the game does not give it the ability to tell which is the case. We then have a distribution for each such triple.

We turn to defining the utility functions. At first glance it may seem that one should define privacy by having each party *gain* whenever it learns something meaningful on the other party's private input. Nevertheless, it seems that it is better to make a party *lose* if the other party learns anything about its secret information. Intuitively, the reason is that it must be worthwhile for the party who holds the data to maintain it a secret. In other words, having the other party gain any profit when breaking secrecy is irrelevant, since it does not introduce any incentive for the former party to prevent this leakage. (Note however that here the utility of a party depends on events that are not visible to it during the execution.) The following definition formalizes the above.

Definition 8 (Distribution ensemble for privacy). *The* distribution ensemble for privacy for P_0 *for a two-party function f is the ensemble $\mathcal{D}^{\mathsf{p}}_f = \{D^{\mathsf{p}}_{f,n}\}_{n\in\mathbf{N}}$ where $D^{\mathsf{p}}_{f,n} = \{D_{a_0,a_1,b}\}_{a_0,a_1,b\in\{0,1\}^n, f(a_0,b)=f(a_1,b)}$, and $D_{a_0,a_1,b}$ outputs (x,b), where $x \overset{R}{\leftarrow} (a_0,a_1)$.*

Distribution ensembles for privacy for P_1 are defined analogously.

Let π be a two-party protocol computing a function f. Then, for every n, a, b, c as above and for every PPT algorithm $\mathcal{B}$, let the *augmented protocol for privacy* for π, with guess algorithm $\mathcal{B}$, be the protocol that first runs π, and then runs $\mathcal{B}$ on the local state of π and two additional auxiliary values. We assume that $\mathcal{B}$ outputs a binary value. This value is interpreted as a guess for which of the two auxiliary values is the input value of the other party.

Definition 9 (Utility function for privacy). *Let π be a two-party protocol and f be a two party function. Then, for every a_0, a_1, b such that $f(a_0,b) = f(a_1,b)$, and for every guessing algorithm $\mathcal{B}$, the* utility function for privacy *for party P_0, on input $x \in \{a_0,a_1\}$, is defined by:*

$$u^{\mathsf{p}}_0(\mathsf{history}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1}(x,b,n), a_0, a_1, b) \mapsto \begin{cases} -1 \text{ if } \mathsf{guess}_{\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}},1} = g \text{ and } x = a_g \\ 0 \text{ otherwise} \end{cases}$$

The utility function for party P_1 is defined analogously. Note that if the history of the execution is empty, ie, no message has been exchanged between the parties, and the inputs of the parties are taken from a distribution ensemble for privacy, then u_0^{p} equals at least $-1/2$. This is due to the fact that P_1 can only guess x with probability at most $1/2$. Therefore, intuitively, it will be rational for P_0 to participate in the protocol (rather than to abort at the beginning) only if (and only if) the other party cannot guess the input of P_0 with probability significantly greater than $1/2$. The definition of Game-Theoretic privately is as follows:

Definition 10 (Game-Theoretic private protocols). *Let f and π be as above. Then, we say that π is* Game-Theoretic private for party P_0 *if $\pi^{\mathsf{p}}_{\mathsf{Aug},\mathcal{B}}$ is a Nash protocol with respect to $u_0^{\mathsf{p}}, u_1^{\mathsf{p}}$ and $\mathcal{D}_f^{\mathsf{p}}$ and all valid* PPT *$\mathcal{B}$.*

Game-Theoretic private protocol for P_1 is defined analogously. A protocol is Game-Theoretic private if it is Game-Theoretic private both for P_0 and for P_1.

Theorem 11. *Let f be a deterministic two-party function, and let π be a two-party protocol that computes f correctly (cf. Definition 2). Then, π is Game-Theoretic private if and only if π privately computes f in the presence of fail-stop adversaries.*

The proof can be found in the full version [2].

Correctness in Game Theoretic view. We continue with a formulation of a utility function that captures the notion of correctness as formalized in Definition 12. That is, we show that a protocol correctly computes a deterministic function if and only if the strategy that never aborts the protocol is in a computational Nash equilibrium with respect to the set of utilities specified as follows. The parties get high payoff only if they output the correct function value on the given inputs (types), or abort before the protocol starts; in addition, the players get no payoff for incorrect output. More formally, we introduce the set of distributions for which we will prove the Nash theorem. The distribution ensemble for correctness is simply the collection of all point distributions on pairs of inputs:

Definition 12 (Distribution ensemble for correctness). *Let f be a deterministic two-party function. Then, the* distribution ensemble for correctness *is the ensemble $\mathcal{D}_f^{\mathsf{c}} = \{D_n^{\mathsf{c}}\}_{n\in\mathbf{N}}$ where $D_n^{\mathsf{c}} = \{D_{a,b}\}_{a,b\in\{0,1\}^n}$, and $D_{a,b}$ outputs (a,b) w.p. 1.*

Note that a fail-stop adversary cannot affect the correctness of the protocol as it plays honestly with the exception that it may abort. Then, upon receiving an abort message we have the following: (i) either the honest party already learnt its output and so, correctness should be guaranteed, or, (ii) the honest party did not learn the output yet, for which it outputs $\perp$ together with its guess for the output (which corresponds to a legal output by Definition 2). Note that this guess is different than the guess appended in Definition 9 of utility definition for privacy, as here, we assume that the protocol instructs the honest party how to

behave in case of an abort. Furthermore, an incorrect protocol in the presence of fail-stop adversary implies that the protocol is incorrect regardless of the parties' actions (where the actions are continue or abort).

This suggests the following natural way of modeling a utility function for correctness: The parties gain a higher utility if they output the correct output, and lose if they output an incorrect output. Therefore, the `continue` strategy would not induce a Nash Equilibrium in case of an incorrect protocol, as the parties gain a higher utility by not participating in the execution. More formally:

Definition 13 (Utility function for correctness). *Let π be a two-party fail-stop game as above. Then, for every a, b as above the* utility function for correctness *for party P_0, denoted u_0^{c}, is defined by:*

- $u_0^{\mathsf{c}}(\mathsf{history}^{\phi}_{\pi,0}) = 1$.
- $u_0^{\mathsf{c}}(\mathsf{output}_{\pi,0}, a, b) \mapsto \begin{cases} 1 \text{ if } \mathsf{output}_{\pi,0} = f(a,b) \\ 0 \text{ otherwise} \end{cases}$

where $\mathsf{history}^{\phi}_{\pi,0}$ denotes the case that the local history of P_0 is empty. (Namely, P_0 does not participate in the protocol).

Intuitively, this implies that the protocol is a fail-stop Game if it is correct and vice versa. A formal statement follows below. u_1^{c} is defined analogously, with respect to P_1.

Theorem 14. *Let f be a deterministic two-party function, and let π a two-party protocol. Then, π is a Nash protocol with respect to $u_0^{\mathsf{c}}, u_1^{\mathsf{c}}$ and $\mathcal{D}_f^{\mathsf{c}}$ if and only if π correctly computes f in the presence of fail-stop adversaries.*

The proof can be found in the full version [2].

4 Exploring Fairness in the Two-Party Setting

Having established the notions of privacy and correctness using Game Theoretic formalism, our next goal is to capture fairness in this view. However, this turns out to be tricky, mainly due to the highly "reciprocal" and thus delicate nature of this notion. To illustrate, consider the simplistic definition for fairness that requires that one party learns its output if and only if the second party does. However, as natural as it seems, this definition is lacking since it captures each party's output as an atomic unit. As a result, it only considers the cases where the parties either learnt or did not learn their output *entirely*, and disregards the option in which partial information about the output may be gathered through the execution. So, instead, we would like to have a definition that calls a protocol fair if at any point in the execution both parties gather, essentially, the same partial information about their respective outputs.

Motivated by this discussion, we turn to the Game Theoretic setting with the aim to design a meaningful definition for fairness, as we did for privacy and correctness. This would, for instance, allow investigating known impossibility results under a new light. Our starting point is a definition that examines the

information the parties gain about their outputs during the game, where each party loses nominatively to the success probability of the other party guessing its output. (This is motivated by the same reasoning as in privacy). In order to obtain this, we first define a new set of utility functions for fairness for which we require that the game would be Nash; see Section 4.1 for the complete details.

Having defined fairness for rational parties, we wish to examine its strength against cryptographic attacks. We therefore introduce a new game-based definition that formalizes fairness for two-party protocols and is, in fact, equivalent to the Game Theoretic definition; see Theorem 21.

We then introduce in Section 4.3 a new notion of simulation based definition for capturing security of protocols that follow our game-based notion of fairness, specified above. This new notion is necessary as (gamed-based) fair protocols most likely cannot be simulatable according to the traditional simulation based definition [12]. We consider the notion of "partial information" in the ideal world alongside preserving some notion of privacy. We then prove that protocols that satisfy this new definition are also fair with respect to game-based definition.

Finally, we consider the realizability of our notion of fairness. We then observe that our notion is meaningful even in the case where parties are not guaranteed to always learn the output when both parties never abort. Somewhat surprisingly, in cases where the parties learn the output with probability one half or smaller, our notion of fairness is in fact *achievable* with no set-up or trusted third parties. We demonstrate two-party protocols that realize the new notion in this settings. We also show that whenever this probability raises above one half, our notion of fairness cannot be realized at all.

4.1 Fairness in Game Theoretic View

In this section we present our first definition for fairness that captures this notion from a Game Theoretic view. As for privacy and correctness, this involves definitions for utility functions, input distributions and a concrete fail-stop game (or the sequence of games). We begin with the description of the input distributions. As specified above, the input of each party is picked from a domain of size two, where all the outputs are made up of distinct outputs. More formally,

Definition 15 (Collection of distribution ensembles for fairness). *Let f be a two-party function. Let $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ be an input tuple such that: $|x_0^0| = |x_0^1| = |x_1^0| = |x_1^1| = n$, and for every $b \in \{0,1\}$ it holds that:*

- $f_0(x_0^0, x_1^b) \neq f_0(x_0^1, x_1^b)$ *(in each run there are two possible outputs for P_0).*
- $f_1(x_0^b, x_1^0) \neq f_1(x_0^b, x_1^1)$*, (in each run there are two possible outputs for P_1).*

Then, a collection of distribution ensembles for fairness $\mathcal{D}_f^{\mathsf{f}}$ *is a collection of distributions $\mathcal{D}_f^{\mathsf{f}} = \{D_{x_0^0,x_0^1,x_1^0,x_1^1}\}_{x_0^0,x_0^1,x_1^0,x_1^1,n}$ such that for every $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ as above, $D_{x_0^0,x_0^1,x_1^0,x_1^1,n}$ is defined by*

$$(x_0, x_1) \leftarrow D_{x_0^0,x_0^1,x_1^0,x_1^1,n}(1^n), \quad \text{where} \quad x_0 \overset{R}{\leftarrow} (x_0^0, x_0^1) \quad \text{and} \quad x_1 \overset{R}{\leftarrow} (x_1^0, x_1^1).$$

Next, let $\pi_{\mathcal{B}}$ be the protocol, where $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$. By this notation, we artificially separate between the protocol and the predicting algorithms in case of prematurely abort. More precisely, in the case that P_0 prematurely aborts, P_1 invokes algorithm $\mathcal{B}_1$ on its input, its auxiliary information and the history of the execution, and outputs whatever $\mathcal{B}_1$ does. $\mathcal{B}_0$ is defined in a similar manner. In fact, we can refer to these two algorithms by the instructions of the parties regarding the values they need to output after each round, capturing the event of an early abort. We stress these algorithms are embedded within the protocol. However, this presentation enables us to capture scenarios where one of the parties follow the guessing algorithm as specified by the protocol, whereas, the other party follows an arbitrary algorithm. That is, we can consider protocols $\pi_{\mathcal{B}'}$ (with $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$) that are equivalent to the original protocol $\pi_{\mathcal{B}}$ except for the fact that P_0 guesses its output according to $\widetilde{\mathcal{B}}_0$ instead of $\mathcal{B}_0$.

We describe the fairness game $\Gamma_{\pi_{\mathcal{B}},u^{\mathsf{f}}}$ for some $\mathcal{B} = (\mathcal{B}_0, \mathcal{B}_1)$. The inputs of the parties, x_0, x_1, are selected according to some distribution ensemble $D_{x_0^0,x_0^1,x_1^0,x_1^1}$ as defined in Definition 15. Then, the parties run the fail-stop game, where their strategies instruct them in each step whether to `abort` or `continue`. In case that a party P_i aborts, the outputs of both parties are determined by the algorithms $(\mathcal{B}_0, \mathcal{B}_1)$. Let $\mathsf{output}_{\pi_{\mathcal{B}'},i}$ denote the output of P_i in game $\pi_{\mathcal{B}'}$, then a utility function for fairness is defined by:

Definition 16 (Utility function for fairness). *Let f be a deterministic two-party function, and let π be a two-party protocol. Then, for every $x_0^0, x_0^1, x_1^0, x_1^1, n$ as above (cf. Definition 15), for every pair of strategies (σ_0, σ_1) and for every* PPT $\widetilde{\mathcal{B}}_0$*, the* utility function for fairness *for party P_0, denoted by u_0^{f}, is defined by:*

$$u_0^{\mathsf{f}}(\sigma_0,\sigma_1) \mapsto \begin{cases} 1 \text{ if } \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},0} = f_0(x_0,x_1) \wedge \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},1} \neq f_1(x_0,x_1) \\ -1 \text{ if } \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},0} \neq f_0(x_0,x_1) \wedge \mathsf{output}_{\pi^{\mathsf{f}}_{\mathcal{B}'},1} = f_1(x_0,x_1) \\ 0 \text{ otherwise} \end{cases}$$

where x_0, x_1 are as in Definition 15 and $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$. Moreover, the utility for P_1, $u_1^{\mathsf{f}} = 0$.

Since the utility function of P_1 is fixed, only P_1 has no incentive to change its strategy. Moreover, we consider here the sequence of games where P_1 always guesses its output according to $\mathcal{B}_1$, the "original" protocol. This actually means that P_1 always plays as honest, in the cryptographic point of view. We are now ready to define a protocol that is Game-Theoretic fair for P_1 as:

Definition 17 (Game-Theoretic fairness for P_1). *Let f and $\pi_{\mathcal{B}}$ be as above. Then, we say that $\pi_{\mathcal{B}}$ is* Game-Theoretic fair for party P_0 *if $\Gamma_{\pi_{\mathcal{B}'},(u_0^{\mathsf{f}},u_1^{\mathsf{f}})}$ is a Nash protocol with respect to $(u_0^{\mathsf{f}}, u_1^{\mathsf{f}})$ and $\mathcal{D}_f^{\mathsf{f}}$ and all* PPT $\widetilde{\mathcal{B}}_0$*, where $\mathcal{B}' = (\widetilde{\mathcal{B}}_0, \mathcal{B}_1)$.*

Similarly, we define Game-Theoretic fair for party P_0, where here we consider all the protocols $\pi_{\mathcal{B}'}$, for all PPT $\widetilde{\mathcal{B}}_1$ and $\mathcal{B}' = (\mathcal{B}_0, \widetilde{\mathcal{B}}_1)$, and the utilities functions are opposite (that is, the utility for P_0 is fixed into zero, whereas the utility of P_1 is modified according to its guess). We conclude with the definition for Game-Theoretic protocol:

Definition 18 (Game-Theoretic fair protocol). *Let f and π be as above. Them. we say that π is* Game-Theoretic fair protocol *if it is Game-Theoretic fair for both P_0 and P_1.*

4.2 A New Indistinguishability-Based Definition of Fairness

We now define a game-based definition (or, an indistinguishability definition) for fairness in cryptographic settings. Again, as in the game-theoretic settings, we assume that the protocol instructs the party what to output in case of abortion. Our definition tests the protocol in a "fail" environment, where each party has two possible inputs and its effective input is chosen uniformly at random from this set. Moreover, both parties know the input tuple and the distribution over the inputs. Before introducing the game-based definition, we first introduce non-trivial functionalities, to avoid functionalities that one of the parties may know the correct output without participating.

Definition 19 (Non-trivial functionalities.). *Let f be a two-party function. Then, f is non trivial if for all sufficiently large n's, there exists an input tuple $(x_0^0, x_0^1, x_1^0, x_1^1, n)$ such that $|x_0^0| = |x_0^1| = |x_1^0| = |x_1^1| = n$ and $\{f_0(x_0, x_1^b), f_0(x_0, x_1^b)\}_{b\in\{0,1\}}$, $\{f_1(x_0^b, x_1^0), f_1(x_0^b, x_1^1)\}_{b\in\{0,1\}}$ are distinct values.*
We are now ready to introduce our formal definition for fairness:

Definition 20 (Game-based definition for fairness.). *Let f be a non-trivial two-party function, and let π be a two-party protocol. Then, for every input tuple (cf. Definition 19) and any* PPT *fail-stop adversary $\mathcal{A}$, we define the following game:*
Game $\mathsf{Fair}_{\pi,\mathcal{A}}(x_0^0, x_0^1, x_1^0, x_1^1, n)$*:*

1. Two bits b_0, b_1 are picked at random.
2. Protocol π is run on inputs $x_0^{b_0}$ for P_0 and $x_1^{b_1}$ for P_1, where $\mathcal{A}$ sees the view of P_{i^*}.
3. Whenever $\mathcal{A}$ outputs a value y, P_{1-i^*} is given an `abort` message. (At this point, P_{1-i^*} would write its guess for $f_{1-i^*}(x_0^{b_0}, x_1^{b_1}, n)$ on its output tape.)
4. The output of the game is:
   - 1 if (i) $y = f_0(x_0^{b_0}, x_1^{b_1}, n)$ and (ii) P_{1-i^*} does not output $f_1(x_0^{b_0}, x_1^{b_1}, n)$.
   - -1 if (i) $y \neq f_0(x_0^{b_0}, x_1^{b_1}, n)$ and (ii) P_{1-i^*} outputs $f_1(x_0^{b_0}, x_1^{b_1}, n)$.
   - 0 in any other possibility (both parties output correct outputs, or both parties output incorrect outputs).

We say that π fairly computes *f if for every* PPT *adversary $\mathcal{A}$, there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large inputs it holds that,*

$$\mathrm{E}(\mathsf{Fair}_{\pi,\mathcal{A}}(x_0^0, x_0^1, x_1^0, x_1^1, n)) \leq \mu(n)$$

At first sight it may seem that Definition 20 is tailored for the fair exchange function, i.e., when the parties trade their inputs. This is due to the fact that the parties' output completely reveal their inputs. Nevertheless, we note that the definition does not put any restriction on f in this sense, and is aimed to capture fairness with respect any nontrivial function. We continue with the following theorem:

Theorem 21. *Let f be a two-party function and let π be a protocol that computes f correctly. Then, π is Game-Theoretic fair (in the sense of Definition 18), if and only if π fairly computes f in the presence of fail-stop adversaries, (in the sense of Definition 20).*
The proof for this Theorem can be found in the full version [2].

4.3 A New Notion of Simulation Based Fairness

We formulate a new simulation-based notion of fair two party computation. The goal is to capture in a simulation-based way the same concept captured by the previous notions in this section. That is, we wish to allow the parties to obtain "partial information" on each other's secrets, as long as the gain of information is "essentially the same" for both parties.

The basic idea is to consider an ideal functionality (namely, a trusted party) which, in addition to obtaining from the parties their own inputs and a priori information on the input of the other party, also obtains from the ideal adversary (i.e., the simulator), a sampling PPT machine M. The functionality then runs M on the inputs of the parties and sends the parties the outputs that M returns. In order for our definition to make sense in the fair setting, we require that M should be "fair" in the sense that the values obtained by the parties are correlated with their respective outputs in essentially the same way.

For lack of space, we only sketch the highlights of this definition. See full details in [2]. We make the following requirements from the machine M:

1. CORRECTNESS: We require that $y'_0 = f_0(x_0, x)$, $y'_1 = f_1(x', x_1)$ for some x, x', where x_0, x_1 are the inputs of the parties that were sent to the trusted party and y'_0, y_1 are M's outputs. Namely, the sampling machine can never output a value for some party that is uncorrelated with its input.
2. FAIRNESS: we require that there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large n's it holds that:

$$\left|\Pr\left[y_{i^*} = f_{i^*}(x_0, x_1)\right] - \frac{1}{2}\right| \leq \Pr\left[y_{1-i^*} = f_{1-i^*}(x_0, x_1)\right] - \frac{1}{2} + \mu(n) \quad (1)$$

where $(y_0, y_1) = M(x_0, x_1, z_0, z_1, r)$, the adversary controls party i^* and the probability is taken over the random coins of M.

The definition of security is now the standard one:

Definition 22. *Protocol π is said to* securely compute *f if for every* PPT *adversary $\mathcal{A}$ in the real model, there exists a* PPT *simulator $\mathcal{S}$ in the ideal model, such that:*

$$\{\text{NIDEAL}_{f,\mathcal{S}}(\boldsymbol{x})\}_{\boldsymbol{x}\in\{0,1\}^*} \equiv \{\text{REAL}_{\pi,\mathcal{A}}(\boldsymbol{x})\}_{\boldsymbol{x}\in\{0,1\}^*}$$

where the ideal execution uses any (adversatively chosen) sampling machine M that satisfies the above requirements.

Theorem 23. *Let f, π be as above. Then, if π is simulatable (in the sense of Definition 22), then π is fair with respect to the Game-Based (in the sense of Definition 20).*

4.4 The Feasibility of Our Definition

In this section, we study our new game-based cryptographic definition of fairness in a cryptographic context. Our starting point is any correct protocol, where both parties learn their output if playing honestly. We then show, that by relaxing the (negligibly close to) perfect completeness requirement, which implies that the parties should (almost) always learn their output if playing honestly, we can fully characterize the set of two-party protocols according partial correctness. Informally,

1. In case correctness holds with probability that is non-negligibly greater than 1/2, we present an impossibility result, saying that there does not exists a fair protocol with this probability of correctness. This implies that the difficulties in designing fair protocols are already embedded within the fail-stop setting. Stating differently, these difficulties already emerge whenever early abort is permitted.
2. On the positive side, in case correctness holds with probability that is smaller equals to 1/2, we show how to design a fair protocol that meets our notion of fairness. Specifically, we present a family of such protocols, parameterized by this probability of correctness. The implications of this is that there may be still hope for the fail-stop setting with respect to designing fair protocols.

An impossibility result. In this section we demonstrate that our game-based definition for fairness cannot be achieved for protocols that guarantee correctness with probability greater than 1/2. Before turning to our main theorem we present a definition of an α-correct protocol.

Definition 24. *Let f be a non-trivial two-party function, and let π be a two-party protocol. We say that the protocol π is a α-*correct for *f if there exists a negligible function $\mu(\cdot)$ such that for all sufficiently large x_0, x_1, n such that $|x_0| = |x_1| = n$,*

$$\left|\Pr[\mathsf{output}_{\pi,0}(x_0, x_1) = f_0(x_0, x_1) \wedge \mathsf{output}_{\pi,1}(x_0, x_1) = f_1(x_0, x_1, n)] - \alpha\right| \leq \mu(n)$$

where $\mathsf{output}_{\pi,i}(x_0, x_1)$ denote the output of party P_i when invoked on input x_i, while P_{1-i} is invoked on x_{1-i}, and both parties are honest.

Our theorem of impossibility:

Theorem 25. *Let f be a non-trivial two-party function. Then, for every non-negligible function $\epsilon(\cdot)$ and every $\alpha > 1/2 + \epsilon(n)$, there does not exist an α-*correct *protocol which is also fair (in the sense of Definition 20), with a polynomial round complexity.*

A positive result. Interestingly, we show that for relaxed correctness (i.e., lower equal than 1/2), there *do* exist non-trivial functionalities that can be computed fairly in this setting. In the following, we present a fair protocol in which either both parties learn the correct output together, or alternatively neither

party obtains a correct result. The case where in each execution exactly one party learns its correct output can also be achieved with fairness. More generally, denote by α the probability in which both parties should learn their outputs. Then, we show that for every $\alpha \leq 1/2$, there exists an α-correct protocol that is also fair, even in the non-simultaneous channel model. This relaxation is necessary to obtain fairness, as higher α values set a threshold for achieving this property (as shown in Section 4.4). Intuitively, the fact that each party does not know whether it has the correct output implies that a corrupted party would not have any incentive to abort after learning its output, since it does not give the honest party any new information anyway.

The protocol. The protocol is invoked over tuples of inputs with the distribution of choosing each input randomly out of a known pair. Let $x_0^0, x_0^1, x_1^0, x_1^1$ denote such an input tuple and denote by $x_0^{\mathsf{true}} \stackrel{\text{def}}{=} f_0(x_0^{b_0}, x_1^{b_1}), x_0^{\mathsf{false}} \stackrel{\text{def}}{=} f_0(x_0^{b_0}, x_1^{1-b_1})$, $x_1^{\mathsf{true}} \stackrel{\text{def}}{=} f_1(x_0^{b_0}, x_1^{b_1})$, and $x_1^{\mathsf{false}} \stackrel{\text{def}}{=} f_1(x_0^{1-b_0}, x_1^{b_1})$.

Then, function f^α; formally defined below, sets the output of the parties such that both learn the correct output with probability α, as required from an α-correct protocol. Moreover, the parties realize function f^α via protocol $\pi^\alpha_{\mathsf{abort}}$, which is secure-with-abort.

The Ideal Functionality f^α

- **Input:** P_0 inserts $b_0, x_0^0, x_0^1, x_1^0, x_1^1$. P_1 inserts $b_1, x_0^0, x_0^1, x_1^0, x_1^1$.
- **The function:**
  - Toss a coin σ that equals 0 with probability 2α, and equals 1 with probability $1 - 2\alpha$.
  - If $\sigma = 0$ (*parties learn same output*) do:
    - Toss another coin τ_0 uniformly at random from $\{0, 1\}$.
    - If $\tau_0 = 0$: set the output of P_0, P_1 to be $(x_0^{\mathsf{true}}, x_1^{\mathsf{true}})$, respectively.
    - If $\tau_0 = 1$: set the output of P_0, P_1 to be $(x_0^{\mathsf{false}}, x_1^{\mathsf{false}})$, respectively.
  - If $\sigma = 1$ (*parties learn true and false outputs*) do:
    - Toss another coin τ_1 uniformly at random from $\{0, 1\}$.
    - Set the output of P_{τ_1} to be $x_{\tau_1}^{\mathsf{true}}$.
    - Set the output of $P_{1-\tau_1}$ to be $x_{1-\tau_1}^{\mathsf{false}}$.

In the protocol, the parties compute the function f^α using security-with-abort. At the end of this computation, the adversary is the first to see the output. In case that the adversary deicides to abort, the honest party guesses its output at random from the two optional outputs. Intuitively, fairness is achieved since both parties learn the correct output with the same probability (1/2). On the other hand, in case where both parties play honestly, there is a correlation between the two outputs, as required from an α-correct protocol. For a full description of the protocol, together with a proof for the following theorem, see the full version [2].

Theorem 26. *Let f be a non-trivial two-party function. Then, for every $1/2 \geq \alpha \geq 0$, protocol π^{α} is an α-correct protocol in the f^{α}-hybrid model, and is simulatable (in the sense of Definition 22).*

References

1. Abraham, I., Dolev, D., Gonen, R., Halpern, J.Y.: Distributed Computing Meets Game Theory: Robust Mechanisms for Rational Secret Sharing and Multiparty Computation. In: PODC, pp. 53–62 (2006)
2. Asharov, G., Canetti, R., Hazay, C.: Towards a Game Theoretic View of Secure Computation (full version) (in ePrint)
3. Asharov, G., Lindell, Y.: Utility Dependence in Correct and Fair Rational Secret Sharing. Journal of Cryptology 24(1), 157–202 (2011)
4. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: 30th FOCS, pp. 468–473 (1989)
5. Canetti, R.: Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology 13(1), 143–202 (2000)
6. Cleve, R.: Limits on the Security of Coin Flips when Half the Processors are Faulty. In: 18th STOC, pp. 364–369 (1986)
7. Dodis, Y., Halevi, S., Rabin, T.: A Cryptographic Solution to a Game Theoretic Problem. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 112–130. Springer, Heidelberg (2000)
8. Dodis, Y., Rabin, T.: Cryptography and Game Theory. In: Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
9. Fuchsbauer, G., Katz, J., Naccache, D.: Efficient Rational Secret Sharing in Standard Communication Networks. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 419–436. Springer, Heidelberg (2010)
10. Fudenberg, D., Tirole, J.: Game Theory. The MIT Press, Cambridge (1991)
11. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
12. Goldreich, O.: Foundations of Cryptography. Basic Applications, vol. 2. Cambridge University Press, Cambridge (2004)
13. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game – A Completeness Theorem for Protocols with Honest Majority. In: 19th STOC, pp. 218–229 (1987)
14. Goldreich, O., Kahan, A.: How To Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology 9(3), 167–190 (1996)
15. Goldwasser, S., Levin, L.A.: Fair computation of general functions in presence of immoral majority. In: Menezes, A., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 77–93. Springer, Heidelberg (1991)
16. Goldwasser, S., Micali, S.: Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
17. Goldwasser, S., Micali, S., Rachoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Computing 18(1), 186–208 (1989)
18. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC, pp. 413–422 (2008)
19. Gordon, S.D., Katz, J.: Rational Secret Sharing, Revisited. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 229–241. Springer, Heidelberg (2006)

20. Gordon, S.D., Katz, J.: Partial Fairness in Secure Two-Party Computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 157–176. Springer, Heidelberg (2010)
21. Gradwohl, R., Livne, N., Rosen, A.: Sequential Rationality in Cryptographic Protocols. In: FOCS, pp. 623–632 (2010)
22. Halpern, J., Teague, V.: Efficient Rational Secret Sharing in Standard Communication Networks. In: 36th STOC, pp. 623–632 (2004)
23. Halpern, J., Pass, R.: Game Theory with Costly Computation. In: ICS, pp. 120–142 (2010)
24. Izmalkov, S., Lepinski, M., Micali, S.: Verifiably Secure Devices. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 273–301. Springer, Heidelberg (2008)
25. Izmalkov, S., Micali, S., Lepinski, M.: Rational Secure Computation and Ideal Mechanism Design. In: 46th FOCS, pp. 585–595 (2005)
26. Katz, J.: Bridging Game Theory and Cryptography: Recent Results and Future Directions. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 251–272. Springer, Heidelberg (2008)
27. Kol, G., Naor, M.: Games for exchanging information. In: 40th STOC, pp. 423–432 (2008)
28. 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)
29. Lysyanskaya, A., Triandopoulos, N.: Rationality and Adversarial Behavior in Multi-party Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 180–197. Springer, Heidelberg (2006)
30. Ong, S.J., Parkes, D.C., Rosen, A., Vadhan, S.P.: Fairness with an Honest Minority and a Rational Majority. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 36–53. Springer, Heidelberg (2009)
31. Pass, R., Shelat, A.: Renegotiation-Safe Protocols. In: Innovations in Computer Science, ICS 2011 (2011)

Highly-Efficient Universally-Composable Commitments Based on the DDH Assumption*

Yehuda Lindell

Department of Computer Science
Bar-Ilang University, Israel
lindell@cs.biu.ac.il

Abstract. Universal composability (a.k.a. UC security) provides very strong security guarantees for protocols that run in complex real-world environments. In particular, security is guaranteed to hold when the protocol is run concurrently many times with other secure and possibly insecure protocols. Commitment schemes are a basic building block in many cryptographic constructions, and as such universally composable commitments are of great importance in constructing UC-secure protocols. In this paper, we construct highly efficient UC-secure commitments from the standard DDH assumption, in the common reference string model. Our commitment stage is non-interactive, has a common reference string with $O(1)$ group elements, and has complexity of $O(1)$ exponentiations for committing to a group element (to be more exact, the effective cost is that of $23\frac{1}{3}$ exponentiations overall, for both the commit and decommit stages). We present a construction that is secure in the presence of static adversaries, and a construction that is secure in the presence of adaptive adversaries with erasures, where the latter construction has an effective additional cost of just $5\frac{1}{3}$ exponentiations.

1 Introduction

Background – universal composability and efficiency. Modern cryptographic protocols are run in complex environments. Many different secure and insecure protocols are executed concurrently, and some protocols may have been designed specifically to attack others [13]. The classic definitions of security that consider stand-alone executions only do not guarantee security in modern real-world setting. Universal composability (or UC security) is a definitional framework that guarantees security even if the protocol is run concurrently with arbitrarily many other secure and insecure protocols, and even if related inputs are used. More specifically, a UC-secure protocol behaves like an ideal execution (where an incorruptible trusted party carries out the computation for the parties) no matter what other protocols are being run by the honest parties at the time.

* This research was supported by the European Research Council as part of the ERC project "LAST", and by the ISRAEL SCIENCE FOUNDATION (grant No. 781/07).

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 446–466, 2011.

The UC-framework models the *real-world* execution environment in a far more realistic way than the classic stand-alone definitions. As such, one would expect the framework to be adopted by practitioners and those interested in implementing cryptographic protocols that could be run in practice. In the setting of key exchange this is indeed the case. For just two examples, the SIGMA family of key exchange protocols that are part of IKE (the standardized Internet key exchange protocol) and the HMQV protocol have been proven secure in the UC-framework [5,15]. However, beyond key exchange, there seems to have been little interest in UC-security from the applied cryptographic community. (We stress that this is in contrast to the recent growing interest in implementations of general and specific protocols for secure two-party and multiparty computation; see [17,2,23,20,19] for just a few examples.) There are a number of reasons for this. We believe that one of the primary reasons is the lack of *efficient* UC-secure primitives, the exception being UC-secure oblivious transfer [22]. Given this state of affairs, it is very difficult to construct efficient UC-secure protocols that can be reasonably implemented.

UC commitments. Commitment schemes are one of the most basic building blocks for cryptographic protocols. A commitment scheme is made up of two phases: a *commit phase* in which a committer commits to a value while keeping it hidden, and a *decommit phase* in which the committer reveals the value that it previously committed to. The binding property of a commitment states that the committer is bound to a single value after the commit phase and can only decommit to that value; the hiding property states that the receiver learns nothing about the committed value until it is revealed. As such, a commitment scheme has been intuitively described as a digital envelope containing the committed value: once the envelope has been closed the committer cannot change the value, and until the envelope is opened the receiver cannot learn what is inside. Despite this appealing description, regular commitments do not behave in this way. For just one example, they may be malleable (e.g., it may be possible to generate a commitment to $2x$ from a commitment to x, without knowing x). In contrast, UC-secure commitments are non-malleable, cannot be copied, and are guaranteed to remain secure irrespective of whatever other protocols are run.

Commitment schemes that are secure in the UC-framework were first presented by Canetti and Fischlin in [4]. They also showed that it is impossible to construct UC commitments in the plain model, and thus some setup like a common reference string is required. The commitment schemes of [4] have the property that $O(1)$ asymmetric operations are needed for every bit committed to. Soon after, Damgård and Nielsen [9] presented UC commitments with the property that $O(1)$ exponentiations are sufficient for committing to an entire string (that can be mapped into a group element). This is a significant improvement. However, the Damgård-Nielsen construction suffers from a few drawbacks. First, it requires a common reference string that grows linearly with the number of parties in the system; specifically, one group element is needed for every party. This is a significant obstacle in implementations because it means that it is not possible to publish a single common reference string that can then be used

by arbitrary parties who wish to run the protocol. Second, the Damgård-Nielsen constructions are based on the N-residuosity and p-subgroup assumptions. These are less established assumptions than RSA and DDH, for example. Furthermore the N-residuosity assumption, which has become accepted since its introduction in [21], suffers from a significant computational overhead. This is due to the fact that exponentiations are modulo N^2 (at least) and thus a modulus N of size 1536 – which is needed for reasonable security – results in exponentiations modulo a number of length *3072 bits*. In contrast, basic discrete log exponentiations can be run in Elliptic curves of size 224 or 256 bits and are significantly faster. In cryptographic protocols where many UC commitments are needed (see below for an example), this can be a real obstacle. **Our results.** We present a conceptually simple and efficient protocol for UC-secure commitments in the common reference string model that is based on the DDH assumption. Our protocol requires $O(1)$ regular group exponentiations and has a common reference string with $O(1)$ group elements for any number of parties. In addition, we have a version that provides security in the presence of adaptive adversaries with erasures that has only slightly additional cost. A comparison of our result with the construction of Damgård-Nielsen, which is the previous best known, yields the following:

- *Assumptions:* We rely on the standard DDH assumption, while Damgård-Nielsen rely on the N-residuosity or p-subgroup assumptions.
- *Common reference string (CRS):* Our common reference string contains a description of the discrete log group, its order and 7 group elements, and can be used by *any number of parties*. Thus, a single CRS can be published for all to use. In contrast, Damgård-Nielsen need a CRS with a single (ring or group) element for every party in the system.
- *Efficiency:* Our protocol has a non-interactive commitment phase with just 5 exponentiations to be computed by the committer. The decommit phase is interactive and requires both parties overall to compute 21 exponentiations. Using optimizations for computing multi-exponentiations of the form $g^r \cdot h^s$ the overall cost in both phases is $23\frac{1}{3}$ regular DDH exponentiations. In contrast Damgård-Nielsen have an interactive commitment phase with 10 large modulus exponentiations and a non-interactive decommit phase requiring 4 exponentiations[1]. Based on experiments, we estimate that our commitment scheme is approximately 25–30 times faster than the scheme of Damgård-Nielsen. (This estimate is not based on an implementation of the schemes, but rather a comparison of the time taken to compute 14 exponentiations $\mathrm{mod} N^2$ with a modulus N of size 2048 bits versus $23\frac{1}{3}$ Elliptic curve "exponentiations" over a field of size 256 bits, using the Crypto++ library [25]. When using a modulus N of size 1536 bits versus a curve over a field of size 224 bits, our scheme is approximately 20 times faster.)

[1] The question of whether there is more cost in the commitment or decommitment phase is significant in protocols of the cut-and-choose type where many commitments are sent and only some of them opened. In such cases, it is preferable to use a commitment scheme with a faster commitment phase.

– *Adaptive security:* The Damgård-Nielsen construction is secure only for static corruptions (where the set of corrupted parties is fixed before any commitments are sent), whereas we also have a construction that is secure in the presence of adaptive corruptions with erasures. (In this model, the adversary can choose to corrupt parties over time, but an honest party can follow erase instructions and it is assumed that once data is erased it cannot be retrieved by the adversary if it later corrupts the party.) The additional cost of achieving adaptive security is just $5\frac{1}{3}$ exponentiations, yielding a total of $28\frac{2}{3}$. In this case, however, the majority of the work is in the commitment stage and not in the decommitment stage.

An example – UC zero-knowledge from Sigma-protocols. Since our efficiency improvement over prior work is *concrete* and not asymptotic, we demonstrate its potential significance in implementations. We do this by considering the ramification of our efficiency improvement on constructions of efficient UC-secure zero-knowledge protocols. Many, if not most, useful efficient zero-knowledge protocols are based on Sigma protocols [8]. In the stand-alone case, transformations from Sigma protocols to zero-knowledge and zero-knowledge proofs of knowledge are highly efficient, requiring only a few additional exponentiations; see [10, Section 6.5]. Unfortunately, no such efficient analogue is known for achieving UC zero-knowledge from Sigma protocols. Rather, it is necessary to repeat the Sigma protocol L times in order to achieve a soundness error of 2^{-L}. In addition, 3 UC-commitments are needed for each repetition (but only two are opened); see [12] and [16, App. C] for a description of the transformation. Setting $L = 40$ for a reasonable soundness error, we have that 120 UC commitments are needed for the transformation. Assuming 47 milliseconds for our scheme and 1.35 seconds for Damgård Nielsen (based on estimates using the Crypto++ library), we have that the additional overhead resulting from the UC commitments is 5.6 seconds for our protocol versus 162 seconds for Damgård-Nielsen (the difference is actually even greater since 40 of the 120 commitments are not opened; see Footnote 1). We conclude that in *protocol implementations* the efficiency improvement gained by using our new UC commitment protocol can be definitive.

2 Preliminaries and Definitions

Universal composability [3] is a definition of security that considers a stand-alone execution of a protocol in a special setting involving an environment machine $\mathcal{Z}$, in addition to the honest parties and adversary. As with the classic definition of secure computation, ideal and real models are considered where a trusted party carries out the computation in the ideal model and the real protocol is run in the real model. The environment adaptively chooses the inputs for the honest parties, interacts with the adversary throughout the computation, and receives the honest parties' outputs. Security is formulated by requiring the existence of an ideal-model simulator $\mathcal{S}$ so that no environment $\mathcal{Z}$ can distinguish between the case that it runs with the real adversary $\mathcal{A}$ in the real model and the case that it runs with the ideal-model simulator $\mathcal{S}$ in the ideal model.

In slightly more detail, we denote by $\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(n,z)$ the output of the environment $\mathcal{Z}$ with input z after an ideal execution with the ideal adversary (simulator) $\mathcal{S}$ and functionality $\mathcal{F}$, with security parameter n. We will only consider black-box simulators $\mathcal{S}$, and so we denote the simulator by $\mathcal{S}^{\mathcal{A}}$ meaning that it works with the adversary $\mathcal{A}$ attacking the real protocol. Furthermore, we denote by $\text{REAL}_{\pi,\mathcal{A},\mathcal{Z}}(n,z)$ the output of environment $\mathcal{Z}$ with input z after a real execution of the protocol π with adversary $\mathcal{A}$, with security parameter n. Our protocols are in the common reference string (CRS) model. Formally, this means that the protocol π is run in a hybrid model where the parties have access to an ideal functionality $\mathcal{F}_{\text{CRS}}$ that chooses a CRS according to the prescribed distribution and hands it to any party that requests it. We denote an execution of π in such a model by $\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi,\mathcal{A},\mathcal{Z}}(n,z)$. Informally, a protocol π UC-realizes a functionality $\mathcal{F}$ in the $\mathcal{F}_{\text{CRS}}$ hybrid model if there exists a probabilistic polynomial-time simulator $\mathcal{S}$ such that for every non-uniform probabilistic polynomial-time environment $\mathcal{Z}$ and every probabilistic polynomial-time adversary $\mathcal{A}$, it holds that

$$\left\{\text{IDEAL}_{\mathcal{F},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(n,z)\right\}_{n\in\mathbb{N};z\in\{0,1\}^*} \stackrel{\text{c}}{\equiv} \left\{\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi,\mathcal{A},\mathcal{Z}}(n,z)\right\}_{n\in\mathbb{N};z\in\{0,1\}^*}.$$

The importance of this definition is a composition theorem that states that any protocol that is universally composable is secure when run concurrently with many other arbitrary protocols; see [3] for definitions and details. **UC commitments.** The multi-commitment ideal functionality $\mathcal{F}_{\text{MCOM}}$, which is the functionality that we UC realize in this paper, is formally defined in Figure 1.

FIGURE 1 (Functionality $\mathcal{F}_{\text{MCOM}}$)

$\mathcal{F}_{\text{MCOM}}$ proceeds as follows, running with parties $P_1,\ldots,P_m$, a parameter 1^n, and an adversary $\mathcal{S}$:

- **Commit phase:** Upon receiving a message $(\mathsf{commit}, sid, ssid, P_i, P_j, x)$ from P_i where $x \in \{0,1\}^{n-\log^2 n}$, record the tuple $(ssid, P_i, P_j, x)$ and send the messages $(\mathsf{receipt}, sid, ssid, P_i, P_j)$ to P_j and $\mathcal{S}$. Ignore any future commit messages with the same $ssid$ from P_i to P_j.
- **Reveal phase:** Upon receiving a message $(\mathsf{reveal}, sid, ssid)$ from P_i: If a tuple $(ssid, P_i, P_j, x)$ was previously recorded, then send the message $(\mathsf{reveal}, sid, ssid, P_i, P_j, x)$ to P_j and $\mathcal{S}$. Otherwise, ignore.

Fig. 1. The ideal commitment functionality

For technical reasons, the length of the committed value x is $n - \log^2 n$. It is defined in this way because our commitment involves encrypting x together with the session identifiers $sid, ssid$ and the parties' identities (i,j). Now, the encryption that we use is of a single group element that is of length n, and so the combined length of $x, sid, ssid, i, j$ must be n. We therefore define each identifier and identity to be of size $\frac{\log^2 n}{4}$; this means that each comes from a superpolynomial domain and so there are enough to ensure that the session identifiers do not repeat and each party has a unique identity. Thus, taking x of size $n - \log^2 n$ we have that the string $(x, sid, ssid, i, j)$ is of length n.

3 Efficient UC Commitments

3.1 Protocol Idea and Overview

Before describing the idea behind our construction, recall that a UC-secure commitment must be both *extractable* (meaning that a simulator can extract the value that a corrupted party commits to) and *equivocal* (meaning that a simulator can generate commitments that can be opened to any value), without the simulator rewinding the adversary. In addition, the adversary must not be able to generate commitments that are related to commitments generated by honest parties; thus, the commitment must be essentially non-malleable. Our protocol is in the common reference string (CRS) model; this is justified by the fact that UC commitments cannot be achieved in the plain model [4].

The high-level idea behind our construction is as follows. The committer commits to a string by encrypting it with a CCA2-secure encryption scheme E^{cca}, using a public-key pk_1 that is found in the common reference string (CRS). Observe that this enables extraction because when simulating a protocol that is in the CRS model, the simulator is allowed to choose the CRS itself. Thus, it can choose the public key so that it knows the corresponding private decryption key. This enables it to decrypt and obtain the committed value. Next, in order to decommit, it is clearly not possible to reveal the value and randomness used to encrypt, because encryptions are perfectly binding and so it is not possible to equivocate. Thus, in order to decommit, the committer instead sends the committed value and then *proves* in zero knowledge that this is indeed the correct value. At first sight, this approach may seem futile because in the UC setting it seems no easier to construct UC zero-knowledge than UC commitments. Nevertheless, we observe that the proof need not be a full fledged UC zero-knowledge protocol, and in particular there is no need to extract the witness from the proof. Rather, the only property that we need is that it be possible to simulate without rewinding. This is due to the fact that the extraction of the committed value already took place in the commit stage and this proof is just to ensure that corrupted parties decommit to the same value that they committed to. Thus, only soundness is necessary. (Of course, the ability for a simulator to equivocate is due to its ability to run a zero-knowledge simulator and essentially lie about the value committed to.) The proof that we use is based on a Sigma protocol and we make it zero knowledge (without rewinding) by having the verifier first commit to its challenge and then run the Sigma protocol with the verifier decommitting. In order to have a straight-line simulator we make this commitment from the verifier be an encryption of the challenge under a different public key pk_2 in the CRS. As above, in the simulation the simulator can choose the public-key so that it knows the corresponding private key, enabling it to extract the challenge from the verifier. Once it has extracted the challenge, it can run the simulator for the Sigma protocol which is perfect and straight line once given the verifier challenge. Although intuitively appealing, this is problematic because soundness of this transformation from a Sigma protocol to a zero-knowledge proof can only be proven if the commitment is *perfectly hiding*. But this then clashes with the requirement to have the commitment be extractable. We solve this efficiently

by using a *dual mode* cryptosystem E^{dual}, as introduced by [22][2], although we only need a simpler version. Such a cryptosystem has a regular key generation algorithm and an alternative one, and has the property that it behaves as a regular public-key encryption scheme when a regular key is generated, but perfectly hides the encrypted value when an alternative key is generated. Furthermore, the regular and alternative keys are indistinguishable. As we will see in the proof, this suffices for proving soundness, because at the point where soundness is needed we no longer need to be able to extract the verifier's challenge and thus can replace the key in the common reference string by an alternative one. Note that a regular key for E^{dual} is used in a real protocol execution, and the ability to generate an alternative key is used within the proof of security. (Note also that we cannot use this method for the actual UC commitment because we need to *simultaneously* extract and equivocate.)

The above yields the following template for UC commitments:

PROTOCOL 1 (UC-commitment template)

Common reference string: (pk_1, pk_2) where pk_1 is the public-key of a CCA2-secure encryption scheme, and pk_2 is the public-key of a dual mode cryptosystem, as described above.

The commit phase:

1. The committer commits to x by encrypting it under pk_1 and sending the ciphertext $c = E^{cca}_{pk_1}(x; r)$ to the receiver (i.e., it encrypts x using random coins r).
   (Actually, x is encrypted together with a unique session identifier and the identities of the parties, but we ignore these details here.)

The decommitment phase:

1. The committer sends x to the receiver (without revealing r)
2. Let (α, ε, z) denote the message of a Sigma protocol for proving that c is an encryption of x (using witness r).
   (a) The receiver sends $c' = E^{dual}_{pk_2}(\varepsilon; r')$
   (b) The committer sends α
   (c) The receiver decommits to ε by sending ε and r'
   (d) The committer checks that $c' = E^{dual}_{pk_2}(\varepsilon; r')$ and if yes, computes the reply z for the Sigma protocol, based on (α, ε)
   (e) The receiver outputs x as the decommitted value if and only if (α, ε, z) is an accepting Sigma-protocol transcript

Before proceeding, we explain why the value x is committed to by encrypting it under an encryption scheme that is secure under adaptive chosen-ciphertext attacks (CCA2 secure). Specifically, we have already discussed why some notion of non-malleability is needed, but CCA2-security is stronger than NM-CPA

[2] We use the formulation as it appears in [22], although the idea of having alternative keys that provide perfect hiding or regular encryption goes back earlier. Two examples of where similar notions were defined are in [11,14]. In fact, our construction of dual encryption is exactly the same as the ambiguous commitment used in [11].

(non-malleability under chosen plaintext attacks). In order to understand why we nevertheless need CCA2 security, recall that a simulator must equivocate. Specifically, in the simulation in the ideal model, the simulator receives commitment receipts that contain no information about the committed value. However, in the real world, the adversary receives encryptions of the actual committed value. Thus, whenever it receives a commitment receipt, the simulator encrypts 0 and hands it to the real-world adversary. Later, when the commitment is opened and the simulator learns that it was to a value x, it cheats in the Sigma protocol and "proves" that the encryption of 0 was actually an encryption of x. In order to prove that encrypting 0 (as the simulator does) and encrypting x (as an honest party does) makes no difference, it is necessary to reduce this to the security of the encryption scheme. In such a reduction, an adversary attacking the encryption scheme simulates the UC commitment execution such that if it received encryptions of 0 then the result should be the same as the ideal simulation, and if it received encryptions of real values x then the result should be the same as a real execution with honest parties and the real adversary. To be more exact, this reduction is carried out by running the simulator for the UC commitment scheme and using challenge ciphertexts obtained in the encryption game instead of the simulator generating commitments itself. Of course, in this reduction the simulator does not choose the CCA2-secure public key to place in the CRS but rather places the public-key that it receives as part of the encryption distinguishing game. However, as we have already discussed, the simulator must also be able to *extract* committed values generated by the adversary by decrypting, at the same time as we carry out this reduction. This brings us to the crux of the problem which is that it can only carry out this decryption because it knows the private key, and so it cannot decrypt when proving the reduction. This problem is solved by using CCA2-secure encryption because now in the distinguishing game the adversary is allowed to ask for decryptions of ciphertexts, and so the simulator can decrypt the commitments from the adversary, as required.

Efficient implementations. It remains to describe how all of the elements of the protocol can be efficiently implemented. First, we use the Cramer-Shoup (CS) encryption scheme [7] as the CCA2-secure encryption scheme. This scheme is defined as follows:

- **CS key generation:** Let $(\mathbb{G}, q, g_1, g_2)$ be such that $\mathbb{G}$ is a group of order q and g_1, g_2 are two distinct generators. Choose $x_1, x_2, y_1, y_2, z \in_R \mathbb{Z}_q$ at random and compute $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$ and $h = g_1^z$. The public key is $(\mathbb{G}, q, g_1, g_2, c, d, h)$ and the secret key is (x_1, x_2, y_1, y_2, z).
- **CS encryption:** Let $m \in G$. Then, in order to encrypt m, choose a random $r \in_R \mathbb{Z}_q$, compute $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r \cdot m$, $\omega = H(u_1, u_2, e)$ where H is a collision-resistant hash function, and $v = (c \cdot d^\omega)^r$. The ciphertext is (u_1, u_2, e, v).
- **CS decryption:** Compute $\omega = H(u_1, u_2, e)$. If $u_1^{x_1} \cdot u_2^{x_2} \cdot (u_1^{y_1} \cdot u_2^{y_2})^\omega = v$, then output $m = e/(u_1^z)$.

The crucial observation that we make is that in order to verify that a ciphertext (u_1, u_2, e, v) is a valid encryption of a message m, it suffices to prove that there exists a value $r \in \mathbb{Z}_q$ such that

$$u_1 = g_1^r, \quad u_2 = g_2^r, \quad \frac{e}{m} = h^r, \quad \text{and} \quad v = (cd^\omega)^r.$$

Furthermore, since ω can be computed publicly from the public-key and ciphertext, all the values except for r are public. Thus, we have that in order to prove that a ciphertext encrypts some given value m, we just need to run a proof that 4 values have the same discrete log with respect to their respective bases. Highly efficient Sigma protocols exist for this task (this is the same as proving that a tuple is of the Diffie-Hellman form). Thus, the CCA2-secure encryption scheme together with the required proof can both be implemented very efficiently.

It remains to show how a dual-model encryption scheme can be efficiently implemented. We essentially use the construction of [22], but we need only their basic cryptosystem and not their full dual-mode one. Specifically, we need the ability to construct a fake public-key that is indistinguishable from a regular one, so that if encryption is carried out under this key, then the encrypted value is perfectly hidden. Such an encryption scheme can be constructed at double the cost of El Gamal as follows:

- **Dual regular key generation:** Let $(\mathbb{G}, q, g_1, g_2)$ be as above. Choose $\rho \in_R \mathbb{Z}_q$ and compute $h_1 = g_1^\rho$ and $h_2 = g_2^\rho$. The public key is $(\mathbb{G}, q, g_1, g_2, h_1, h_2)$, and the private key is ρ.
- **Dual alternative key generation:** As above, except choose $\rho_1, \rho_2 \in_R \mathbb{Z}_q$ with $\rho_1 \neq \rho_2$ and compute $h_1 = g_1^{\rho_1}$ and $h_2 = g_2^{\rho_2}$.
- **Dual encryption:** To encrypt $m \in G$, choose random $R, S \in \mathbb{Z}_q$ and compute $u = g_1^R \cdot g_2^S$ and $v = h_1^R \cdot h_2^S \cdot m$. The ciphertext is $c = (u, v)$.
- **Dual decryption:** To decrypt (u, v), compute $m = v/u^\rho$.

In order to see that this scheme has the desired properties, observe that decryption works as in El Gamal, an alternative key is indistinguishable from a real one by the DDH assumption, and encryption under an alternative key is perfectly hiding since when $\rho_1 \neq \rho_2$ the values u and v are independent.

Naively, the cost of encryption is 4 exponentiations which is twice that of El Gamal. However, using the method of simultaneous multiple exponentiations in [18, Sec. 14.6], we have that u and v can be computed at the equivalent cost of $1\frac{1}{3}$ exponentiations each. Thus, the cost is $2\frac{2}{3}$ exponentiations.

3.2 The Actual Protocol

The full specification of our commitment scheme appears in Protocol 2. The proof carried out in the decommitment phase is based on a Sigma protocol for Diffie-Hellman tuples. Regarding completeness of this proof, observe that if P_i is honest, then $g_1^z = g_1^{s+\varepsilon r} = g_1^s \cdot (g_1^r)^\varepsilon = \alpha \cdot u_1^\varepsilon$, $g_2^z = g_2^{s+\varepsilon r} = g_2^s \cdot (g_2^r)^\varepsilon = \beta \cdot u_2^\varepsilon$, $h^z = h^{s+\varepsilon r} = h^s \cdot (h^r)^\varepsilon = \gamma \cdot \left(\frac{e}{m}\right)^\varepsilon$, and

$$(cd^\omega)^z = (cd^\omega)^{s+\varepsilon r} = (cd^\omega)^s \cdot ((cd^\omega)^r)^\varepsilon = (cd^\omega)^s \cdot (c^r d^{r\omega})^\varepsilon = \delta \cdot v^\varepsilon.$$

PROTOCOL 2 (UC-Secure Commitment Protocol)

Common reference string: $(\mathbb{G}, q, g_1, g_2, c, d, h, h_1, h_2)$ where $\mathbb{G}$ is a group of order q with generators g_1, g_2, and $c, d, h \in_R \mathbb{G}$ are random elements of $\mathbb{G}$, and $h_1 = g_1^\rho$, $h_2 = g_2^\rho$ for a random $\rho \in_R \mathbb{Z}_q$. (Note that $(\mathbb{G}, q, g_1, g_2, c, d, h)$ is a Cramer-Shoup public key, and $(\mathbb{G}, q, g_1, g_2, h_1, h_2)$ is the regular public key of a dual-mode encryption scheme.)
Let $G(y)$ be a mapping of a string $y \in \{0,1\}^n$ to $\mathbb{G}$, and assume that G^{-1} is also efficiently computable.

The commit phase: Upon input $(\mathsf{commit}, sid, ssid, P_i, P_j, x)$ where $x \in \{0,1\}^{n-\log^2 n}$ and $sid, ssid \in \{0,1\}^{\log^2 n/4}$, party P_i works as follows:

1. P_i computes $m = G(x, sid, ssid, i, j)$. (The identities i, j can be mapped to $\{0,1\}^{\log^2 n/4}$ and so overall $(x, sid, ssid, i, j)$ is an n-bit string.)
2. P_i chooses a random $r \in_R \mathbb{Z}_q$, computes $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r \cdot m$, $\omega = H(u_1, u_2, e)$ and $v = c^r \cdot d^{r\omega}$, where H is a collision-resistant hash function (formally, the key for the hash function can appear in the CRS; we ignore this for simplicity).
3. P_i sets $c = (u_1, u_2, e, v)$, and sends $(sid, ssid, c)$ to P_j.
4. P_j stores $(sid, ssid, P_i, P_j, c)$ and outputs $(\mathsf{receipt}, sid, ssid, P_i, P_j)$. P_j ignores any later commitment messages with the same $(sid, ssid)$ from P_i.

The decommit phase:

1. Upon input $(\mathsf{reveal}, sid, ssid, P_i, P_j)$, party P_i sends $(sid, ssid, x)$ to P_j
2. P_j computes $m = G(x, sid, ssid, i, j)$
3. *Proof of committed value:* P_i proves to P_j that m is the encrypted value. This is equivalent to P_i proving that there exists a value r such that

$$u_1 = g_1^r, \quad u_2 = g_2^r, \quad \frac{e}{m} = h^r, \quad \text{and} \quad v = (cd^\omega)^r$$

The proof is carried out as follows:

(a) P_j sends $(sid, ssid, c')$ to P_i, where $c' = (g_1^R \cdot g_2^S, h_1^R \cdot h_2^S \cdot G(\varepsilon))$ is a commitment to a random challenge $\varepsilon \in_R \{0,1\}^n$, and $R, S \in_R \mathbb{Z}_q$.

(b) P_i computes $\alpha = g_1^s$, $\beta = g_2^s$, $\gamma = h^s$ and $\delta = (cd^\omega)^s$, and sends $(sid, ssid, \alpha, \beta, \gamma, \delta)$ to P_j.

(c) P_j sends the decommitment $(sid, ssid, R, S, \varepsilon)$ to the challenge to P_i.

(d) P_i verifies that $c' = (g_1^R \cdot g_2^S, h_1^R \cdot h_2^S \cdot G(\varepsilon))$. If no, P_i aborts. Otherwise, P_i computes $z = s + \varepsilon r$ and sends $(sid, ssid, z)$ to P_j.

(e) P_j outputs $(\mathsf{reveal}, sid, ssid, P_i, P_j, x)$ if and only if

$$g_1^z = \alpha \cdot u_1^\varepsilon, \quad g_2^z = \beta \cdot u_2^\varepsilon, \quad h^z = \gamma \cdot \left(\frac{e}{m}\right)^\varepsilon, \quad \text{and} \quad (cd^\omega)^z = \delta \cdot v^\varepsilon$$

Concrete efficiency: The cost of the protocol in the number of exponentiations (all other operations are insignificant) is as follows:

1. P_i computes 5 exponentiations in order to generate the commitment, and 8 exponentiations in the decommit phase (note that 4 of these exponentiations are in order to verify the challenge ε from P_j, and since cd^ω was already computed in the commit stage only a single exponentiation is needed for δ).
2. P_j computes 0 exponentiations in the commit phase, and 13 exponentiations in the decommit phase.

Overall, the parties compute 26 exponentiations. Observe that P_i can preprocess all but 6 of its exponentiations. This is because it can compute $g_1^r, g_2^r, h^r, c^r, d^r$ and $g_1^s, g_2^s, h^s, c^s, d^s$ before m and thus ω is known. Once $(x, sid, ssid, P_i, P_j)$ is given and thus m can be computed, P_i just needs to compute $(d^r)^\omega$ to finish the commitment and $(d^s)^\omega$ to finish the first message of the decommit stage. Finally, it needs 4 more exponentiation to verify the ε sent by P_j Likewise, P_j can preprocess 4 of its exponentiations by generating c' ahead of time. We conclude that the protocol requires 26 exponentiations overall, but using preprocessing the committer P_i needs to compute only 6 exponentiations and the receiver P_j needs to compute only 9 exponentiations. In addition, the computations $g_1^R \cdot g_2^S$ and $h_1^R \cdot h_2^S$ needed for computing and verifying the encryption of ε can be computed at the cost of $1\frac{1}{3}$ exponentiations each, using the optimization appearing in [18, Sec. 14.6]. Thus, the effective number of exponentiations can be reduced to $23\frac{1}{3}$.

3.3 Proof of Security

Theorem 1. *Assuming that the DDH assumption holds in the group $\mathbb{G}$, Protocol 2 UC-securely realizes the $\mathcal{F}_{\text{MCOM}}$ functionality in the $\mathcal{F}_{\text{CRS}}$-hybrid model, in the presence of static adversaries.*

Proof: The intuition behind the proof of security already appears in Section 3.1. We therefore proceed directly to the description of the simulator and the proof of security.

The simulator $\mathcal{S}$:

- **Initialization step:** $\mathcal{S}$ chooses a public-key/private-key pair for the Cramer-Shoup cryptosystem; let $(\mathbb{G}, q, g_1, g_2, c, d, h)$ be the public-key. In addition, $\mathcal{S}$ chooses a random ρ and computes $h_1 = g_1^\rho$ and $h_2 = g_2^\rho$. $\mathcal{S}$ sets the CRS to be $(\mathbb{G}, q, g_1, g_2, c, d, h, h_1, h_2)$.
- **Simulating the communication with $\mathcal{Z}$:** Every input value that $\mathcal{S}$ receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.
- **Simulating the commit stage when the committer P_i is corrupted and the receiver P_j is honest:** Upon receiving $(sid, ssid, c)$ from $\mathcal{A}$ as it intends to send from P_i to P_j, the simulator $\mathcal{S}$ uses its knowledge of the Cramer-Shoup secret key to decrypt c. Let $m = G(x, sid', ssid', i', j')$ be the result. If $(sid', ssid', i', j') \neq (sid, ssid, i, j)$ or the decryption is invalid, then $\mathcal{S}$ sends a dummy commitment $(\mathsf{commit}, sid, ssid, P_i, P_j, 0)$ to $\mathcal{F}_{\text{MCOM}}$. Otherwise, $\mathcal{S}$ sends $(\mathsf{commit}, sid, ssid, P_i, P_j, x)$ to $\mathcal{F}_{\text{MCOM}}$.
- **Simulating the decommit stage when P_i is corrupted and P_j is honest:** $\mathcal{S}$ runs the honest strategy of P_j with $\mathcal{A}$ controlling P_i. If P_j would output $(\mathsf{reveal}, sid, ssid, P_i, P_j, x)$, then $\mathcal{S}$ sends $(\mathsf{reveal}, sid, ssid, P_i, P_j)$ to $\mathcal{F}_{\text{MCOM}}$. Otherwise, it does nothing.
- **Simulating the commit stage when P_i is honest and P_j is corrupted:** Upon receiving $(\mathsf{receipt}, sid, ssid, P_i, P_j)$ from $\mathcal{F}_{\text{MCOM}}$, the simulator $\mathcal{S}$ computes a Cramer-Shoup encryption c of 0, and hands $(sid, ssid, c)$ to $\mathcal{A}$, as it expects to receive from P_i.

– **Simulating the decommit stage when P_i is honest and P_j is corrupted:** Upon receiving $(\mathsf{reveal}, sid, ssid, P_i, P_j, x)$ from $\mathcal{F}_{\text{MCOM}}$, $\mathcal{S}$ works as follows:
  1. $\mathcal{S}$ hands $(sid, ssid, x)$ to $\mathcal{A}$, as it expects to receive from P_i.
  2. $\mathcal{S}$ receives c' from $\mathcal{A}$ and uses its knowledge of the discrete log ρ of h_1, h_2 (in the CRS) in order to decrypt the encryption c' of $G(\varepsilon)$ and obtain ε.
  3. Let $c = (u_1, u_2, e, v)$ be as computed by $\mathcal{S}$ in the commit stage. $\mathcal{S}$ chooses a random $z \in_R \mathbb{Z}_q$ and computes $\alpha = g_1^z/u_1^\varepsilon$, $\beta = g_2^z/u_2^\varepsilon$, $\gamma = h^z/(e/m)^\varepsilon$ and $\delta = (cd^\omega)^z/v^\varepsilon$, and hands $(\alpha, \beta, \gamma, \delta)$ to $\mathcal{A}$.
  4. $\mathcal{S}$ receives (R', S', ε') from $\mathcal{A}$. If $c' \neq (g_1^{R'} \cdot g_2^{S'}, h_1^{R'} \cdot h_2^{S'} \cdot G(\varepsilon'))$ then $\mathcal{S}$ simulates P_i aborting the decommitment. Otherwise, $\varepsilon' = \varepsilon$ (this must be the case because when the regular public-key of the dual encryption scheme is used the encryption is perfectly binding), and $\mathcal{S}$ hands z to $\mathcal{A}$.

  Simulation in the cases that both P_i and P_j are honest is straightforward. This is due to the fact that when both parties are honest, the simulator can choose the value ε itself and generate a valid proof for any value needed.

Analysis of the simulation: Denoting Protocol 2 by π and recalling that it runs in the $\mathcal{F}_{\text{CRS}}$-hybrid model, we need to prove that for every $\mathcal{A}$ and every $\mathcal{Z}$,

$$\left\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}}, \mathcal{S}^{\mathcal{A}}, \mathcal{Z}}(n, z)\right\}_{n \in \mathbb{N}; z \in \{0,1\}^*} \stackrel{\text{c}}{\equiv} \left\{\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi, \mathcal{A}, \mathcal{Z}}(n, z)\right\}_{n \in \mathbb{N}; z \in \{0,1\}^*}.$$

We prove this via a series of hybrid games.

Hybrid game HYB-GAME^1: In this game, the ideal functionality gives the simulator $\mathcal{S}_1$ the value x committed to by an honest P_i together with the regular $(\mathsf{receipt}, sid, ssid, P_i, P_j)$ message. $\mathcal{S}_1$ works in exactly the same way as $\mathcal{S}$ except that when simulating the commit stage when P_i is honest and P_j is corrupted, it computes c as an encryption of $m = G(x, sid, ssid, i, j)$ as an honest P_i would. Otherwise, it behaves exactly as $\mathcal{S}$ in the simulation. In order to show that the output of $\mathcal{Z}$ in HYB-GAME^1 is indistinguishable from its output in IDEAL, we need to reduce the difference to the security of the encryption scheme. However, $\mathcal{S}$ and $\mathcal{S}_1$ need to decrypt in the simulation of the commit stage when the committer P_i is corrupted and P_j is honest (see the simulator description). $\mathcal{S}$ and $\mathcal{S}_1$ can carry out this decryption because they know the Cramer-Shoup secret-key. But, this means that security cannot be reduced to this scheme. We solve this problem by using the fact that the Cramer-Shoup encryption scheme is CCA2-secure. Thus, $\mathcal{S}$ and $\mathcal{S}_1$ can decrypt by using their decryption oracle. We use the LR-formulation of CCA2-security [1]. In this formulation a bit b is randomly chosen and the adversary can ask for many *encryption challenges*. Each query consists of a pair (m_0, m_1) and the adversary receives back an encryption of m_b (always with the same b). The aim of the adversary is to guess the bit b. Of course, given that this is a CCA2 game, the adversary can ask for a decryption of any ciphertext that was not received as an encryption of one of the pairs.

Formally, we construct a CCA2 adversary $\mathcal{A}_{\text{CS}}$ attacking the Cramer-Shoup scheme as follows. Let $(\mathbb{G}, q, g_1, g_2, c, d, h)$ be the public-key given to $\mathcal{A}_{\text{CS}}$. Adversary $\mathcal{A}_{\text{CS}}$ chooses $\rho \in_R \mathbb{Z}_q$, computes $h_1 = g_1^\rho$ and $h_2 = g_2^\rho$, and sets the CRS to be

$(\mathbb{G}, q, g_1, g_2, c, d, h, h_1, h_2)$. Then $\mathcal{A}_{\text{CS}}$ simulates an execution of $\text{IDEAL}_{\mathcal{F}_{\text{MCOM}},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}$ with the following differences:

1. Whenever an honest P_i commits to a value x, instead of $\mathcal{S}$ encrypting 0 (or $\mathcal{S}_1$ encrypting x), $\mathcal{A}_{\text{CS}}$ generates the encryption in the ciphertext by asking for an encryption challenge of the pair $(0, G(x, sid, ssid, i, j))$. The ciphertext c received back is sent as the commitment. (Note that $\mathcal{A}_{\text{CS}}$ knows x because it runs $\mathcal{Z}$ and so knows the inputs handed to the honest parties.)
2. Whenever a corrupted P_i sends a commitment value $(sid, ssid, c)$ and the simulator needs to decrypt c, $\mathcal{A}_{\text{CS}}$ queries its decryption oracle with c. If c was received as a ciphertext challenge then $\mathcal{A}_{\text{CS}}$ has the simulator send a dummy commitment $(\mathsf{commit}, sid, ssid, P_i, P_j, 0)$ to $\mathcal{F}_{\text{MCOM}}$ as in the case that $(sid', ssid', i', j') \neq (sid, ssid, i, j)$ in the simulation. Since c was received as a ciphertext challenge, indeed it holds that $(sid', ssid', i', j') \neq (sid, ssid, i, j)$ and so this is the same.

Finally, $\mathcal{A}_{\text{CS}}$ outputs whatever $\mathcal{Z}$ outputs.

Now, if $b = 0$ in the CCA2 game, then all of the commitments c generated when the committer P_i is honest are to 0. Thus, the simulation is exactly like $\mathcal{S}$ and the output of $\mathcal{A}_{\text{CS}}$ is exactly that of $\text{IDEAL}_{\mathcal{F}_{\text{MCOM}},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(n, z)$. (Note that all other instructions are carried out identically to $\mathcal{S}$.) In contrast, if $b = 1$, then the commitments generated are to the correct values x and so the simulation is exactly like $\mathcal{S}_1$. Thus, the output of $\mathcal{A}_{\text{CS}}$ is exactly that of $\text{HYB-GAME}^1_{\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(n, z)$. We conclude that

$$\left\{\text{HYB-GAME}^1_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(n, z)\right\}_{n;z} \stackrel{\text{c}}{\equiv} \left\{\text{IDEAL}_{\mathcal{F}_{\text{MCOM}},\mathcal{S}^{\mathcal{A}},\mathcal{Z}}(n, z)\right\}_{n;z},$$

by the fact that the Cramer-Shoup encryption scheme is CCA2-secure.

Hybrid game HYB-GAME^2: In this game, the simulator $\mathcal{S}_2$ works in exactly the same way as $\mathcal{S}_1$, except that when simulating the decommitment phase when P_i is honest and P_j is corrupted, it computes the messages $(\alpha, \beta, \gamma, \delta)$ and z in the proof exactly as an honest P_i would. It can do this because the commitment c sent in the commitment phase is to the correct value $m = G(x, sid, ssid, i, j)$ and so it can play the honest prover. The output distribution of this game is *identical* to HYB-GAME^1 by the perfect simulation property of the proof of the decommitment phase. This proof is based on a standard Sigma protocol that a tuple is a Diffie-Hellman tuple and it is straightforward to verify that the distributions are identical. We therefore have that: We conclude that

$$\left\{\text{HYB-GAME}^2_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(n, z)\right\}_{n;z} \equiv \left\{\text{HYB-GAME}^1_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_1^{\mathcal{A}},\mathcal{Z}}(n, z)\right\}_{n;z}.$$

Completing the proof: It remains to show that the output of $\mathcal{Z}$ after an execution of π in the $\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}$ model is indistinguishable from its output after the HYB-GAME^2 game. First, observe that the commitment and decommitment messages in the case of an honest committer P_i are identical in both HYB-GAME^2 and a real protocol execution in the $\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}$ model. Thus, the only difference between the output of $\mathcal{Z}$ in both cases can be due to the value x output by an honest receiver P_j after a decommit from a corrupted sender P_i. This is due to the

fact that in HYB-GAME^2, the value x output by an honest P_j is the value sent by $\mathcal{S}_2$ to $\mathcal{F}_{\text{MCOM}}$ after decrypting the associated ciphertext in the commit stage using the Cramer-Shoup secret-key. In contrast, in HYBRID$^{\mathcal{F}_{\text{CRS}}}$ the value x output by an honest party is that sent by $\mathcal{A}$ in the first step of the decommitment stage (as long as the proof passes). These values can only be different if $\mathcal{A}$ can convince an honest P_j to output x in the decommitment phase, even though the encrypted value c sent in the commitment phase is not to $m = G(x, sid, ssid, i, j)$. Thus, this difference reduces to the *soundness* of the proof in the decommitment phase. Recall that by the special soundness property of Sigma protocols, in the case that c is *not* an encryption of $m = G(x, sid, ssid, i, j)$, for every first message $(\alpha, \beta, \gamma, \delta)$ there is only a *single* ε for which there exists a convincing answer z.

It is tempting to conclude that since the encryption of ε is semantically secure, the adversary cannot cheat in the Sigma protocol. However, this requires a reduction and such a reduction cannot be carried out because the adversary does not "reveal" to us whether it succeeds in the proof until we decrypt ε. Thus, one cannot reduce the ability of the adversary to cheat to the hiding of ε (in such a reduction, one cannot reveal ε together with the randomness used to encrypt). However, it *is* possible to replace the values h_1, h_2 where $h_1 = g_1^{\rho}$ and $h_2 = g_2^{\rho}$ with values $h_1 = g_1^{\rho_1}$ and $h_2 = g_2^{\rho_2}$ for $\rho_1, \rho_2 \in_R \mathbb{Z}_q$. In such a case, as we have discussed, the encryption c' perfectly hides the value ε. Furthermore, there is no need to ever decrypt c' here (the simulator $\mathcal{S}_2$ in HYB-GAME^2 does not decrypt these values). Thus, there is no problem replacing h_1, h_2 in this way. Finally, recall that the alternative key h_1, h_2 is indistinguishable from the regular one. Thus, defining HYB-GAME^3 to be the same as HYB-GAME^2 except that the keys h_1, h_2 are different as described, and letting $\mathcal{S}_3 = \mathcal{S}_2$ (except again for how h_1, h_2 are chosen), we have

$$\left\{\text{HYB-GAME}^3_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_3^{\mathcal{A}},\mathcal{Z}}(n,z)\right\}_{n;z} \stackrel{c}{\equiv} \left\{\text{HYB-GAME}^2_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(n,z)\right\}_{n;z}.$$

We are now ready to conclude the proof. Since the encryption c' perfectly hides the challenge ε, the probability that $\mathcal{A}$ successfully proves an incorrect statement in the decommitment stage is at most 2^{-n} (recall that there is exactly one ε that it can answer). Thus, the value sent by $\mathcal{S}_3$ to $\mathcal{F}_{\text{MCOM}}$ is the same value as that output by an honest P_j, except with negligible probability. The only other difference is that in HYB-GAME^3 an alternative public-key for the dual mode cryptosystem is used, whereas in HYBRID a regular one is used. Recalling that these keys are computationally indistinguishable, we conclude that

$$\left\{\text{HYBRID}^{\mathcal{F}_{\text{CRS}}}_{\pi,\mathcal{A},\mathcal{Z}}(n,z)\right\}_{n\in\mathbb{N};z\in\{0,1\}^*} \stackrel{c}{\equiv} \left\{\text{HYB-GAME}^3_{\mathcal{F}_{\text{MCOM}},\mathcal{S}_2^{\mathcal{A}},\mathcal{Z}}(n,z)\right\}_{n\in\mathbb{N};z\in\{0,1\}^*}.$$

Combining all of the above, we have that the output of $\mathcal{Z}$ with $\mathcal{A}$ after an execution of π in the $\mathcal{F}_{\text{CRS}}$-hybrid model is computationally indistinguishable from its output after an execution with $\mathcal{S}^{\mathcal{A}}$ and $\mathcal{F}_{\text{MCOM}}$ in the IDEAL model, and so Protocol 2 is UC-secure in the presence of static adversaries, as required. ■

4 Adaptive Adversaries with Erasures

4.1 Background and Outline of Solution

In the setting of adaptive corruptions, the adversary can corrupt parties throughout the computation. Upon corruption, it receives the local state of the parties, including randomness it has used and so on. In the model with erasures, a protocol can instruct a party to erase some of its state (e.g., old keys), and in such a case the adversary does not obtain the erased state upon corruption. Adaptive corruptions accurately models the realistic setting where parties can be "hacked" during a computation. As such, it is desirable to have protocols that are secure in this model.

This model introduces significant difficulties when proving security. Specifically, observe that Protocol 2 is not secure in the presence of adaptive adversaries, even with erasures, because the committer must store the randomness r used to commit to x in order to run the decommitment stage. Now, in our simulation, the simulator commits to 0, even when the commitment is really to x. However, upon corruption in the real world, the adversary obtains r and x such that c is encryption of x using randomness r. In the simulation, such randomness can never be produced because c is an encryption of 0 and not of x (there does not exist an r' that can explain c as an encryption of $x \neq 0$).

Achieving adaptive security. Our protocol can be modified so that it achieves adaptive security with erasures, with little additional cost. Interestingly, the only modifications necessary are a change in the order of operations and 1 additional Pedersen commitment. In order to see this, recall that the problem with achieving adaptive security is that the committer cannot erase r before sending c in the commit phase, because then it will not be able to prove the proof in the decommit phase. However, it is possible for the parties to run most of the proof already in the commit phase, *before* the commitment is even sent (actually, the ciphertext c is committed to equivocally, but not yet revealed). That is, the committer and receiver run the zero-knowledge protocol before c is sent, without the committer sending the last message z. In addition, the committer commits to its first message $(\alpha, \beta, \gamma, \delta)$ of the protocol instead of sending it in the clear. (Thus, the receiver sends a commitment to ε; the committer sends a commitment to $(\alpha, \beta, \gamma, \delta)$; the receiver decommits revealing ε; finally, the committer prepares z based on ε without sending it.) Following this preamble, the committer erases all of its randomness, except for that needed to decommit to the first message of the zero-knowledge protocol, and only then reveals c. This completes the commit phase. The decommit phase simply consists of the committer sending the decommitment to $(\alpha, \beta, \gamma, \delta)$ and the message z (which has already been prepared), and the receiver verifies the decommitment and that $((\alpha, \beta, \gamma, \delta), \varepsilon, z)$ constitutes an accepting transcript.

Observe that before the committer sends c, nothing has actually been revealed; the committer only sent a commitment to $(\alpha, \beta, \gamma, \delta)$. Thus, this does not affect the hiding property of the original commitment scheme. Furthermore, the committer erases all secret state before sending c, and in particular erases the

random coins used to generate c. Thus adaptive corruptions make no difference because the committer has no secret state once c is sent, and has revealed no information before c is sent. In actuality, in order to achieve this property that all messages sent before c are independent of x, we have to have the committer commit to the first message of the proof using a *perfectly hiding* commitment scheme. Furthermore, it needs to be adaptively secure in that upon corruption, the prover can open it to anything that it wishes. Fortunately, this can be easily achieved by using a Pedersen commitment $Com(x) = g^r \cdot \hat{h}^x$ with a value $\hat{h}$ that appears in the CRS[3]. (Note that given the discrete log $\hat{\rho}$ of $\hat{h}$ it is possible to decommit to any value desired. Specifically, commit by computing $c = g^a$ for a known a. Now, given x we wish to find r such that $c = g^a = g^r \cdot \hat{h}^x$. Given that $\hat{h} = g^{\hat{\rho}}$ this means that we need to find r such that $g^a = g^{r+\hat{\rho}x}$, or equivalently r such that $a = r + \hat{\rho}x \bmod q$. Thus, just take $r = a - \hat{\rho}x \bmod q$.) We remark that although the above works, it introduces an additional difficulty because the soundness of the Sigma protocol now also rests on the hardness of finding the discrete log of $\hat{h}$. This requires an additional reduction; see the proof for details. See Protocol 3 for the outline of the modified protocol.

PROTOCOL 3 (UC-commitment template for adaptive security)

Common reference string: $(\boldsymbol{pk_1}, \boldsymbol{pk_2}, \mathbb{G}, \boldsymbol{q}, \boldsymbol{g}, \hat{\boldsymbol{h}})$ where pk_1 is the public-key of a CCA2-secure encryption scheme, pk_2 is the public-key of a dual mode cryptosystem, and $(\mathbb{G}, q, g, \hat{h})$ are parameters for the Pedersen commitment scheme.

The commit phase:

1. The committer computes a commitment to x as $c = E^{cca}_{pk_1}(x; r)$, and sends a Pedersen commitment of c to the receiver, using $(g, \hat{h})$.
2. The receiver sends $c' = E^{dual}_{pk_2}(\varepsilon; r')$; its commitment to the first message of the proof.
3. The committer sends a Pedersen commitment to the first prover message α to the receiver, computed from the ciphertext c (observe that c has not yet been revealed).
4. The receiver decommits to ε by sending ε and r'
5. The committer checks that $c' = E^{dual}_{pk_2}(\varepsilon; r')$ and if yes, it computes the reply z for the Sigma protocol, based on (α, ε).
6. The committer now *erases* r and the randomness used to generate α and z, stores α, z and the randomness used to generate the Pedersen commitments, and finally sends c and its decommitment to the receiver.

The decommitment phase:

1. The committer sends x, (α, z), and the randomness used to generate the Pedersen commitment to α to the receiver.
2. The receiver outputs x as the decommitted value if and only if the Pedersen commitment was to α and (α, ε, z) is an accepting Sigma-protocol transcript.

[3] We note that such a commitment is not extractable but we do not need it to be.

4.2 The Adaptive Protocol

The scheme that is adaptively secure with erasures appears in Protocol 4.

PROTOCOL 4 (UC-Secure Commitment – Adaptive with Erasures)

Common reference string: $(\mathbb{G}, q, g_1, g_2, c, d, h, h_1, h_2, \hat{h})$ where all parameters are as in Protocol 2, and $(\mathbb{G}, q, g_1, \hat{h})$ are parameters for Pedersen commitments.)

The commit phase: Upon input $(\mathsf{commit}, sid, ssid, P_i, P_j, x)$ where $x \in \{0,1\}^{n-\log^2 n}$ and $sid, ssid \in \{0,1\}^{\log^2 n/4}$, party P_i works as follows:

1. P_i computes $m = G(x, sid, ssid, i, j)$. (The identities i, j can be mapped to $\{0,1\}^{\log^2 n/4}$ and so overall $(x, sid, ssid, i, j)$ is an n-bit string.)
2. P_i chooses a random $r \in_R \mathbb{Z}_q$, computes $u_1 = g_1^r$, $u_2 = g_2^r$, $e = h^r \cdot m$, $\omega = H(u_1, u_2, e)$ and $v = c^r \cdot d^{r\omega}$, where H is a collision-resistant hash function (formally, the key for the hash function can appear in the CRS; we ignore this for simplicity). P_i sets $c = (u_1, u_2, e, v)$.
3. P_i chooses $\kappa_1 \in_R \mathcal{Z}_q$, computes $c^1_{ped} = g_1^{\kappa_1} \cdot \hat{h}^{H(c)}$, and sends c^1_{ped} to P_j.
4. P_j sends $c' = (g_1^R \cdot g_2^S, h_1^R \cdot h_2^S \cdot G(\varepsilon))$ to P_i, where $\varepsilon \in_R \{0,1\}^n$ and $R, S \in_R \mathbb{Z}_q$.
5. P_i computes $\alpha = g_1^s$, $\beta = g_2^s$, $\gamma = h^s$ and $\delta = (cd^\omega)^s$, and computes a Pedersen commitment $c^2_{ped} = g_1^{\kappa_2} \cdot \hat{h}^{H(\alpha,\beta,\gamma,\delta)}$, where $\kappa_2 \in_R \mathbb{Z}_q$. P_i sends c^2_{ped} to P_j.
6. P_j sends (R, S, ε) to P_i.
7. P_i verifies that $c' = (g_1^R \cdot g_2^S, h_1^R \cdot h_2^S \cdot G(\varepsilon))$. If no, it aborts. Otherwise, P_i computes $z = s + \varepsilon r$.
8. P_i erases r and s, and stores $(x, \alpha, \beta, \gamma, \delta, \kappa_2, z)$. P_i sends (κ_1, c) to P_j.
9. P_j verifies that $c^1_{ped} = g_1^{\kappa_1} \cdot \hat{h}^{H(c)}$. If yes, it stores $(sid, ssid, P_i, P_j, c, \varepsilon, c^2_{ped})$ and outputs $(\mathsf{receipt}, sid, ssid, P_i, P_j)$. P_j ignores any later commitment messages with the same $(sid, ssid)$ from P_i.

The decommit phase:

1. Upon input $(\mathsf{reveal}, sid, ssid, P_i, P_j)$, P_i sends $(x, \alpha, \beta, \gamma, \delta, \kappa_2, z)$ to P_j.
2. P_j sets $m = G(x, sid, ssid, i, j)$ and outputs $(\mathsf{reveal}, sid, ssid, P_i, P_j, x)$ if and only if

$$c^2_{ped} = g_1^{\kappa_2} \cdot \hat{h}^{H(\alpha,\beta,\gamma,\delta)},\ g_1^z = \alpha \cdot u_1^\varepsilon,\ g_2^z = \beta \cdot u_2^\varepsilon,\ h^z = \gamma \cdot \left(\frac{e}{m}\right)^\varepsilon,\ (cd^\omega)^z = \delta \cdot v^\varepsilon$$

We note one difference between the actual protocol and the intuitive explanation above, regarding the Pedersen commitments. We use these commitments to commit to group elements in $\mathbb{G}$. However, the input of a Pedersen commitment is in $\mathbb{Z}_q$ and not $\mathbb{G}$. One solution to this is to break the elements up into pieces and separately commit to each piece. We use a different solution which is to compute $Com(m) = g^r \cdot \hat{h}^{H(m)}$ where H is a collision-resistant hash function. The commitment is still perfectly hiding and can be opened to any value. The only difference is that the binding property relies now both on the hardness of the discrete log problem (as in the standard case) and on the collision resistance of H. By convention, in Protocol 4, all messages are sent together with $(sid, ssid)$.

Efficiency. The complexity of Protocol 4 is the same as the static version (Protocol 2) plus two additional Pedersen commitment that must be computed and verified. Naively, this costs an additional 8 exponentiations overall. However, again using the multiexponentiation optimization, these can be computed at the effective cost of $5\frac{1}{3}$ exponentiations. Thus, we have an overall cost of $28\frac{2}{3}$ exponentiations.

4.3 Proof of Security

Theorem 2. *Assuming that the DDH assumption holds in the group $\mathbb{G}$, Protocol 4 UC-securely realizes the $\mathcal{F}_{\text{MCOM}}$ functionality in the $\mathcal{F}_{\text{CRS}}$-hybrid model, in the presence of adaptive adversaries with erasures.*

Proof: The proof of security is very similar to the static case, with the addition of how to deal with adaptive corruptions. We remark that we follow the convention where the only part of the commitment message not seen by the ideal adversary is the commitment value [6]. Thus, when an honest P_i sends a message to the $\mathcal{F}_{\text{MCOM}}$ functionality, the adaptive ideal adversary knows what type of message it is and who the intended recipient is.

Due to lack of space in this extended abstract, we sketch the main difference between the proof here, and the proof in the static case. The main observation is that if a committing party is corrupted *before* the commitment stage is finished, then no meaningful information has been given away. This is due to the use of Pedersen commitments and the fact that the simulator can open them to any way it wishes by choosing $\hat{h}$ so that it knows its discrete log. Furthermore, if a committing party is corrupted *after* the commitment phase is finished, then the randomness used to generate the Cramer-Shoup encryption and the Sigma protocol prover messages has already been erased. Thus, all the simulator has to do is to run the Sigma-protocol simulator using the proof statement based on the commitment value obtained, and this will look exactly like an honestly generated commitment.

Due to the above, the simulation and proof in this case of adaptive corruptions is very similar to the case of static corruptions. However, there is one major difference regarding the last step where we must prove the soundness of the proofs provided by corrupted parties. Specifically, we need to claim that a corrupted party can prove an incorrect statement with probability that is at most negligible. In order to prove this, we first replace the dual mode public key with the alternative one, as in the proof of the case of static corruptions. However, this does not yet suffice because the adversary may be able to prove an incorrect claim by breaking the computational binding of the Pedersen commitments (recall that the last message of the proof is decommitted to only after the challenge ε is revealed). Despite this, we use the fact that the ability to decommit to two different values is equivalent to find the discrete log of $\hat{h}$. Specifically, given c_{ped} together with $(\kappa, m) \neq (\kappa', m')$ such that $c_{ped} = g_1^{\kappa} \cdot \hat{h}^m = g_1^{\kappa'} \cdot \hat{h}^{m'}$, it holds that $\hat{h} = g_1^{(\kappa-\kappa')(m'-m)^{-1}}$ and so the discrete log of $\hat{h}$ is $\frac{\kappa-\kappa'}{m'-m}$ which can be efficiently computed.

We therefore prove soundness as follows. Assume that there exists an environment $\mathcal{Z}$, adversary $\mathcal{A}$, and an input z to $\mathcal{Z}$ such that for infinitely many n's, $\mathcal{A}$ succeeds in proving an incorrect statement with non-negligible probability. In this case, $\mathcal{A}$ will succeed in proving an incorrect statement with non-negligible probability also in HYB-GAME^3. (We remark that it is possible to detect this event because we can decrypt the Cramer-Shoup encryption and see what value was actually encrypted.) Now, let $(\mathbb{G}, q, g_1, \hat{h})$ be parameters for a Pedersen commitment. An adversary $\mathcal{A}_{ped}$ attempting to break the commitment scheme receives the parameters and works as follows. It chooses all of the values in the common reference string like $\mathcal{S}$ (based on $(\mathbb{G}, q, g_1)$) and then simulates the HYB-GAME^3 experiment running $\mathcal{Z}$ with input z and $\mathcal{A}$. If $\mathcal{A}$ proves an incorrect statement, then $\mathcal{A}_{ped}$ *rewinds* the entire execution (including $\mathcal{Z}$) until the point that $\mathcal{A}$ sent c^2_{ped}. $\mathcal{A}_{ped}$ then sends a different decommitment (R', S', ε') to a fresh random ε'. Note that since at this point the public key for the dual-mode cryptosystem is the alternative one, and $\mathcal{A}_{ped}$ knows the discrete logs ρ_1, ρ_2 of h_1, h_2, it can efficiently find (R', S', ε') such that $c' = (g_1^R \cdot g_2^S, h_1^R \cdot h_2^S \cdot G(\varepsilon))$ even though c' was originally generated as an encryption of some $\varepsilon \neq \varepsilon'$. (In order to do this, $\mathcal{A}_{ped}$ also needs to know the discrete logs of g_2 and $G(\varepsilon), G(\varepsilon')$ relative to g_1, but these value can be chosen in that way. See the explanation of the concrete dual-mode cryptosystem at the end of Section 3.1 in order to see what equations $\mathcal{A}_{ped}$ needs to solve.) $\mathcal{A}_{ped}$ then continues the execution until the point that $\mathcal{A}$ decommits to the transcript. If it is accepting, then $\mathcal{A}$ must have opened the Pedersen commitment c^2_{ped} differently (because $\mathcal{A}$ can only answer one ε if the statement is incorrect). In this case $\mathcal{A}_{ped}$ has found the discrete log of $\hat{h}$ and halts. Otherwise, $\mathcal{A}_{ped}$ repeatedly rewinds until $\mathcal{A}$ does provide an accepting transcript. This yields an expected polynomial-time adversary $\mathcal{A}_{ped}$; a strict polynomial-time adversary can be derived by just truncating the execution after enough time. We remark that although we are working in the UC framework, $\mathcal{A}_{ped}$ is allowed to rewind in the reduction because this has nothing to do with the simulation, and we are reducing the difference between HYB-GAME^3 and HYBRID to the hardness of finding the discrete log of $\hat{h}$. ■

Acknowledgements

We thank Ran Canetti and Benny Pinkas for helpful discussions.

References

1. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography, ch. 7 (course notes) (2007)
2. Ben-David, A., Nisan, N., Pinkas, B.: FairplayMP: a System for Secure Multi-Party Computation. In: The 15th ACM CCS, pp. 257–266 (2008)
3. Canetti, R.: Universally Composable Security: A New Paradigm for Cryptographic Protocols. In: 42nd FOCS, pp. 136–145 (2001), full version http://eprint.iacr.org/2000/067
4. Canetti, R., Fischlin, M.: Universally Composable Commitments. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 19–40. Springer, Heidelberg (2001)

5. Canetti, R., Krawczyk, H.: Security Analysis of IKE's Signature-Based Key-Exchange Protocol. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 143–161. Springer, Heidelberg (2002)
6. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally Composable Two-Party and Multi-Party Computation. In: 34th STOC, pp. 494–503 (2002), full version http://eprint.iacr.org/2002/140
7. 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)
8. Damgård, I.: On Σ Protocols, http://www.daimi.au.dk/$\sim$ivan/Sigma.pdf
9. Damgard, I., Nielsen, J.: Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 581–596. Springer, Heidelberg (2002)
10. Hazay, C., Lindell, Y.: Efficient Secure Two-Party Protocols – Techniques and Constructions, October 2010. Springer, Heidelberg (2010)
11. Hazay, C., Katz, J., Koo, C.Y., Lindell, Y.: Concurrently-Secure Blind Signatures without Random Oracles or Setup Assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
12. Hazay, C., Nissim, K.: Efficient Set Operations in the Presence of Malicious Adversaries. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 312–331. Springer, Heidelberg (2010); Full version in the Cryptology ePrint Archive, report 2009/594
13. Kelsey, J., Schneier, B., Wagner, D.: Protocol Interactions and the Chosen Protocol Attack. In: Christianson, B., Lomas, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998)
14. 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)
15. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
16. Lindell, Y., Pinkas, B.: Secure Two-Party Computation via Cut-and-Choose Oblivious Transfer. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 329–346. Springer, Heidelberg (2011)
17. Lindell, Y., Pinkas, B., Smart, N.P.: Implementing Two-Party Computation Efficiently with Security Against Malicious Adversaries. In: Ostrovsky, R., De Prisco, R., Visconti, I. (eds.) SCN 2008. LNCS, vol. 5229, pp. 2–20. Springer, Heidelberg (2008)
18. Menezes, A., Van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
19. Moran, T., Moore, T.: The Phish-Market Protocol: Securely Sharing Attack Data between Competitors. In: Sion, R. (ed.) FC 2010. LNCS, vol. 6052, pp. 222–237. Springer, Heidelberg (2010)
20. Osadchy, M., Pinkas, B., Jarrous, A., Moskovich, B.: SCiFI - A System for Secure Face Identification. In: the 31st IEEE Symposium on Security and Privacy, pp. 239–254 (2010)
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. 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)
23. Pinkas, B., Schneider, T., Smart, N.P., Williams, S.C.: Secure Two-Party Computation Is Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 250–267. Springer, Heidelberg (2009)
24. Vanstone, S.: Deployments of Elliptic Curve Cryptography. In: the 9th Workshop on Elliptic Curve Cryptography (ECC) (2005)
25. The Crypto++ Library, http://www.cryptopp.com

Concurrent Composition in the Bounded Quantum Storage Model

Dominique Unruh

Saarland University

Abstract. We define the BQS-UC model, a variant of the UC model, that deals with protocols in the bounded quantum storage model. We present a statistically secure commitment protocol in the BQS-UC model that composes concurrently with other protocols and an (a-priori) polynomially-bounded number of instances of itself. Our protocol has an efficient simulator which is important if one wishes to compose our protocol with protocols that are only computationally secure. Combining our result with prior results, we get a statistically BQS-UC secure constant-round protocol for general two-party computation without the need for any setup assumption.

Keywords: Bounded quantum storage, composability, two-party computation.

1 Introduction

Since the inception of quantum key distribution by Bennett and Brassard [2], it has been known that quantum communication permits to achieve protocol tasks that are impossible given only a classical channel. For example, a quantum key distribution scheme [2] permits to agree on a secret key that is statistically secret, using only an authenticated but not secret channel. (By statistical security we mean security against computationally unbounded adversaries, also known as information-theoretical security.) In contrast, when using only classical communication, it is easy to see that such a secret key can always be extracted by a computationally sufficiently powerful adversary. In light of this result, one might hope that quantum cryptography allows to circumvent other classical impossibility results, possibly even allowing for statistically secure multi-party computation protocols. Yet, Mayers [13] showed that also in the quantum setting, even statistically secure commitment schemes are impossible, let alone general multi-party computation. This is unfortunate, because from commitments one can build OT (Bennett, Brassard, Crépeau, and Skubiszewska [3]), and from OT general multi-party computation (Kilian [11]). A way to work around this impossibility was found by Damgård, Fehr, Salvail, and Schaffner [6]. They showed that if we assume that the quantum memory available to the adversary is bounded (we speak of *bounded quantum storage* (BQS)), we can construct statistically secure commitment and OT schemes. Although such a result is not truly unconditional, it avoids hard-to-justify complexity-theoretic assumptions. Also, it achieves long-term security: even if the adversary can surpass the memory bound after the protocol execution, this will not allow him to retroactively break the protocol.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 467–486, 2011.

Yet, we still have not reached the goal of statistically secure multi-party computation. Although we have protocols for commitment and OT, we cannot simply plug them into the protocols by Bennett et al. [3] and by Kilian [11]. The reason is that it is not clear under which circumstances protocols in the BQS model may be composed. For example, Dziembowski and Maurer [7] constructed a protocol that is secure in the classical bounded storage model, but that looses security when composed with a computationally secure protocol. To overcome this remaining difficulty, works by Wehner and Wullschleger [17] and by Fehr and Schaffner [8] give security definitions in the BQS model that enable secure sequential composition. Both works also present secure OT protocols in their respective settings. Based on these, we can construct secure multi-party computation protocols in the BQS model. There are, however, a few limitations. First, since only sequential composition is supported, all instances of the OT protocol used by the multi-party computation need to be executed one after another, leading to a high round-complexity. Second, interactive functionalities such as a commitment are difficult to use: the restriction to sequential composability requires that we have to commit and immediately open a commitment before being allowed to execute the next commitment. Third, the security proof of their OT protocols uses a computationally unlimited simulator. As discussed in [15], a protocol with an unlimited simulator cannot be composed with a computationally secure protocol. Fourth, since we have no concurrent composability, it is not clear what happens if the protocols are executed in an environment where we do not have total control about which protocols are executed at what time.

To overcome the limitations of sequential composition *in the classical setting*, Canetti [4] introduced the Universal Composability (UC) model. In this model, protocols can be arbitrarily composed, even concurrently with other protocols and with copies of themselves. The UC model has been adapted to the quantum setting by Ben-Or and Mayers [1] and by Unruh [14,15]. In light of the success of the UC model, it seems natural to combine the ideas of the UC model with those of the BQS model in order to allow for concurrent composition.

1.1 Our Contribution

We define the notion of BQS-UC-security, which is an extension of quantum-UC-security [15]. We have composability in the following sense: If π is a secure realization of a functionality $\mathcal{F}$, and $\sigma^{\mathcal{F}}$ securely realizes $\mathcal{G}$ by using one instance of $\mathcal{F}$, then σ^{π}, the result of replacing $\mathcal{F}$ by π, still securely realizes $\mathcal{G}$. In contrast to quantum-UC-security, however, BQS-UC-security does not allow for concurrent self-composition: if π is secure, this does not automatically imply that two concurrent instances of π are secure[1].

[1] The reader may wonder how it can be that σ and π may compose in general while π and π do not. Might not σ and π be the same protocol? The reason lies in the exact conditions of the composition theorem: In order to compose, σ needs to be secure against adversaries with a higher quantum memory bound than π tolerates. Thus σ and π cannot be the same protocol.

In order to get protocols that even self-compose concurrently, we design a commitment scheme π_{COM} such that n concurrent instances of π_{COM} securely realize n instances of the commitment functionality in the presence of a-memory bounded adversaries. Here a and n are arbitrary (polynomially-bounded), but the protocol depends on a and n.

The challenging part in the construction of π_{COM} is that BQS-UC-security requires the following: There must be an *efficient* simulator (which is allowed to have more quantum memory than the adversary) that can extract the committed value (extractability) or change it after the commit phase (equivocality). Prior constructions of commitment schemes in the BQS model required computationally unbounded simulators. Also, the fact that we directly analyze the concurrent composition of several instances of π_{COM} requires care: In the proof, we have hybrid networks in which instances of both π_{COM} and of the simulator occur. Since the simulator uses more quantum memory than π_{COM} tolerates, one needs to ensure that the simulator cannot be (mis)used by the adversary to break the commitment.

Finally, using the composition theorem and π_{COM}, for any two-party functionality $\mathcal{G}$, we get a statistically secure protocol π realizing $\mathcal{G}$ in the BQS model[2]. The protocol is secure even when running n concurrent instances of the protocol. (Again, this holds for any n and any memory bound, but the protocol depends on n and the memory-bound.) The protocol is constant-round. It does not use any quantum memory or quantum computation and thus is in the reach of today's technology.

A full version of this paper with complete proofs and details can be found at [16].

2 Bounded Quantum Storage UC

2.1 The BQS-UC Model

To understand the definition of BQS-UC-security, we first have to understand the idea underlying UC security. In the UC model, a protocol π emulates (realizes, implements, is as secure as) another protocol ρ if any attack on π can also happen on ρ. Thus, if ρ is secure by definition (e.g., because it contains only one trusted machine, a so-called ideal functionality), π must also be secure. To formalize this, we introduce the concept of an adversary and a simulator. We require that for any adversary attacking π (real model), there is a simulator attacking ρ (ideal model) such that the real and the ideal model are indistinguishable. To define indistinguishability, we introduce another machine, the environment. Its task is to try and distinguish between the real and the ideal model. The environment provides the inputs to the protocol parties, gets their outputs, and may talk to the adversary/simulator. We then get the following definition: π UC-emulates ρ

[2] We are restricted to two-party functionalities because our construction uses a subprotocol by Wolf and Wullschleger [18,19] to reverse the direction of an OT; this protocol only makes sense in a two-party setting.

if for any adversary Adv there is a simulator Sim such that for all environments $\mathcal{Z}$, the probability that $\mathcal{Z}$ outputs 1 is approximately the same in the networks $\pi, \mathrm{Adv}, \mathcal{Z}$ and $\rho, \mathrm{Sim}, \mathcal{Z}$.

To translate this to the quantum setting, we only need to change the machine model to allow for quantum machines instead of classical machines. Given networks S, S' of quantum machines with $\mathcal{Z} \in S, S'$, we say that S and S' are *ε-close* if $|P - P'| \leq \varepsilon$ where P is the probability that $\mathcal{Z}$ outputs 1 in an execution of S, and P' is defined analogously. Networks are *negligible-close* if ε is negligible, and *perfectly close* if $\varepsilon = 0$. We call a machine M *a-memory bounded* if it keeps at most a qubits of quantum memory between activations. (An activation is the computation performed by a machine between receiving a message and sending the immediate response.) We do not impose any limitations on the computation-time or memory-use during a single activation of M. We write $\mathrm{QM}(M)$ for the memory bound of M (i.e., $\mathrm{QM}(M)$ is the smallest a such that M is a-memory bounded). A protocol π is a network not containing adversary, simulator, or environment. The set of corruptible protocol parties is denoted $\mathit{parties}_\pi$. Given $C \subseteq \mathit{parties}_\pi$, we denote by π^C the protocol where all parties in C have been replaced by *corruption parties* which are controlled by the adversary. For details on the machine and the network model, we refer to the full version [16] or to [15].

We can now formulate BQS-UC-security. Intuitively, a protocol is BQS-UC-secure if it is UC-secure for memory-bounded adversaries. To formulate this, we need to explicitly parametrize the definition over a memory bound a. Then we require that the total quantum memory used by environment and adversary is bounded by a. The reason why we include the environment's memory is that the latter can be involved in the actual attack: If only the adversary's memory was bounded, the adversary could use the environment as an external storage to perform the attack (see also our discussion on page 471)[3].

It remains to decide whether the simulator should be memory bounded. If we allow the simulator to be unbounded, composition becomes difficult: In some cases, the simulator of one protocol plays the role of the adversary of a second protocol. Thus, if simulators where not memory bounded, the second protocol would have to be secure against unbounded adversaries. However, if we require the simulator to be a-memory bounded, we will not be able to construct non-trivial protocols: In order to perform a simulation, the simulator needs to have some advantage over an honest protocol participant (in the computational UC setting, e.g., this is usually the knowledge of some trapdoor). In our setting, the advantage of the simulator will be that he has more quantum memory than the adversary. Thus we introduce a second parameter s which specifies the amount of quantum memory the simulator may use for the simulation. More precisely, we allow the simulator to use $s + \mathrm{QM}(\mathrm{Adv})$ qubits because the simulator will usually internally simulate the adversary Adv as a black-box and therefore have to additionally reserve sufficient quantum memory to store the adversary's state.

[3] This is captured more formally by the completeness of the so-called dummy-adversary (see the full version [16]), which shows that one can even shift the complete attack into the environment.

Definition 1 (BQS-UC-security). *Fix protocols π and ρ. Let $a, s \in \mathbb{N}_0 \cup \{\infty\}$ (possibly depending on the security parameter). We say π (a,s)-*BQS-UC-emulates[4] *ρ iff for every set $C \subseteq \mathit{parties}_\pi$ and for every adversary* Adv *there is a simulator* Sim *with* $\mathrm{QM}(\mathrm{Sim}) \leq s + \mathrm{QM}(\mathrm{Adv})$ *such that for every environment $\mathcal{Z}$ with $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$, the networks $\pi^C \cup \{\mathrm{Adv}, \mathcal{Z}\}$ (called the real model) and $\rho^C \cup \{\mathrm{Sim}, \mathcal{Z}\}$ (called the ideal model) are negligible-close. We furthermore require that if* Adv *is quantum-polynomial-time, so is* Sim.

In most cases, the behavior of the ideal protocol ρ is described by a single machine $\mathcal{F}$, the so-called ideal functionality. We can think of this functionality as a trusted third party that perfectly implements the desired protocol behavior. In order to apply Definition 1 to ideal functionalities (e.g., π BQS-UC-emulates $\mathcal{F}$) we have to be able to consider an ideal functionality as a protocol. Following [4,15], we do this by introducing dummy-parties. That is, the protocol $\mathcal{F}$ consists of the functionality $\mathcal{F}$ together with a *dummy-party* $\tilde{A}$ for every party A. The dummy-parties just forward inputs/outputs between the functionality $\mathcal{F}$ and the environment. They can, however, be corrupted by the adversary/simulator. This allows the adversary/simulator to control the inputs/outputs of that party. When we write π BQS-UC-emulates $\mathcal{F}$, we always assume the presence of dummy-parties in the ideal model. For details, we refer to the full version [16] or to [15].

On the Memory Bound of the Environment. In Definition 1, we impose the memory bound on both the adversary and the environment. In this, we differ from the modeling by Wehner and Wullschleger [17]. In their definition, the environment (which is implicit in the definition of the indistinguishability $\equiv_\varepsilon$ of quantum channels) provides the input state to protocol and adversary, then gets the outputs of protocol and adversary, and finally the environment has to guess whether it interacted with the real or the ideal model. During the interaction of the protocol, the environment is not allowed to communicate with any other machine. Between its two activations, the environment is allowed to keep an arbitrarily large quantum state[5]. The interesting point here is that, in contrast to our Definition 1, Wehner and Wullschleger do not impose the memory bound on the environment, only on the adversary. They motivate unlimited environments by pointing out that it is more realistic to assume that a particular memory bound (say, 100 qubits) applies to a particular adversary (e.g., a smart card) than to the whole environment (i.e., all computers world-wide). We believe, however, that this reasoning has to be applied with care: Only when we have

[4] Since we only consider statistical security in this work, we omit the qualifier "statistical". Similarly, when we speak about classical-UC-security and quantum-UC-security, we mean the statistical variant of that notion.

[5] Note that strictly speaking, the formalism of [17, full version] does not model an environment with quantum memory: For quantum channels Λ, Λ' they define $\Lambda \equiv_\varepsilon \Lambda'$ iff for all quantum states ρ, the trace distance between $\Lambda(\rho)$ and $\Lambda'(\rho)$ is at most ε. To model environments with quantum memory, we should instead require that for all Hilbert spaces $\mathcal{H}$ and all quantum states ρ, the trace distance between $(\Lambda \otimes id_{\mathcal{H}})(\rho)$ and $(\Lambda' \otimes id_{\mathcal{H}})(\rho)$ is at most ε. We believe that the latter was the intended meaning of $\equiv_\varepsilon$.

the guarantee that the adversary (e.g., the smart card) cannot communicate with any other machines can we assume a smart card is limited to 100 qubits. Otherwise, we have to assume that the smart card effectively has (in the worst case) access to all the quantum memory of the environment. Thus, except in very specific cases, the memory bound we assume needs to be large enough to encompass the environment's memory as a whole. Thus, the bound we assume not to be surpassed by the adversary's memory needs to be large enough that it makes sense to assume that the environment does not surpass this bound either. But in this case, we can safely assume in Definition 1 that the environment is restricted by that memory bound.

We stress that even if our environment is memory bounded, we do take into account the fact that an environment can have a quantum state that is entangled with that of the adversary; we just limit this quantum state to the memory bound.

We get, however, an interesting variant of our model if we follow the approach of Wehner and Wullschleger as follows. We call a machine a-$\diamond$-memory-bounded[6] if its state between activations consists of two registers A and B. The register A contains at most a qubits, and register B is unlimited but is only accessed in the first and the last activation of B. We denote by $\mathrm{QM}_\diamond(\mathcal{Z})$ the $\diamond$-memory bound of $\mathcal{Z}$. We define (a,s)-$\diamond$-BQS-UC-emulation like (a,s)-BQS-UC-emulation (Definition 1), except that we use $\mathrm{QM}_\diamond(\mathcal{Z})$ instead of $\mathrm{QM}(\mathcal{Z})$. (But we still use $\mathrm{QM}(\mathrm{Adv})$ and $\mathrm{QM}(\mathrm{Sim})$.) We stress that our techniques also work for this definition. All results of this section still hold with essentially unmodified proofs (except that we always have to refer to the $\diamond$-memory bound of the environment instead memory bound). The results from Section 3 (BQS-UC commitments) are based on the existence of certain commitment schemes that are hiding with respect to memory bounded adversaries. We use the commitment scheme from [12] (see Theorem 6). To extend the results from Section 3 to $\diamond$-BQS-UC, we need schemes that are hiding with respect to $\diamond$-memory bounded adversaries instead. Besides that, the proofs of the results in Section 3 stay essentially the same (except for using $\diamond$-memory bounds instead of memory bounds).

2.2 Composition

For some protocol σ, and some protocol π, by σ^π we denote the protocol where σ invokes (up to polynomially many) instances of π. That is, in σ^π the machines from σ and from π run together in one network, and the machines from σ access the inputs and outputs of π. (That is, σ plays the role of the environment from the point of view of π. In particular, $\mathcal{Z}$ then talks only to σ and not to the subprotocol π directly.) A typical situation would be that $\sigma^\mathcal{F}$ is some protocol that makes use of some ideal functionality $\mathcal{F}$, say a commitment functionality, and then σ^π would be the protocol resulting from implementing that functionality with some protocol π, say a commitment protocol. One would hope that such an implementation results in a secure protocol σ^π. That is, we hope that if π

[6] We use the symbol $\diamond$ because $\diamond$-memory-bounded environments essentially model indistinguishability with respect to the so-called $\diamond$-norm.

BQS-UC-emulates $\mathcal{F}$ and $\sigma^{\mathcal{F}}$ BQS-UC-emulates $\mathcal{G}$, then σ^{π} BQS-UC-emulates $\mathcal{G}$. Fortunately, this is the case, as long as we pick the memory bounds in the right way:

Theorem 2 (Composition Theorem). *Let π and σ be quantum-polynomial-time protocols and $\mathcal{F}$ and $\mathcal{G}$ be quantum-polynomial-time functionalities. Assume that σ invokes at most one subprotocol instance. Assume that π (a, s)-BQS-UC-emulates $\mathcal{F}$ and that $\sigma^{\mathcal{F}}$ $(a - \mathrm{QM}(\sigma) + s, s')$-BQS-UC-emulates $\mathcal{G}$. Then σ^{π} $(a - \mathrm{QM}(\sigma), s + s')$-BQS-UC-emulates $\mathcal{G}$.*

The proof of this theorem is very similar to that in [15], except that we have to keep track of the quantum memory used by various machines constructed in the proof.

Notice that in this composition theorem, the outer protocol σ is only allowed to invoke one instance of the subprotocol π. This stands in contrast to the universal composition theorem for classical-UC [4] and for quantum-UC [15] where any polynomially-bounded number of concurrent instances of π is allowed. In fact, this is not just a limitation of our proof technique[7]. For example, assume a protocol $\pi_{\mathrm{COM}}^{A\to B}$ that (a, s)-BQS-UC-emulates the commitment functionality $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ with sender A and recipient B. Assume further that $\pi_{\mathrm{COM}}^{A\to B}$ does not use any functionalities as setup. As we will see later, such a protocol exists. Now let $\pi_{\mathrm{COM}}^{B\to A}$ be the protocol that results from exchanging the roles of A and B. Then $\pi_{\mathrm{COM}}^{B\to A}$ (a, s)-BQS-UC-emulates $\mathcal{F}_{\mathrm{COM}}^{B\to A}$. Consider the concurrent composition of $\pi_{\mathrm{COM}}^{A\to B}$ and $\pi_{\mathrm{COM}}^{B\to A}$. In this protocol, a corrupted Bob may reroute all messages between the Alice in the first protocol and Alice in the second protocol. Thus, if Alice commits to a random value v in the first protocol, Bob commits to the same value v in the second protocol without knowing it. It is easy to see that in a concurrent composition of $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ and $\mathcal{F}_{\mathrm{COM}}^{B\to A}$, this is not possible. Thus the composition of $\pi_{\mathrm{COM}}^{A\to B}$ and $\pi_{\mathrm{COM}}^{B\to A}$ does not (a', s')-BQS-UC-emulate the composition of $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ and $\mathcal{F}_{\mathrm{COM}}^{B\to A}$ (for any parameters a', s'). To convert this into an example of a protocol that does not even compose with itself, just consider the protocol $\pi_{\mathrm{COM}}^{A\leftrightarrow B}$ in which Bob may choose whether $\mathcal{F}_{\mathrm{COM}}^{A\to B}$ or $\mathcal{F}_{\mathrm{COM}}^{B\to A}$ should be executed. It might be possible to make $\pi_{\mathrm{COM}}^{A\leftrightarrow B}$ self-composable by adding suitable tags inside the messages, but the definition of BQS-UC-security does not enforce this.

Although BQS-UC-security does not guarantee for concurrent self-composability, individual protocols may have this property. In order to formulate this, we introduce the concept of the multi-session variant of a protocol. Given a protocol π and a polynomially-bounded n, we define π^n to be the protocol that executes n instances of π concurrently.

[7] In the proof, the difficulty arises from a hybrid argument where the protocol π is executed together in one network with the protocol ρ and the corresponding simulator. Since the simulator may use more quantum memory than π is resistant against, we cannot guarantee security of π in this hybrid setting and the proof cannot proceed.

Then, from Theorem 2, we immediately get the following corollary:

Corollary 3. *Let π and σ be quantum-polynomial-time protocols and $\mathcal{F}$ and $\mathcal{G}$ be quantum-polynomial-time functionalities. Let $n, m \geq 0$ be integers (depending on the security parameter). Assume that σ invokes at most m subprotocol instances. Assume that π^{nm} (a, s)-BQS-UC-emulates $\mathcal{F}^{nm}$ and that $(\sigma^{\mathcal{F}})^n$ $(a - n\mathrm{QM}(\sigma) + s, s')$-BQS-UC-emulates $\mathcal{G}^n$. Then $(\sigma^{\pi})^n$ $(a - n\mathrm{QM}(\sigma), s + s')$-BQS-UC-emulates $\mathcal{G}^n$.*

3 Commitments

3.1 Extractable Commitments

In this section, we present the notion of online-extractable commitments in the BQS model. These will be used as a building block for constructing BQS-UC commitments in the next section.

Definition 4 ((ε, a)-BQS-hiding). *Given a commitment protocol π with sender Alice and recipient Bob, and an adversary B' corrupting Bob, we denote with $\langle A(m), B'\rangle_{B'}$ the output of B' in an interaction between Alice and B' where Alice commits to m.*

*We call π (ε, a)-*BQS-hiding *iff for all a-memory bounded B' and all $m_1, m_2 \in M$, we have that $\left|\Pr[\langle A(m_1), B'\rangle_{B'} = 1] - \Pr[\langle A(m_2), B'\rangle_{B'} = 1]\right| \leq \varepsilon$. Here M is the message space of the commitment scheme.*

Instead of the binding property, we will need a stronger property: online-extractability. This property guarantees that there is a machine (the *extractor*) that, when running as the recipient of the commit protocol, is able to output the committed value V already after the commit phase. This extractor should be indistinguishable from an honest recipient. Note that this does not contradict (ε, a)-BQS-hiding since we allow the extractor's quantum memory to contain more than a qubits. For our purposes, we will only need a definition of online-extractability that does not impose a memory bound on the adversary. We do, however, make the memory bound s of the extractor explicit.

Definition 5 ((ε, s)-online-extractable). *Given a commitment protocol π with sender Alice and recipient Bob, an* extractor *is a machine B_S that, after the commit phase, gives an output V' and then executes the (honest) code of Bob for the open phase and outputs a value V (the accepted value). (In particular, B_S needs to provide an initial state for the program of the open phase of Bob that matches the interaction so far.) We write $V = \perp$ if the open phase fails.*

For an adversary A', we denote with $\langle A', B\rangle_{A'}$ ($\langle A', B_S\rangle_{A'}$) the output of A' in an interaction between A' and Bob (B_S) where A' is given V after Bob (B_S) terminates.

*We call π (ε, s)-*online-extractable *iff there exists an s-memory bounded quantum-polynomial-time extractor B_S such that for all adversaries A', we have that $\left|\Pr[\langle A', B\rangle_{A'} = 1] - \Pr[\langle A', B_S\rangle_{A'} = 1]\right| \leq \varepsilon$ and in an interaction of A' and B_S, we have $\Pr[V \notin \{V', \perp\}] \leq \varepsilon$.*

Theorem 6 (Online-extractable commitments). *For any polynomially-bounded integers a and ℓ, there is a constant-round 0-memory bounded (ε, a)-BQS-hiding (ε, s)-online-extractable commitment scheme π for some exponentially-small ε and some polynomially-bounded s. The message space of π is $M = \{0,1\}^\ell$.*

A protocol with the properties from Theorem 6 was constructed in [12]. They did not, however, show that it is online-extractable. In the full version [16] we show how their proof of the binding property can be extended to online-extractability.

3.2 BQS-UC Commitments

In this section, we present a commitment scheme π_{COM} that is BQS-UC-secure for memory bound a and for n concurrent instances of π_{COM}. The parameters a and n can be arbitrary, but π_{COM} depends on them. To state our result, we first define the ideal functionality for commitments.

Definition 7 (Commitment). *Let A and B be two parties. The functionality $\mathcal{F}_{\text{COM}}^{A\to B,\ell}$ behaves as follows: Upon (the first) input $(\texttt{commit}, x)$ with $x \in \{0,1\}^{\ell(k)}$ from A, store x and send $\texttt{committed}$ to B. Upon (the first) input $\texttt{open}$ from A send $(\texttt{open}, x)$ to B (unless x is still undefined). All communication/input/output is classical. We call A the sender and B the recipient.*

Note that this definition also defined the behavior of the functionality in the case where A or B is corrupted. In this case, the adversary (or simulator) is allowed to send/receive the inputs/outputs in the name of A or B, respectively. For example, if A is corrupted, the adversary can decide when to commit to what message and when to open.

Intuition. The protocol π_{COM} is depicted in Figure 1. Before we prove its security, we first explain the underlying intuition. In order to prove the BQS-UC-security of π_{COM}, it is necessary to construct a simulator (that may use more quantum memory than the adversary) that achieves the following: When being in the role of the recipient, the simulator is able to extract the commitment after the commit phase. When being in the role of the sender, the simulator should be able to open the commitment to any value of his choosing (equivocality). The first requirement can easily be achieved by using the online-extractable commitment scheme from Theorem 6. That scheme, however, is not equivocal. In order to make our protocol equivocal, we intentionally weaken the binding property of the commitment. Instead of committing to a single value v, the sender commits using a commitment scheme C_2 to random values $\underline{R} := R_1, \ldots, R_m$. Then he sends $v \oplus F(\underline{R})$ with F being a universal hash function and sends the syndrome σ of $\underline{R}$ with respect to a suitable linear code. In the open phase, the sender does not open all commitments R_i, but instead just sends $\underline{R}$ to the recipient. The recipient chooses a test set T, and the sender opens R_i for $i \in T$. The modified scheme is still binding: Assume the sender wishes to be able to open the commitment with two different values. Then he has to find values $\underline{R}' \neq \underline{R}$ that both pass the recipients checks in the open phase. If $\underline{R}'$ differs from $\underline{R}$ in many

Parameters: Integers ℓ (the length of the committed value), m, $c < m$, b, d, $\kappa < m$. A b-block (m, κ, d)-linear code[8] where $\mathbf{S}(\omega) \in \{0,1\}^{(m-\kappa)b}$ denotes the syndrome of a codeword $\omega \in \{0,1\}^{mb}$. A family $\mathbf{F}$ of strongly universal hash functions $F : \{0,1\}^{mb} \to \{0,1\}^{\ell}$. All parameters may depend on the security parameter k.
Subprotocols: A commitment scheme C_1 with sender Bob, and a commitment scheme C_2 with sender Alice, both 0-memory bounded (not using quantum memory)[9].
Parties: The sender Alice A and the recipient Bob B.
Inputs: In the commit phase, Alice gets $(\texttt{commit}, v)$ with $v \in \{0,1\}^{\ell}$. In the open phase, Alice gets `open`. Bob gets no inputs.
Commit phase:
C1. Bob picks a random $T \subseteq \{1, \ldots, m\}$ with $\#T = c$. Then Bob commits to T using C_1. (We assume some encoding of sets T that does not allow to encode sets with $\#T \neq c$.)
C2. Alice picks $R_1, \ldots, R_m \in \{0,1\}^b$. For each i, Alice commits to R_i using C_2. (The commitments may be performed concurrently.)
C3. Alice picks a hash function $F \leftarrow \mathbf{F}$, computes $p := v \oplus F(R_1 \| \ldots \| R_m)$, computes the syndrome $\sigma := \mathbf{S}(R_1 \| \ldots \| R_m)$, and sends (F, σ, p) to Bob. (This may be done concurrently with the commitments to R_i.)
C4. Bob outputs `committed`.
Open phase:
O1. Alice sends $R_1 \| \ldots \| R_m$ to Bob.
O2. Bob opens T using C_1.
O3. For each $i \in T$, Alice opens R_i using C_2. (The open phases may be executed concurrently.)
O4. Bob checks that the values R_i sent by Alice match the values R_i opened by Alice for all $i \in T$, and that $\sigma = \mathbf{S}(R_1 \| \ldots \| R_m)$.
O5. Bob computes $v := p \oplus F(R_1 \| \ldots \| R_m)$ and outputs $(\texttt{open}, v)$. (I.e., Bob accepts the opened value v.)

Fig. 1. Our commitment protocol π_{COM}

blocks R_i, with high probability the verifier will require that one of these R_i is opened and the sender will be caught. If $\underline{R}'$ and $\underline{R}$ differs in only few blocks, then $\underline{R} - \underline{R}'$ has a low Hamming weight and is not in the code. Hence the syndrome of $\underline{R} - \underline{R}'$ is not zero, and, since the code is linear, the syndromes of $\underline{R}'$ and $\underline{R}$ cannot both equal σ. Thus the sender is caught, too. Furthermore, our scheme is online-extractable if C_2 is online-extractable since the simulator can extract the committed values $\underline{R}$. However, we have not yet achieved the equivocality. In order to open the commitment to a different value, the sender needs to know T before sending $\underline{R}'$. To achieve this, the recipient commits to T before the commit phase (using an online-extractable commitment scheme C_2). A simulator wishing to change the value of the commitment simply extracts T. Then he knows which R_i can be changed without being detected and can thus change $F(R_1, \ldots, R_m)$ to any value he wishes.

[8] That is, a code where the code words consist of m blocks of b bit, that contains $2^{b\kappa}$ codewords, and where every non-zero codeword contains at least d nonzero blocks.
[9] Note that this does not refer to the memory bound of the adversary. We only state that honest Alice and Bob do not need to use quantum memory in C_1 and C_2.

Difficulties with Concurrent Composition. The main difficulty in showing the BQC-UC-security of π_{COM} lies in coping with the fact that several (say n) instances of π_{COM} might run concurrently. Consider for example the case that Alice is corrupted. In this case, the adversary may produce the C_2-commitments to $R_1, \ldots, R_m$ in n instances of Alice. The simulator needs to run the nm extractors to extract these commitments. Each of these extractors needs some quantum memory s_2. Thus our simulator needs nms_2 bits of quantum memory. On the other hand, we need to make sure that the C_1-commitments to T, produced by the simulator, are hiding. C_1 needs to be hiding against a_1-bounded adversaries with $a_1 > nms_2 \geq s_2$ (because we cannot be sure that the memory used by the simulator is not misused by the adversary). But then the extractor for C_1 needs to use $s_1 > a_1$ qubits; otherwise the adversary could run the extractor to break the protocol. Similarly, we can see that when Bob is corrupted, C_2 needs to be hiding against a_2-memory bounded adversaries with $a_2 > ns_1 \geq s_1$, and $s_2 > a_2$. Thus we need $a_1 > s_2 > a_2 > s_1 > a_1$ which is impossible.

Solving the Difficulties. The way out is to carefully track the memory used by the simulators; it turns out that in the proof of security against corrupted Alice, we can make sure that the adversary is not able to "misuse" the memory of the simulator. When Alice is corrupted, we need to construct a simulator that extracts the values v of n concurrent commitments produced by Alice, while being indistinguishable from an execution of the honest recipient Bob. More precisely, we show that the simulator is indistinguishable if C_1 is a_1-BQS-hiding and C_2 is s_2-online-extractable and environment and adversary are a_1-memory-bounded. We do not require that $a_1 > s_2$, thus breaking the above-mentioned circularity in the choices of a_1, s_1, a_2, s_2.

Let B^* be defined like the honest Bob, except that instead of honestly running the recipient's code for C_2, B^* runs the extractor B_S for C_2 to extract the committed values $\underline{R}$ in the C_2-commitments. From $\underline{R}$, B^* computes a guess v' for the committed value v. B^* does not, however, use this guess at any point.

First, note that B^* is indistinguishable from honest Bob: This follows from the fact that the extractor for C_2 is indistinguishable from the recipient for C_2. Furthermore, as discussed in the section "Intuition" above, extracting v will be successful as long as Alice does not learn anything about T, i.e., as long as C_1 is hiding. But C_1 is only a_1-BQS-hiding. And B^* uses ms_2 qubits to run the extractors for C_2, so the total memory used in the network is $a_1 + ms_2$ which is beyond the memory bound tolerated by C_1. Fortunately, however, honest Bob runs the recipient of C_2 after the end of the commit phase and before the beginning of the open phase of C_1. Thus, from the point of view of C_1, the extractors are executed within a single atomic computation. And we defined BQS-hiding to hold even if the adversary uses unlimited memory within a single activation. Thus the ms_2 qubits used by the extractors do not break the hiding property, and we get that B^* guesses the right v' with overwhelming probability.

This argument does, however, only work when a single instance of B^* is executed. If several instances of B^* are executed, one instance may run the commit or open phase of C_1 concurrently with another instance's extractors.

Two show that n concurrent instances of B^* extract successfully, we use the following argument: For each j, we have that if only the j-th Bob instance is replaced by B^*, then B^* extracts correctly. Furthermore, Bob and B^* are indistinguishable, thus if we replace all the other instances of Bob by B^*, the j-th instance still extracts correctly. Thus, for any j, if there are n instances of B^*, then the j-th instance extracts correctly. Thus all instances of B^* extract correctly. And the instances of B^* are indistinguishable from the instances of Bob.

Finally, we can construct a simulator that runs B^* instead of Bob and uses the value v' extracted by B^* as input to the commitment functionality. Since $v = v'$ with overwhelming probability, this simulator is successful.

Thus we have shown that a_1 can be chosen independently of s_2. This allows to break the circularity in the choices of a_1, s_1, a_2, s_2: We first start with an arbitrary $a = a_1$. Then we pick an arbitrary a_1-BQS-hiding and s_1-online-extractable C_1, and then an arbitrary $a_2 := a + ns_1$-BQS-hiding and s_2-online-extractable commitment C_2. For the case of corrupted Bob, we then construct a simulator that uses ns_1 qubits and is secure against $a = a_2 - ns_1$-memory bounded environments and adversaries. And for the case of corrupted Alice, using the argument above, we get a simulator that uses nms_2 qubits and is secure against $a = a_1$-bounded environments and adversaries.

The Analysis. We proceed with the formal analysis of π_{COM}. We first consider the case where the recipient is corrupted.

Lemma 8. *Assume that ε, δ are negligible, n, c are polynomially-bounded, and $2\kappa b - mb - 2cb - \ell$ is superlogarithmic (in the security parameter k). Assume that C_1 is (ε, s_1)-online-extractable and C_2 is $(\delta, a + ns_1)$-BQS-hiding. Assume that $\mathbf{F}$ is a family of affine strongly universal hash functions.*

Then π^n_{COM} (a, ns_1)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\mathrm{COM}})^n$ for corrupted recipient B.

Proof. First, we describe the structure of the real and ideal model in the case that the party B (Bob) is corrupted:

In the real model, we have the environment $\mathcal{Z}$, the adversary Adv, the honest party A (Alice), the corruption party B^C. The adversary controls the corruption party B^C, so effectively he controls the communication between Alice and Bob. The environment provides Alice's inputs $(\texttt{commit}, v)$ and $\texttt{open}$. See Figure 2 (a).

In the ideal model, we have the environment $\mathcal{Z}$, the simulator Sim (to be defined below), the dummy-party $\tilde{A}$, the corruption party B^C, and the commitment functionality $\mathcal{F}_{\mathrm{COM}}$. The inputs $(\texttt{commit}, v)$ and $\texttt{open}$ of $\mathcal{F}_{\mathrm{COM}}$ are provided by the dummy-party $\tilde{B}$ and thus effectively by the environment $\mathcal{Z}$. The simulator Sim controls the corruption party B^C and hence gets the outputs $\texttt{committed}$ and $(\texttt{open}, v)$ of $\mathcal{F}_{\mathrm{COM}}$. See Figure 2 (b).

Fix an adversary Adv. To show Lemma 8, we need to find a simulator Sim with $\mathrm{QM}(\mathrm{Sim}) \leq ns_1$ such that, for any environment $\mathcal{Z}$ with $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$, the real model and the ideal model are negligible-close. This simulator is described in Figure 3. We use the abbreviations $\underline{R} := R_1 \| \dots \| R_m$ and $\underline{R}' := R'_1 \| \dots \| R'_m$.

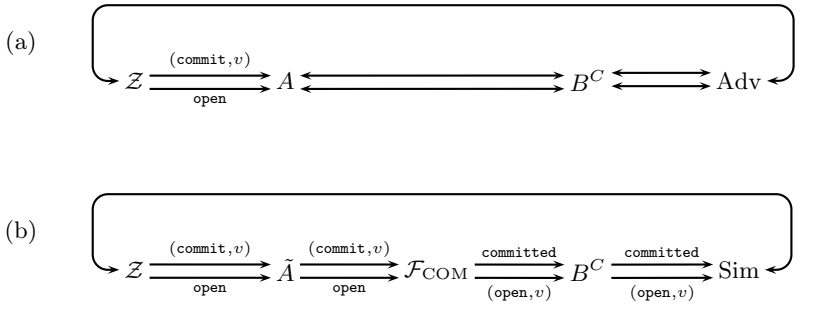


Fig. 2. Networks occurring in the proof of Lemma 8

To show that the real and the ideal model are negligible-close, we start with the real model, and change the machines in the real model step-by-step until we end up with the ideal model. In each step, we show that the network before and after that step are negligible-close.

Game 1. We change the machine A as follows: Instead of executing the program of the honest recipient of C_1, A executes the extractor A_S. ◇

Let T' denote the extracted value. The modified A does not use T'. Since there are up to n copies of A, and since C_1 is (ε, s_1)-online-extractable, the real model and Game 1 are $n\varepsilon$-close.

Game 2. We change the machine A to abort if the opening of T succeeds and reveals a value $T \neq T'$. ◇

Since C_1 is (ε, s_1)-online-extractable, in each instance of A, this happens with probability at most ε, thus Game 1 and Game 2 are $n\varepsilon$-close.

Notice that the only machines that use quantum memory in Game 2 are $\mathcal{Z}$, Adv, and n copies of A_S. Since A_S is s_1-memory bounded, and $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$ we have that the total amount of quantum memory used in Game 2 is bounded by $a + ns_1$.

Game 3. We change the machine A to commit to 0^b instead of R_i for each $i \notin T'$. ◇

To see that Game 2 and Game 3 are negligible-close, we introduce an intermediate hybrid game, Game 3_j, in which only the first j of the commitments to $R_i, i \notin T$ are replaced by commitments to 0^b. Since at most $a + ns_1$ qubits of quantum memory are used in Game 2 and therefore also in Game 3_j, and since the C_2-commitments to $R_i, i \notin T$ are never opened, from the fact that C_2 is $(\delta, a + ns_1)$-BQS-hiding it follows that Game 3_j and Game 3_{j+1} are δ-close. Note that there are, in the whole game, up to n copies of A and thus up to nc C_2-commitments to some $R_i, i \notin T$. Thus Game 2 = Game 3_0 and Game 3 = Game 3_{nc} are $nc\delta$-close.

Commit phase (on input `committed`):
- When Bob commits to T using C_1, the simulator runs the extractor A_S for C_1 instead of the honest recipient's program. (Since C_1 has recipient Alice, we write A_S, not B_S.) Let T' denote the value extracted by A_S.
- The simulator picks $R_1, \dots, R_m \in \{0,1\}^b$. For each i, Sim commits (honestly) to R_i (if $i \in T$) or to 0^b (if $i \notin T$) using C_2.
- Sim picks a hash function $F \xleftarrow{R} \mathbf{F}$, picks a random $p \xleftarrow{R} \{0,1\}^\ell$, computes the syndrome $\sigma := \mathbf{S}(\underline{R})$, and sends (F, σ, p) to Bob.

Open phase (on input $(\texttt{open}, v)$ **with** $v \in \{0,1\}^\ell$**):**
- Sim picks $\underline{R}' \in \{\underline{R}' : \forall i \in T'.R_i = R'_i, \sigma = \mathbf{S}(\underline{R}'), p \oplus F(\underline{R}') = v\}$ uniformly[10].
- Sim sends $\underline{R}'$ to Bob.
- Sim waits for Bob to open T using C_1. If $T \neq T'$, Sim aborts.
- For each $i \in T'$, Sim (honestly) opens R_i using C_2.

Fig. 3. Simulator Sim for the case of corrupted Bob. The program described in this figure is executed for each instance of the n instances of π_{COM}. Communication with Bob is sent to an internally simulated instance of the adversary Adv.

Game 4. We modify A to set $\underline{R}' := \underline{R}$ and to send $\underline{R}'$ instead of $\underline{R}$ to Bob in step O1. ◇

This modification is for notational purposes only, Game 3 and Game 4 are perfectly close.

Game 5. We modify the way A chooses $F, \underline{R}, \underline{R}', \sigma, p$: In Game 4, we have $F \xleftarrow{R} \mathbf{F}$, $\underline{R} \xleftarrow{R} \{0,1\}^{mb}$, $\sigma := \mathbf{S}(\underline{R})$, $p := v \oplus F(\underline{R})$, $\underline{R}' := \underline{R}$. (We call this distribution $\mathcal{D}_1$.) In Game 5 we use $F \xleftarrow{R} \mathbf{F}$, $\underline{R} \xleftarrow{R} \{0,1\}^{mb}$, $p \xleftarrow{R} \{0,1\}^\ell$, $\sigma := \mathbf{S}(\underline{R})$, $\underline{R}' \xleftarrow{R} \{\underline{R}' : \forall i \in T'.R_i = R'_i, \sigma = \mathbf{S}(\underline{R}'), p \oplus F(\underline{R}') = v\} =: \mathcal{R}_{F,\underline{R},p}$. (We call this distribution $\mathcal{D}_2$.) ◇

To show that Game 4 and Game 5 are negligible-close, we use the following claim:

Claim 1. *Let $R_T := (R_i)_{i \in T}$. For any $v \in \{0,1\}^\ell$, the statistical distance between $(F, R_T, \underline{R}', \sigma, p)$ chosen according to $\mathcal{D}_1$ and $(F, R_T, \underline{R}', \sigma, p)$ chosen according to $\mathcal{D}_2$ is at most $2^{cb+mb/2+\ell/2-\kappa b-1}$.*

The proof of this claim uses the fact that p is the result of applying F to a random variable $\underline{R}$ with high min-entropy. Due to the leftover-hash-lemma [9], p is indistinguishable from randomness. We refer to the full version [16] for the proof. Using Claim 1 and the fact that we have n instances of A, we immediately get that Game 4 and Game 5 are $n(2^{cb+mb/2+\ell/2-\kappa b-1})$-close because the values $(R_i)_{i \notin T}$ are never used by A (except indirectly through $\underline{R}'$, σ, and p).

Finally, note that by construction of Sim, Game 5 and the ideal model are perfectly close. Thus the real and the ideal model are γ-close with $\gamma := 2n\varepsilon +$

[10] Note $\underline{R}'$ can be sampled efficiently since the conditions $\forall i \in T'.R_i = R'_i$, $\sigma = \mathbf{S}(\underline{R}')$, and $p \oplus F(\underline{R}') = v$ are a system of linear equations. This uses that $\mathbf{S}$ is the syndrome of a linear code, and that $\mathbf{F}$ is a family of affine functions.

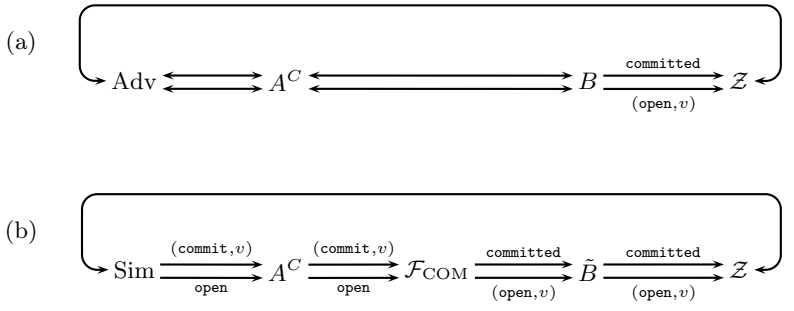


Fig. 4. Networks occurring in the proof of Lemma 9

$nc\delta + n(2^{cb+mb/2+\ell/2-\kappa b-1})$. Since ε, δ are negligible, and n, c are polynomially-bounded, and $2\kappa b - mb - 2cb - \ell$ is superlogarithmic, we have that γ is negligible. Thus π^n_{COM} (a, ns_1)-BQS-UC-emulates $(\mathcal{F}_{\mathrm{COM}}^{A\to B,\ell})^n$ in the case of corrupted Bob. □

Lemma 9. *Assume that ε, δ are negligible, n is polynomially-bounded, and $(1 - \frac{d}{m})^c$ is negligible (in the security parameter k). Assume that C_1 is (ε, a)-BQS-hiding and that C_2 is (δ, s_2)-online-extractable. Assume that the code with syndrome $\mathbf{S}$ has efficient error-correction.*

Then π^n_{COM} (a, nms_2)-BQS-UC-emulates $(\mathcal{F}_{\mathrm{COM}}^{A\to B,\ell})^n$ for corrupted sender A.

Proof. First, we describe the structure of the real and the ideal model in the case that the party A (Alice) is corrupted:

In the real model, we have the environment $\mathcal{Z}$, the adversary Adv, the corruption party A^C, and the honest party B (Bob). The adversary controls the corruption party A^C, so effectively he controls the communication between Alice and Bob. The environment gets Bob's outputs `committed` and $(\mathtt{open}, v)$. See Figure 4 (a).

In the ideal model, we have the environment $\mathcal{Z}$, the simulator Sim (to be defined below), the corruption party A^C, the dummy-party $\tilde{B}$, and the commitment functionality $\mathcal{F}_{\mathrm{COM}}$. The inputs $(\mathtt{commit}, v)$ and `open` of $\mathcal{F}_{\mathrm{COM}}$ are provided by the corruption party A^C and thus effectively by the simulator Sim. The environment $\mathcal{Z}$ controls the dummy-party $\tilde{B}$ and hence gets the outputs `committed` and $(\mathtt{open}, v)$ of $\mathcal{F}_{\mathrm{COM}}$. See Figure 4 (b).

Fix an adversary Adv. To show Lemma 9, we need to find a quantum-polynomial-time simulator Sim with $\mathrm{QM}(\mathrm{Sim}) \leq nms_2$ such that, for any environment $\mathcal{Z}$ with $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$, the real model and the ideal model are negligible-close. This simulator is described in Figure 5. Note that Sim is quantum-polynomial-time: The extractor B_S is quantum-polynomial-time by definition, and computing $\underline{R}^*$ is possible in polynomial-time because the code with syndrome $\mathbf{S}$ has efficient error-correction. Since C_2 is (δ, s_2)-online-extractable and Sim uses m instances of B_S per copy of B, $\mathrm{QM}(\mathrm{Sim}) \leq nms_2$. We use the abbreviations $\underline{R} := R_1 \| \dots R_m$ and similarly for $\underline{R}'$ and $\underline{R}^*$.

Commit phase:
- Sim picks a random $T \subseteq \{1,\dots,m\}$ with $\#T = c$. Then Sim (honestly) commits to T using C_1.
- When Alice commits to $R_1,\dots,R_m$, the simulator runs the extractor B_S for C_2 instead of the honest recipient's program. Let $R'_1,\dots,R'_m$ denote the extracted values.
- Sim waits for (F,σ,p) from Alice.
- Sim computes an $\underline{R}^* \in \{0,1\}^{mb}$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}', \underline{R}^*) \leq (d-1)/2$ (remember that ω is the *block-wise* Hamming distance), computes $v' := p \oplus F(\underline{R}^*)$, and sends $(\texttt{commit}, v')$ to $\mathcal{F}_{\mathrm{COM}}$. (If no such $\underline{R}^*$ exists, we set $v' := \bot$.)

Open phase:
- Sim waits for $\underline{R}$ from Alice.
- Sim (honestly) opens T using C_1.
- For each $i \in T$, Sim waits for Alice to open R_i using C_2.
- Sim checks that the values R_i sent by Alice match the values R_i opened by Alice for all $i \in T$, and that $\sigma = \mathbf{S}(R_1 \| \dots \| R_m)$.
- Sim sends $\texttt{open}$ to $\mathcal{F}_{\mathrm{COM}}$.

Fig. 5. Simulator Sim for the case of corrupted Alice. The program described in this figure is executed for each instance of the n instances of π_{COM}. Communication with Alice is sent to an internally simulated instance of the adversary Adv.

Before we proceed, we introduce two variants of the honest recipient B. The machine B^* behaves like B, but when Alice commits to $R_1,\dots,R_m$ using C_2, B^* runs the extractor B_S for C_2 instead of the honest recipient's program. Call the extracted values $R'_1,\dots,R'_m$. Further, B^* computes an $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}', \underline{R}^*) \leq (d-1)/2$ and then computes $v' := p \oplus F(\underline{R}^*)$. (If no such $\underline{R}^*$ exists, $v' := \bot$.) In the open phase, B^* behaves like B. In particular, B^* outputs $(\texttt{open}, v)$, not $(\texttt{open}, v')$. That is, v' is computed but never used.

The machine B^+ behaves like B^*, but outputs $(\texttt{open}, v')$ instead of $(\texttt{open}, v)$.

By definition of online-extractability, and since B^* does not use the value extracted by B_S, we have that B and B^* are δ-indistinguishable. More precisely, for any network S, we have that $S \cup \{B\}$ and $S \cup \{B^*\}$ are δ-close. Since online-extractability was defined with respect to non-memory bounded adversaries, this holds even if S is not memory bounded.

As in Lemma 8, we proceed by investigating a sequence of games.

Game 1. In the game Game 1$_j$, the j-th instance of B is replaced by an instance of B^*. (Note: only one instance is replaced, not the first j instances.) ◇

We use the following claim:

Claim 2. *Let S be an a-memory bounded network. In an execution of $S \cup \{B^*\}$, let v, v' denote the values v, v' computed by B^*. Then* $\Pr[v \notin \{v', \bot\}] \leq \varepsilon + (1 - \frac{d}{m})^c := \eta$.

We prove this claim below. Since C_1 and C_2 are 0-memory bounded, we have that the machine B is 0-memory bounded. $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$. Thus we can apply Claim 2 to Game 1$_j$. Hence in Game 1$_j$, $\Pr[v_j \notin \{v'_j, \bot\}] \leq \eta$ where v_j, v'_j are the values v, v' computed by B^*. We write $v_j := \bot$ if the open phase fails or does not take place (and hence v_j is not computed by B^*).

Game 2. This game is defined like the real model, except that we use n instances of B^* instead of the n instances of B. ◇

Using the fact that B and B^* are δ-indistinguishable, we get that the real model and Game 2 are $n\delta$-close.

Again using that B and B^* are δ-indistinguishable, we get that $\left|\Pr[v_j \notin \{v'_j, \perp\} : \text{Game 1}_j] - \Pr[v_j \notin \{v'_j, \perp\} : \text{Game 2}]\right| \leq (n-1)\delta$. Thus $\Pr[v_j \notin \{v'_j, \perp\} : \text{Game 2}] \leq \eta + (n-1)\delta$. Since this holds for all $j = 1, \ldots, n$, we get:

$$\Pr[\exists j.\ v_j \notin \{v'_j, \perp\} : \text{Game 2}] \leq n\eta + n(n-1)\delta. \tag{1}$$

Game 3. This game is defined like the real model, except that we use n instances of B^+ instead of the n instances of B. ◇

Notice that Game 2 and Game 3 only differ in the fact that in Game 2 we use instances of B^* and in Game 3 instances of B^+. By definition, B^* and B^+ only differ in the value they output: B^* outputs $(\texttt{open}, v)$ and B^+ outputs $(\texttt{open}, v')$. By (1), the probability that the values v, v' are different in some instance of B^* is bounded by $n\eta + n(n-1)\delta$. Hence Game 2 and Game 3 are $(n\eta + n(n-1)\delta)$-close.

Finally, note that by construction of Sim, Game 3 and the ideal model are perfectly close. Thus the real and the ideal model are γ-close with $\gamma := n\delta + n\eta + n(n-1)\delta = n^2\delta + n\varepsilon + n(1 - \frac{d}{m})^c$. Since δ, ε are negligible, n is polynomially-bounded, and $(1 - \frac{d}{m})^c$ is negligible, we have that γ is negligible. Thus π^n_{COM} (a, nms_2)-BQS-UC-emulates $(\mathcal{F}^{A\to B,\ell}_{\text{COM}})^n$ in the case of corrupted Alice.

Proof of Claim 2. Let $\underline{R}$, $\underline{R}'$, $\underline{R}^*$ and T denote the corresponding values as computed by B^*. We abbreviate $R_T := (R_i)_{i\in T}$ and $R'_T := (R'_i)_{i\in T}$. By *Bad* we denote the event that $R_T = R'_T$ and $\mathbf{S}(\underline{R}) = \sigma$ and $\underline{R} \neq \underline{R}^*$. By construction of B^*, $v \neq \perp$ implies $R_T = R'_T$ and $\mathbf{S}(\underline{R}) = \sigma$. And $v \notin \{v', \perp\}$ implies $\underline{R} \neq \underline{R}^*$. Thus $v \notin \{v', \perp\}$ implies *Bad*. Therefore, to show Claim 2, it is sufficient to show $\Pr[Bad] \leq \eta$ in $S \cup \{B^*\}$. To show this, we again proceed using a sequence of games:

Game 4. An execution of $S \cup \{B^*\}$. ◇

Game 5. We change B^* to halt after receiving $\underline{R}$ from Alice. ◇

Then $\Pr[Bad : \text{Game 4}] = \Pr[Bad : \text{Game 5}]$.

Game 6. We change B^* to commit to some (arbitrary) fixed value T_0 instead of committing to T. ◇

We wish to apply the (ε, a)-BQS-hiding property of C_1 in order to show that $\left|\Pr[Bad : \text{Game 5}] - \Pr[Bad : \text{Game 6}]\right| \leq \varepsilon$. Let B_1 denote the sender in the commitment scheme C_1. By definition, to commit to T (or T_0), B^* internally runs B_1. We construct an adversary A'_1 that interacts with B_1. This adversary simulates $S \cup \{B^*\}$ (with B^* as in Game 5) except for the machine B_1 inside B^*. Note that in Game 5, only the commit phase of C_1 is executed. We let A'_1 output 1 iff *Bad* happens. We define $\hat{B}_1$ like B_1, except that $\hat{B}_1$ ignores its input and

commits to T_0. Let P be the probability that A_1' outputs 1 when running with B_1, and let $\hat{P}$ be the probability that A_1' outputs 1 when running with $\hat{B}_1$. By construction, $P = \Pr[Bad : \text{Game 5}]$ and $\hat{P} = \Pr[Bad : \text{Game 6}]$. Thus we only have to show that $|P - \hat{P}| \leq \varepsilon$. To apply the (ε, a)-BQS-hiding property of C_1 we have to check that A_1' is a-memory bounded. A_1' simulates $\mathcal{Z}$, Adv, and B^*. We have $\mathrm{QM}(\mathcal{Z}) + \mathrm{QM}(\mathrm{Adv}) \leq a$ by assumption. But B^* contains the extractor B_S for C_2 which uses additional s_2 qubits of quantum memory. Yet, B_S is executed after the end of the commit phase of C_1. That is, B^* is executed within a single activation of A_1' (since B_1 is not activated any more after the commit phase). Note that, although A_1' might use more than a qubits during the activation in which B^* is simulated, it stores at most a qubits between activations. Thus A_1' is a-memory bounded (remember that our definition of "a-memory bounded" on page 470 only requires that the memory bound holds between activations). Hence $|P - \hat{P}| \leq \varepsilon$ and thus $\left|\Pr[Bad : \text{Game 5}] - \Pr[Bad : \text{Game 6}]\right| \leq \varepsilon$.

Game 7. We change B^* to choose T only after receiving $\underline{R}'$. ◇

Since T is not used earlier by B^*, $\Pr[Bad : \text{Game 6}] = \Pr[Bad : \text{Game 7}]$. Fix values $\underline{R}$, $\underline{R}'$ and σ with $\mathbf{S}(\underline{R}) = \sigma$. We distinguish two cases, depending on whether there exists an $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$. Case "$\underline{R}^*$ exists": Since $\mathbf{S}$ is the syndrome of a b-block (m, κ, d)-linear code, $\mathbf{S}(\underline{R} - \underline{R}^*) = 0$, hence $\underline{R} - \underline{R}^*$ is a codeword. Hence $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}^*) \geq d$. Using the triangle inequality and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$, we get that $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}') \geq d - (d-1)/2 \geq d/2$. Case "$\underline{R}^*$ does not exist": Since no $\underline{R}^*$ with $\mathbf{S}(\underline{R}^*) = \sigma$ and $\omega(\underline{R}^*, \underline{R}') \leq (d-1)/2$ exists, and since $\mathbf{S}(\underline{R}) = \sigma$, we have that $\omega(\underline{R}, \underline{R}') > (d-1)/2$. Hence $\omega(\underline{R}, \underline{R}') \geq d/2$.

Thus, for any fixed choice of $\underline{R}, \underline{R}', \sigma$, we have $\mathbf{S}(\underline{R}) \neq \sigma$ or $\underline{R} = \underline{R}^*$ or $\omega(\underline{R}, \underline{R}') \geq d/2$.

If $\underline{R} = \underline{R}^*$ or if $\mathbf{S}(\underline{R}) \neq \sigma$, the event Bad does not occur by definition.

If $\omega(\underline{R}, \underline{R}') \geq d/2$, we bound the probability of Bad occurring as follows: Let $D := \{i : R_i \neq R_i'\}$. Then, for random $T \subseteq [m]$ with $\#T = c$, we have $\Pr[Bad] \leq \Pr[R_T = R_T'] = \Pr[T \cap D = \varnothing] \leq (1 - \frac{\#D}{m})^c \leq (1 - \frac{d}{m})^c$.

Thus for any fixed $\underline{R}, \underline{R}', \sigma$ we have $\Pr[Bad] \leq (1 - \frac{d}{m})^c$. By averaging over the choice of $\underline{R}, \underline{R}', \sigma$, we get $\Pr[Bad : \text{Game 7}] \leq (1 - \frac{d}{m})^c$.

Summarizing, we have $\Pr[Bad : \text{Game 4}] \leq \varepsilon + (1 - \frac{d}{m})^c = \eta$.

This shows Claim 2. □

Using Reed-Solomon codes for $\mathbf{S}$, and the extractable commitments from Theorem 6 for C_1 and C_2, we can instantiate the parameters of π_{COM} to satisfy the conditions of Lemmas 8 and 9. Thus we get the following theorem:

Theorem 10. *Let ℓ, n, and a be polynomially-bounded. Then there are choices for the parameters of π_{COM} and a polynomially-bounded integer s such that π_{COM} is polynomial-time, constant-round and π_{COM}^n (a, s)-BQS-UC-emulates $(\mathcal{F}_{\mathrm{COM}}^{A \to B, \ell})^n$.*

General Two-Party Computation. By combining known results [19,10,15], we get a constant-round protocol that quantum-UC-emulates any two-party functionality and uses only commitments from Alice to Bob. Combining this with our protocol π_{COM}, we get our final result (for details see the full version [16]):

Theorem 11 (BQS two-party computation). *Let $\mathcal{G}$ be a classical well-formed*[11] *probabilistic-polynomial-time functionality. Let n and a be polynomially-bounded. Then there is a polynomially-bounded s and a constant-round 0-memory bounded protocol π_{bqs2pc} not invoking any functionality such that π^n_{bqs2pc} (a, s)-BQS-UC-emulates $\mathcal{G}^n$.*

Acknowledgements. We thank Christian Schaffner and Jürg Wullschleger for valuable discussions. This work was funded by the Cluster of Excellence "Multimodal Computing and Interaction".

References

1. Ben-Or, M., Mayers, D.: General security definition and composability for quantum & classical protocols (September 2004), http://xxx.lanl.gov/abs/quant-ph/0409062
2. Bennett, C.H., Brassard, G.: Quantum cryptography: Public-key distribution and coin tossing. In: IEEE International Conference on Computers, Systems and Signal Processing 1984, pp. 175–179. IEEE Computer Society, Los Alamitos (1984)
3. Bennett, C.H., Brassard, G., Crépeau, C., Skubiszewska, M.H.: Practical quantum oblivious transfer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 351–366. Springer, Heidelberg (1992)
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145. IEEE Computer Society, Los Alamitos (2001), full and revised version is [5]
5. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. IACR ePrint Archive (January 2005), full and revised version of [4], http://eprint.iacr.org/2000/067.ps
6. Damgård, I., Fehr, S., Salvail, L., Schaffner, C.: Cryptography in the bounded quantum-storage model. In: FOCS 2005, pp. 449–458 (2005), a full version http://arxiv.org/abs/quant-ph/0508222
7. Dziembowski, S., Maurer, U.: On generating the initial key in the bounded-storage model. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 126–137. Springer, Heidelberg (2004), ftp://ftp.inf.ethz.ch/pub/crypto/publications/DziMau04b.pdf
8. Fehr, S., Schaffner, C.: Composing quantum protocols in a classical environment. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 350–367. Springer, Heidelberg (2009)

[11] Well-formedness describes certain technical restrictions stemming from the proof by Ishai et al. [10]: Whenever the functionality gets an input, the adversary is informed about the length of that input. Whenever the functionality makes an output, the adversary is informed about the length of that output and may decide when this output is to be scheduled.

9. 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), full version http://www.icsi.berkeley.edu/~luby/PAPERS/hill.ps
10. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding cryptography on oblivious transfer – efficiently. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 572–591. Springer, Heidelberg (2008), http://www.springerlink.com/content/0l5v11524816u652/
11. Kilian, J.: Founding cryptography on oblivious transfer. In: STOC 1988, pp. 20–31. ACM, New York (1988)
12. König, R.,Wehner, S.,Wullschleger, J.: Unconditional security from noisy quantum storage. arXiv:0906.1030v2 [quant-ph] (June 2009)
13. Mayers, D.: Unconditionally Secure Quantum Bit Commitment is Impossible. Physical Review Letters 78(17), 3414–3417 (1997), http://arxiv.org/abs/quant-ph/9605044
14. Unruh, D.: Simulatable security for quantum protocols (September 2004), http://arxiv.org/ps/quant-ph/0409125
15. Unruh, D.: Universally composable quantum multi-party computation. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 486–505. Springer, Heidelberg (2010), preprint on arXiv:0910.2912 [quant-ph]
16. Unruh, D.: Concurrent composition in the bounded quantum storage model. IACR ePrint 2010/229 (February 2011), full version of this paper
17. Wehner, S., Wullschleger, J.: Composable security in the bounded-quantum-storage model. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 604–615. Springer, Heidelberg (2008), http://arxiv.org/abs/0709.0492v1
18. Wolf, S., Wullschleger, J.: Oblivious transfer is symmetric. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 222–232. Springer, Heidelberg (2006)
19. Wullschleger, J.: Oblivious-Transfer Amplification. Ph.D. thesis, ETH Zurich (March 2007), arXiv:cs/0608076v3 [cs.CR]

Careful with Composition: Limitations of the Indifferentiability Framework

Thomas Ristenpart[1], Hovav Shacham[2], and Thomas Shrimpton[3]

[1] Dept. of Computer Sciences, University of Wisconsin–Madison, USA
rist@cs.wisc.edu
[2] Dept. of Computer Science & Engineering, UC San Diego, USA
hovav@cs.ucsd.edu
[3] Dept. of Computer Science, Portland State University, USA
teshrim@cs.pdx.edu

Abstract. We exhibit a hash-based storage auditing scheme which is provably secure in the random-oracle model (ROM), but easily broken when one instead uses typical indifferentiable hash constructions. This contradicts the widely accepted belief that the indifferentiability composition theorem from [27] applies to *any* cryptosystem. We characterize the uncovered limitations of indifferentiability by showing that the formalizations used thus far implicitly exclude security notions captured by experiments that have multiple, disjoint adversarial stages. Examples include deterministic public-key encryption (PKE), password-based cryptography, hash function nonmalleability, and more. We formalize a stronger notion, reset indifferentiability, that enables a composition theorem covering such multi-stage security notions, but our results show that practical hash constructions cannot be reset indifferentiable. We finish by giving direct security proofs for several important PKE schemes.

1 Introduction

The indifferentiability framework of Maurer, Renner, and Holenstein (MRH) [27] supports modular proofs of security for cryptosystems. A crucial application of the framework has been to allow proofs in the random oracle model (ROM) [8] to be transferred to other idealized models of computation, where a monolithic random oracle is replaced by a hash function constructed from (say) an ideal compression function. This happens via an elegant composition theorem, the usual interpretation of which is: A proof of security for an arbitrary cryptosystem using functionality F (e.g., a random oracle) continues to hold when the cryptosystem instead uses a second functionality F' (e.g., a hash function built from an ideal compression function), so long as F' is *indifferentiable* from F.

In this paper, we show that this interpretation is too generous. We uncover an application (in the context of secure distributed storage) for which composition fails completely. For this application there is a simple scheme provably secure in the ROM, and yet easily broken when using typical indifferentiable hash constructions. We then begin an exploration of the fall out.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 487–506, 2011.

Random oracles and indifferentiability. Let us give a bit more background on why indifferentiability has proved so useful. A wide range of practical, in-use cryptographic schemes enjoy proofs of security in the ROM [8]; for some schemes, ROM proofs are the only ones known. But most in-use hash-function constructions are not suitable for modeling as a RO, even when assuming the primitive underlying the hash function is ideal (e.g., an ideal compression function), because they admit length-extension attacks [32]. These attacks abuse the structure of the iterative modes-of-operation underlying hash functions such as MD5, SHA-1, and SHA-2. And the weakness they expose has led to practical insecurities [18]. Of course, we can build hash functions that resist known length-extension attacks, but it remains unclear whether the resulting functions would also prevent other, unforeseen structure-abusing attacks.

Coron et al. [15] instead suggest an approach to design hash functions that "behave like" random oracles in a provable sense. Specifically, this requires that a hash function will provide security anywhere a random oracle would. The MRH composition theorem seems to give exactly this, taking $F = \mathrm{RO}$ and $F' = H^f$, the latter being a hash function constructed from an (ideal) primitive f. Thus the needed hash function property is that H^f be indifferentiable from a RO. Importantly, this approach preserves proofs of security as well: the MRH theorem transports a cryptosystem's proof of security in the ROM to a proof of security when using an indifferentiable hash function. A number of recent works prove constructions to be indifferentiable from a RO (e.g., [1, 7, 10, 14, 16, 17, 21]), including many candidates for the NIST SHA-3 competition. Given all this, the consensus opinion appears to be that indifferentiability exactly captures "behaving like" a RO, rules out structure-abusing attacks, and that once a cryptosystem is proven in the ROM it is secure using any compatible indifferentiable hash construction.

Hash-based storage auditing. We now describe an application that shows this consensus opinion to be wrong. In the design of secure distributed systems, the following important problem arises: How can parties in a system verify that a storage server is actually storing the files that it should be? A malicious server might tamper arbitrarily with the data entrusted to it; a rational one might discard the file to save space if detection is unlikely. This problem has received much attention since being formalized in 2007 [2, 22]. The particular example we consider in this paper is inspired by a proof-of-storage challenge-response protocol proposed as part of an earlier system, SafeStore [23]. Consider the following protocol. The client sends a random challenge C to the server; the server proves possession of the file M by computing $Z \leftarrow \mathsf{Hash}(M \,\|\, C)$ using a cryptographic hash function Hash and sending Z to the client, who performs the same computation using her copy of M and compares the result to that sent by the server.

Suppose, for simplicity, that both the file M and the challenge C are d bits long, and consider the case that $\mathsf{Hash} = H^f$, where f is an ideal compression function outputting strings of length $n < d$ bits and H returns the first $n/2$ bits of $f(f(IV, M), C)$. (IV is a fixed constant string.) This construction was shown indifferentiable from a RO in [15]. Thus, the MRH composition theorem combined with the fact that the protocol is secure in the ROM assuredly proves

that the protocol is secure when using H^f. Quite baffling, then, is the observation that the server can cheat! The server simply computes $Y \leftarrow f(IV, M)$ when it first gets M, and then deletes M and stores the (shorter) Y. To answer a challenge C, the server computes $Z \leftarrow f(Y, C)$ and returns the first half of Z as its response. The client's check will succeed even though M is not stored.

The attack abuses a structural feature of typical hash functions that we call online computability. A hash function has this property when it can process its input in successive blocks, storing only a small amount of internal state between blocks. This property is desirable in practice and all indifferentiable hash constructions suggested for practical use have it (see, e.g., [1, 7, 10, 14, 16, 21]). As our example shows, however, online computability can be abused.

Let us pause to take stock of the situation. In Section 4 we prove that the SafeStore-inspired auditing scheme is, indeed, secure in the ROM. The proof of indifferentiability for our $\mathsf{Hash} = H^f$ provided by Coron et al. [15] and the proof of the MRH composition theorem are also both correct. But the server is still somehow able to abuse the structure of H^f. So what is going on here?

Characterizing the problem. The gap is that the *MRH theorem does not apply*. The problem is subtle. Carefully revisiting the MRH theorem and its proof, we find that (loosely speaking) they only apply when a cryptosystem's security is measured relative to a security game using a single, stateful adversary. For example, left-or-right indistinguishability [19] for encryption schemes and unforgeability under chosen message attacks [20] each use just a single, stateful adversary. But the security of the challenge-response auditing protocol we just described is fundamentally two-stage. In the first stage, the adversary (the server) receives the message M, derives from M some state st that is smaller than the size of M, and forgets M. In a second stage it attempts to answer challenges using just st. This is an example of what we call a multi-stage game, a notion we will make formal.

In prior treatments of indifferentiability, the restriction to single-stage games is implicit in the underlying formalization of cryptosystems and adversaries. This restriction has not been mentioned in the literature, and our sense is that no researchers (until now) realized it. For completeness, we restate the MRH indifferentiability composition theorem and give its proof for single-stage games (see Section 3).

Repercussions. We do not necessarily expect that practitioners would (or have) deployed the hash-based auditing scheme above. One can simply use $H^f(C \,\|\, M)$ to achieve (provable) security, and in fact this is the actual protocol used in SafeStore [23]. But the flaw this example uncovers is that the common interpretation of composition actually *encourages* use of an insecure auditing mechanism. This is exactly the opposite of how provable security should guide protocol design.

All of this casts doubt on the security of any scheme relative to a multistage game. The scheme may well have provable security in the ROM, but this does not imply the inexistence of dangerous structure-abusing attacks, even when using indifferentiable hash constructions. And unfortunately the danger is widespread. The recent security notions for deterministic [3, 5, 12], hedged [4], and efficiently

searchable [3] public-key encryption (PKE) are all multi-stage. When formalizing password-based cryptography (e.g. [6, 33]) to allow arbitrary, hash-dependent password sampling algorithms, one uses multi-stage games. A recently proposed hash function nonmalleability security notion [11] is multi-stage. Interestingly, this is the only notion (we are aware of) that formalizes security against length-extension attacks, and so although we expect them to, we do not have *proof* that current indifferentiable hash constructions resist length-extension attacks.

So, we cannot generically use indifferentiability-based composition to modularly argue security in the context of multi-stage games. But it could be that indifferentiability remains a sufficient property to establish security in settings beyond hash-based challenge-response auditing. One might hope to prove, without relying on the MRH composition theorem, that a ROM proof of (say) a deterministic PKE scheme holds still when using any indifferentiable hash construction. This seems reasonable since for the applications just listed, online computability of the hash function does not obviously compromise security.

Yet we prove that such proofs do not exist. Namely, we show in Section 5 that indifferentiability does not imply security in the multi-stage settings mentioned above.

Reset indifferentiability. We present a new notion, reset indifferentiability, that does admit a composition theorem covering both single-stage and multi-stage games. In the indifferentiability framework, functionalities have both an honest and an adversarial interface, e.g. $F.hon$, $F.adv$ and $F'.hon, F'.adv$. Functionality F' is indifferentiable from F if there exists a simulator $\mathcal{S}$ such that no distinguisher can determine when it has access to oracles $F.hon$ and $F.adv$ or to $F'.hon$ and $\mathcal{S}^{F'.adv}$. Reset indifferentiability asks that no distinguisher can differentiate those two sets of oracles, but when the distinguisher can reset the simulator to its initial state at arbitrary times. Randomized simulators use freshly-chosen coins after each reset.

The inability to distinguish when resets are allowed enables proving a composition theorem for multi-stage games because the resets allow one to restart the simulator for each stage. However, it is easy to see that reset indifferentiability is a strong property. While constructions that only require stateless, deterministic simulators can be easily shown to achieve reset indifferentiability, it is unclear if any non-trivial constructions requiring randomized, stateful simulators can meet it. Moreover, there is clear intuition that typical hash constructions are unlikely to be reset indifferentiable — they have the property of online computability. Still, that leaves open if other efficient constructions perform better. We answer this question in the negative, proving that a wide class of single-pass hash function domain extension constructions cannot be shown reset indifferentiable. We leave open the problem of proving the existence (or inexistence) of a domain extender, even an impractical one (i.e., one that makes two or more passes over the message), that is reset indifferentiable.

Direct proofs. Having lost the MRH composition as a general way to transport ROM proofs of security for multi-stage games to the setting where one uses

a hash constructed from an ideal primitive, we take up consideration of a specific security goal from public-key encryption. We prove a theorem establishing the chosen-distribution attack (CDA) security for a number of related, ROM-secure, PKE schemes when these are used with any indifferentiable hash function built according to a design paradigm introduced by Dodis, Ristenpart and Shrimpton [17]. The CDA security notion [4] captures message privacy of a PKE scheme when messages and randomness are (jointly) unpredictable, but otherwise adversarially controlled. In particular, this notion is the goal in the context of deterministic PKE [3, 5, 12], hedged PKE (which provides message privacy even in the face of poor randomness) [4, 31], and efficiently searchable encryption (an extension of deterministic PKE) [3]. As expected, this direct proof of security is complex because we have to work directly in the model of the ideal primitive underlying the hash function. This case study shows that direct security results are possible, restoring confidence that in some multi-stage settings security holds with proposed indifferentiable hash constructions.

Other limitations. In the course of understanding the hash-based auditing counter-example, we uncovered other subtle ways in which composition may fail to help one establish security; a discussion of these appears in the full version [29].

Universal composability. Our results have analogous repercussions for composition frameworks similar to indifferentiability, such as universal composability [13]. We discuss other frameworks in the full version [29].

Discussion. We emphasize that we are *not* recommending that indifferentiability be dropped as a target of hash function design. The class of single-stage games includes many very important ones, and even after our results indifferentiability-based composition remains an elegant way to analyze security for these cases. Instead, we stress that one must be careful when using composition to perform a security analysis, ensuring that it does in fact apply as expected.

2 Preliminaries

A code-based games framework. We formalize a version of the code-based games framework of Bellare and Rogaway [9] for representing security experiments, indifferentiability, and the like. We find code-based games useful for formalizing security definitions, in particular, because they allow us to specify execution semantics (i.e. what runs what, and in what order). Here we give only the most important details, deferring others to the full version of this paper. A *procedure* is a sequence of statements together with zero or more inputs (variables) and zero or more outputs (variables). An *unspecified procedure* is one whose pseudocode, inputs, and outputs are understood from context. An *adversary* is an example of an unspecified procedure. Calling a procedure P means providing it with inputs and running its sequence of statements. During its execution P may itself call other procedures. Say that the code of P expects to be able to call k distinct procedures. We will write $P^{Q_1,Q_2,\ldots,Q_k}$ to denote that these calls are handled by $Q_1, Q_2, \ldots, Q_k$. Procedures P_1 and P_2 are said to *export the*

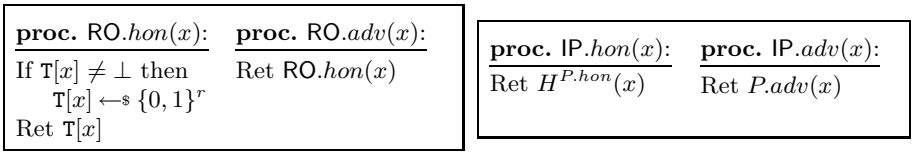


Fig. 1. Procedures implementing the functionality of the random oracle model (ROM) **(left)** and the ideal primitive model (IPM) **(right)**. The number r is set as appropriate for a given context.

same interface if their inputs and outputs agree in number and type. This will typically be clear from context.

A *main procedure* is a distinguished procedure that takes no inputs and has some output. We mark it by **main**. No procedure may call **main**, and **main** can access all variables of other specified procedures. (But not other unspecified procedures.)

Variables are implicitly set initially to default values, i.e. integer variables are set to 0, arrays are everywhere $\perp$, etc. Variables are by default local, meaning they can only be used within a single procedure. The variables used within a procedure maintain their state between calls. A *collection of procedures* is a set of one or more procedures that may instead share their variables. We denote a collection of procedures by using a common prefix ending with a period, e.g. $(P.x, P.y, \ldots)$ and we use the common prefix P to refer to the collection. We will sometimes refer to the unique suffixes, e.g. x, y, as interfaces of P.

Collections of procedures will sometimes implement particular abstract functionalities, for example that of some idealized primitive (e.g. a random oracle). A functionality is a collection $F = (F.hon, F.adv)$; the names of these interfaces, *hon* and *adv* are suggestive as we will see in a moment. When games and adversaries are given access to a functionality a model of computation is induced, for example when the functionality is that of a random oracle, we have the random-oracle model. Thus one can think of functionalities and models somewhat interchangeably. For this work we specifically designate two models. First $\mathsf{RO} = (\mathsf{RO}.hon, \mathsf{RO}.adv)$, shown on the left-hand side of Figure 1, implements a random oracle (with two interfaces) and will give rise to the random-oracle model. Second, let $P = (P.hon, P.adv)$ implement some (ideal) primitive that underlies some understood construction H. Then $\mathsf{IP} = (\mathsf{IP}.hon, \mathsf{IP}.adv)$ shown on the right-side of Figure 1 gives rise to an (ideal) primitive model. For notational compactness, each time we use IP we will specify a construction H and a primitive P and assume these are the ones referred to in Figure 1.

For any two functionalities F_1, F_2, we denote by (F_1, F_2) the functionality that exposes a procedure that allows querying $(F_1.hon, F_2.hon)$ and a procedure that gives access to $(F_1.adv, F_2.adv)$.

A *game* G consists of a single main procedure, denoted "**main** G", together with a set of zero or more other specified procedures. (See for example Figure 2.) A game can make use of a functionality F and a number of adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ together referred to as the *adversary*. We denote this by

$G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$. We fix the convention that the main and specified procedures of G can call $F.hon$ and $\mathcal{A}_1,\ldots,\mathcal{A}_m$ (but may not call $F.adv$) while the adversarial procedures may call $F.adv$ (but may not call $F.hon$). Thus $F.adv$ is the adversarial interface of F, and $F.hon$ is the honest interface. For any $F_1,\mathcal{A}_1,\ldots,\mathcal{A}_m$ and $F_1',\mathcal{A}_1',\ldots,\mathcal{A}_m'$ such that $F_1.hon, F_2.hon$ are interface compatible and $\mathcal{A}_i,\mathcal{A}_i'$ are interface compatible for $1 \le i \le m$, we can write $G^{F_1,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ to mean running game G with one set of external procedures and $G^{F_2,\mathcal{A}_1',\ldots,\mathcal{A}_m'}$ to mean running the same game but now with the second set of external procedures. Running a game $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ means executing the sequence of statements of the game's **main** procedure and the output of G is the value returned by **main**. We denote by $G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y$ the event that the game's output is y, taken over the probability space defined by the coins used to execute G and the coins used in each invocation of the procedures $F.hon, F.adv, \mathcal{A}_1,\ldots,\mathcal{A}_m$. Should G and the adversary not use $F.hon, F.adv$ then we instead write $G^{\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y$. As examples, games that do not use a functionality F are given in Figure 2 while games that do are given in Figures 3 and 4.

For any fixed functionality F and adversary $\mathcal{A}_1,\ldots,\mathcal{A}_m$, two games G and H are *equivalent* if $\Pr\left[\, G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y \,\right] = \Pr\left[\, H^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y \,\right]$ for all values y.

Resources. For simplicity, we fix the convention that each statement of a procedure runs in unit time. The running time of a procedure, then, is the maximum number of statements executed, where the maximum is taken over all possible inputs and over all coins used by the procedure. The number of queries of a procedure is the maximum number of procedure calls it makes in one execution, again with the maximum taken over all possible inputs and all possible coins used by the procedure.

3 Indifferentiability Framework for Single-Stage Games

We describe the indifferentiablity framework [27] using games, unlike prior treatments that used random systems [26, 27] or interactive Turing machines [15]. We feel that using explicit code-based games makes understanding the limitations of indifferentiability easier, because it will enable expressing these limitations as syntactic conditions on the class of games considered. In addition to defining indifferentiability, we will provide a concrete version of the composition theorem given in [27] and characterize its limitations.

Indifferentiability. Fix two functionalities F_1 and F_2. When thinking of indifferentiability from random oracles, for example, we use $F_1 = \mathsf{IP}$ (for some understood H, P) and $F_2 = \mathsf{RO}$. A distinguisher $\mathcal{D}$ is an adversary that outputs a bit. A simulator is a procedure, usually denoted $\mathcal{S}$. Figure 2 defines two games Real and Ideal. Fix some value y (e.g., $y = 1$). The indifferentiability advantage of $\mathcal{D}$ is defined as

$$\mathbf{Adv}^{\mathrm{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) = \Pr\left[\, \mathrm{Real}^{F_1,\mathcal{D}} \Rightarrow y \,\right] - \Pr\left[\, \mathrm{Ideal}^{F_2,\mathcal{D}}_{\mathcal{S}} \Rightarrow y \,\right].$$

We use a concrete security approach, i.e. not providing a strict definition of achieving indifferentiability. However, informally we will say that a functionality

main Real	**proc.** Func(m):	**proc.** Prim(u):
$b' \leftarrow\!\!\$\ \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$ Ret b'	Ret $F_1.hon(m)$	Ret $F_1.adv(u)$
main Ideal$_{\mathcal{S}}$	**proc.** Func(m):	**proc.** Prim(u):
$b' \leftarrow\!\!\$\ \mathcal{D}^{\mathsf{Func},\mathsf{Prim}}$ Ret b'	Ret $F_2.hon(m)$	Ret $\mathcal{S}^{F_2.adv}(u)$

Fig. 2. The games that define indifferentiability. Adversary $\mathcal{D}$ and functionalities F_1, F_2 are unspecified. The simulator $\mathcal{S}$ is a parameter of the game.

F_1 is indifferentiable from a functionality F_2 if for any "reasonable" adversary $\mathcal{D}$ there exists an "efficient" simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is "small". The meanings of "reasonable", "efficient", and "small" will be clear from context.

To get an asymptotic notion, we can assume an implicit security parameter k throughout, and then use the definition of [15]: F_1 is indifferentiable from F_2 if there exists a PT simulator $\mathcal{S}$ such that for any PT $\mathcal{D}$ it is the case that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is negligible in the security parameter. Note that in [27] a different quantifier ordering was used. It said that for all PT $\mathcal{D}$ there must exist a PT simulator $\mathcal{S}$ such that $\mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})$ is negligible in the security parameter. We refer to the [27] notion as weak indifferentiability and to the [15] notion as strong indifferentiability. We will focus on strong indifferentiability here since it implies weak.

Composition. One goal of indifferentiability is to allow the security analysis of a cryptographic scheme when using one functionality to imply security holds when using another. This is enabled by the following, which is a concrete security version of the original composition theorem of Maurer, Renner, and Holenstein [27].

Theorem 1. *Let G be a game expecting access to a functionality and a single adversarial procedure. Let F_1, F_2 be two functionalities with compatible honest interfaces. Let $\mathcal{A}$ be an adversary with one oracle. Let $\mathcal{S}$ be a simulator that exports the same interface as $F_1.adv$. Then there exist adversary $\mathcal{B}$ and distinguisher $\mathcal{D}$ such that for all values y*

$$\Pr\left[\, G^{F_1,\mathcal{A}} \Rightarrow y \,\right] \leq \Pr\left[\, G^{F_2,\mathcal{B}} \Rightarrow y \,\right] + \mathbf{Adv}^{\text{indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) \;.$$

Moreover: $t_{\mathcal{B}} \leq t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}}$, $q_{\mathcal{B}} \leq q_{\mathcal{A}} \cdot q_{\mathcal{S}}$, $t_{\mathcal{D}} \leq t_G + q_{G,1} \cdot t_{\mathcal{A}}$, and $q_{\mathcal{D}} \leq q_{G,0} + q_{G,1} \cdot q_{\mathcal{A}}$, where $t_{\mathcal{A}}, t_{\mathcal{B}}, t_{\mathcal{D}}$ are the maximum running times of $\mathcal{A}, \mathcal{B}, \mathcal{D}$; $q_{\mathcal{A}}, q_{\mathcal{B}}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution; and $q_{G,0}, q_{G,1}$ are the maximum number of queries made by G to the honest interface and to the adversarial procedure. □

The proof of Theorem 1 is readily established by adapting the proof of [27, Th. 1]. We provide a proof here to help support our upcoming discussion.

Proof. Fix any value y. Let $F = (F.hon, F.adv)$ be some unspecified functionality that export the same interface as $(F_1.hon, F_1.adv)$. Let indifferentiability adversary $\mathcal{D}$ be defined as follows. Adversary $\mathcal{D}$ runs game G. Whenever G calls its

honest interface, adversary $\mathcal{D}$ queries $F.hon$ and returns the result. Whenever G calls $\mathcal{A}$, adversary $\mathcal{D}$ runs $\mathcal{A}$ for G using $F.adv$ to answer any queries made by $\mathcal{A}$. Finally $\mathcal{D}$ outputs whatever G outputs. Then by construction $q_{\mathcal{D}} \le q_{G,0}+q_{G,1}q_{\mathcal{A}}$; $t_{\mathcal{D}} \le t_G + q_{G,1}t_{\mathcal{A}}$; and

$$\Pr\left[\,\mathrm{Real}^{\mathcal{D}} \Rightarrow y\,\right] = \Pr\left[\,G^{F_1,\mathcal{A}} \Rightarrow y\,\right] \tag{1}$$

in the case that $F = F_1$. Now we define adversary $\mathcal{B}$ as follows. Adversary $\mathcal{B}$ runs $\mathcal{A}$. When $\mathcal{A}$ queries its oracle, adversary $\mathcal{B}$ runs $\mathcal{S}$ using its $F_2.adv$ oracle to answer any queries $\mathcal{S}$ makes. Adversary $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. By construction, then, we have that $q_{\mathcal{B}} \le q_{\mathcal{A}} \cdot q_{\mathcal{S}}$; $t_{\mathcal{B}} \le t_{\mathcal{A}} + q_{\mathcal{A}} \cdot t_{\mathcal{S}}$; and

$$\Pr\left[\,\mathrm{Ideal}^{\mathcal{D}}_{\mathcal{S}} \Rightarrow y\,\right] = \Pr\left[\,G^{F_2,\mathcal{A}^{\mathcal{S}}} \Rightarrow y\,\right] = \Pr\left[\,G^{F_2,\mathcal{B}} \Rightarrow y\,\right] \tag{2}$$

in the case that $F = F_2$. By substituting according to Equations 1 and 2 into the definition of indifferentiability advantage we derive the advantage relation of the theorem statement. ∎

Single-stage games. The theorem above explicitly restricts attention to games that only use a single adversarial procedure. At first glance, this restriction may seem artificial. Suppose a game G expects access to adversarial procedures $\mathcal{A}_1, \ldots, \mathcal{A}_m$ and now consider generalizing Theorem 1 to account for G. Recall that these adversarial procedures do not share state. In the proof, a key step is defining the adversary $\mathcal{B}$. Following that proof, for this generalization we could define adversarial procedures $\mathcal{B}_1, \ldots, \mathcal{B}_m$ by $\mathcal{B}_i = \mathcal{A}_i^{\mathcal{S}}$ for all i. One may think a proof has been arrived at. However $\mathcal{S}$ is only guaranteed to simulate properly when it maintains its state across all invocations throughout the course of the indifferentiability game. Technically, then, the problem is that the analogue of equation (2) for this proof attempt would fail:

$$\Pr\left[\,G^{F_2,\mathcal{B}_1,\ldots,\mathcal{B}_m} \Rightarrow y\,\right] = \Pr\left[\,G^{F_2,\mathcal{A}_1^{\mathcal{S}},\ldots,\mathcal{A}_m^{\mathcal{S}}} \Rightarrow y\,\right] \neq \Pr\left[\,\mathrm{Ideal}^{\mathcal{D}}_{\mathcal{S}} \Rightarrow y\,\right] .$$

This is true regardless of how we define $\mathcal{D}$. In the next section, we provide a counterexample showing that there is no hope of a proof for this generalization.

All this means that indifferentiability-based composition can only apply to security notions defined via single-stage games, which we now define. Consider a game that has m procedures. We say that a game is *stage minimal* if all games G' that are equivalent to G use the same number of adversarial procedures. We now restrict attention to stage minimal games. Then, an m-stage game is one that has m stages. A single-stage game is one for which $m = 1$ and a multi-stage game is one for which $m > 1$. Let $\mathcal{SG}$ be the set of all single-stage games. Note that $\mathcal{SG}$ includes the games defining indifferentiability above, the classic notions of encryption security such as IND-CPA [19] or IND-CCA [28], unforgeability under chosen message attack UF-CMA [20], and many others.

If $\mathcal{G}$ is the set of all games, then we let $\mathcal{MG} = \mathcal{G} \backslash \mathcal{SG}$ be the set of games that are not single stage. We call any game in $\mathcal{MG}$ a *multi-stage game*. Examples of multi-stage games include chosen distribution attack security for public-key encryption [4] (see Figure 4), non-malleability of hash functions [11], password-based key exchange [6], and others.

Discussion. A game that uses multiple adversarial procedures, but is equivalent to a game with a single adversarial procedure, is not considered multi-stage by our definition above. Many experiments are formalized with multiple adversarial procedures, but the game forwards arbitrary adversarial state from one procedure to the next. It is clear such games are actually equivalent to one with a single adversarial procedure. Some games allow a small amount of state to be passed directly from one adversarial procedure to the next. See for example the hash auditing security property formalized in Figure 3. Here, however, the state is not arbitrary —its length is a fixed constant— and so this game cannot be written with a single adversarial procedure.

We do note, however, that we may extend Theorem 1 to cover multi-stage games that directly share some limited amount of state, but an amount sufficient to enable composition. That is, the shared state must be large enough to transport the state of $\mathcal{S}$ between $\mathcal{B}_i$ calls (in addition to whatever other state an adversary might use). We do not know of any examples of such multi-stage games, and so do not spell out the details of such an extension.

Note that there are other subtleties of composition that might lead to erroneous beliefs and claims. We provide a detailed discussion of these in the full version.

4 A Practically Motivated Counterexample

In this section we define a simple hash function property that is met by a RO, but not met by a broad class of hash functions *proven* to be indifferentiable from a RO. Together these results give a counterexample disproving the desired generalization of Theorem 1 to multi-stage games.

Hash-based storage auditing. The property we study, denoted CRP, is motivated by challenge-response auditing protocols for secure distributed storage [23]. Consider that a client wishes to store some data M on a remote server. It will later verify that M is in fact being stored by sending a random challenge C to the server, and then checking that the server's response matches the hash $H(M \,\|\, C)$. Intuitively, if H is a random oracle, there is no way for the server to "cheat": It must actually store M, or guess the challenge in advance, if it is to respond correctly. (Drawing the challenges from a sufficiently large space or repeating the protocol will make the chance that the server guesses the challenges arbitrarily small.) In particular, if the server stores some state st instead of M, and $|st| \ll |M|$, then we expect the server will fail to respond properly. The CRP experiment in Figure 3 captures a slightly simplified version of this example.

Informally, a CRP-secure hash function H should not admit the storage of a short string (much shorter than the file M) that later allows the server to answer auditing challenges C, except with negligible probability. This guarantees that a rational server interested in saving storage space but subject to auditing will not store some short digest in place of the file.

The following theorem shows that, as expected, a random oracle possesses property CRP. The proof appears in the full version.

$$\frac{\textbf{main } \mathrm{CRP}_{p,n,s}^{F,\mathcal{A}_1,\mathcal{A}_2}}{\begin{array}{l} M \leftarrow\!\!\$\ \{0,1\}^p \\ st \leftarrow\!\!\$\ \mathcal{A}_1^{F.adv}(M) \\ \text{If } |st| > n \text{ then Ret } \mathsf{false} \\ C \leftarrow\!\!\$\ \{0,1\}^s \\ Z \leftarrow\!\!\$\ \mathcal{A}_2^{F.adv}(st, C) \\ \text{Ret } \big(Z = F.hon(M \parallel C)\big) \end{array}}$$

Fig. 3. Game capturing our challenge-response hash function property

Theorem 2. *Fix $p, n, s > 0$. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an adversary that makes a total of q calls. Then*

$$\Pr\left[\,\mathrm{CRP}_{p,n,s}^{\mathsf{RO},\mathcal{A}_1,\mathcal{A}_2} \Rightarrow \mathsf{true}\,\right] \leq \frac{q}{2^{p-n}} + \frac{1}{2^r} + \frac{q}{2^s}$$

where RO *provides the functionality of a random oracle with range $\{0,1\}^r$.* □

Online computability and CRP. We now define a structural property of hash functions, which we refer to as online computability. Consider a hash function $H^f\colon \{0,1\}^* \to \{0,1\}^r$ using some underlying primitive f. Then we say that H^f is (p,n,s)-*online computable* if for $p, n, s > 0$ there exist functions $H_1^f\colon \{0,1\}^p \to \{0,1\}^n$ and $H_2^f\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^r$ such that $H^f(M_1 \parallel M_2) = H_2^f(H_1^f(M_1), M_2))$ for any $(M_1, M_2) \in \{0,1\}^p \times \{0,1\}^s$. Moreover, we require that the time to compute H_1^f and H_2^f is within a small, absolute constant of the time to compute H^f. In words, the hash function H^f can be computed in two stages, processing M_1 and then M_2 sequentially.

We note that most iterative hash function constructions are online computable for a variety of values p, n, s. For example, the so-called NMAC construction from [15]. It uses two underlying ideal objects $f\colon \{0,1\}^{2n} \to \{0,1\}^n$ and $g\colon \{0,1\}^n \to \{0,1\}^n$. Let $f^+\colon (\{0,1\}^n)^+ \to \{0,1\}^n$ be the mapping defined as follows: on input $M = M_1 \parallel \ldots \parallel M_b$, for each $i \in \{1, \ldots, b\}$ compute $V_i = f(V_{i-1} \parallel M_i)$, where V_0 is some fixed n-bit string, and return V_b. Now, let $H^{f,g}(M) = g(f^+(M))$, where the domain is $(\{0,1\}^n)^+$. This construction is (p,n,s)-online computable for any p and s that are multiples of n. Say $p = in$ for any i and $s = n$. Then let $H_1^f(M_1) = f^+(M_1)$ and $H_2^f(V, M_2) = g(f(V, M_2))$. Similarly, many other iterative constructions are online computable for such parameters, for example EMD [7], MDP [21], the Chop and so-called HMAC constructions [15], and numerous SHA-3 candidates.

It is clear to see that any (p,n,s)-online computable hash function *cannot* be CRP for those same parameters. For the NMAC example above, let $\mathcal{A}_1$ output $st = H_1^f(M) = f^+(M)$. Let $\mathcal{A}_2$ output $H_2(st, C) = g(f(st, C))$. The adversary wins with probability 1.

SafeStore and storage auditing in practice. The SafeStore protocol used exactly the opposite ordering of N and M, specifying that audit responses

be computed by $H^f(N \parallel M)$. This construction does indeed have CRP (though one cannot use composition to establish it). The point is that indifferentiability appears to imply that $N \parallel M$ and $M \parallel N$ are equivalently secure. Given the widespread use of hash functions as random oracles in practice (implicitly or explicitly), we must be careful to assess each application's security starting from the ideal primitive underneath the hash function and only use indifferentiability-based composition when it is truly applicable.

5 Indifferentiability Fails for Multi-stage Games

In the last section we saw how indifferentiability-based composition fails for a particular game, this being the CRP game. Here we extend that negative result to show how indifferentiability-based composition fails for many multi-stage games, including ones covering security of password-based key exchange, deterministic public-key encryption, non-malleability of hash functions, and more. To do so, we give a general method to show that indifferentiability does not imply security for games $G \in \mathcal{MG}$.

Our approach will be to show that one can augment any ideal primitive to include a *storage interface*. This will simply take (key,value) pairs from the adversary and allow retrieving values by looking up a key. This augmentation does not affect any existing indifferentiability results involving the primitive — as we show below, a simulator for the original ideal primitive is easily converted to a simulator for the augmented primitive. Finally, we will show how cryptosystems cannot meet some multi-stage notions of security in the augmented primitive model.

Formally, let F_1 be a functionality. Let St be the procedure that exposes a hash table $\mathtt{T}$. That is, on input a pair of strings (X, Y), it sets $\mathtt{T}[X] \leftarrow Y$ and returns nothing. On input a string $(X, \perp)$ it outputs $\mathtt{T}[X]$, which is $\perp$ if $\mathtt{T}[X]$ has yet to be set to another value. Then the storage-augmented functionality $F_1^* = (F_1.hon, F_1^*.adv)$ has the same honest interface as F_1 but $F_1^*.adv$ exposes both $F_1.adv$ and St. That is, $F_1^*.adv = (F_1.adv, St)$.

The following theorem states that if F_1 is indifferentiable from some functionality F_2, then F_1^* is also indifferentiable from F_2. Its proof is straightforward and appears in the full version.

Theorem 3. *Let F_1, F_2 be functionalities and F_1^* be the storage-augmented version of F_1. Let $\mathcal{S}_{\mathcal{B}}$ be a simulator. Then there exists a simulator $\mathcal{S}_{\mathcal{A}}$ such that for all distinguishers $\mathcal{A}$ there exists a distinguisher $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{indiff}}_{F_1^*, F_2, \mathcal{S}_{\mathcal{A}}}(\mathcal{A}) = \mathbf{Adv}^{\text{indiff}}_{F_1, F_2, \mathcal{S}_{\mathcal{B}}}(\mathcal{B})$$

$\mathcal{B}$ runs in time that of $\mathcal{A}$ and uses the same number of queries; $\mathcal{S}_{\mathcal{A}}$ runs in time that of $\mathcal{S}_{\mathcal{B}}$ plus a small constant and uses the same number of queries. □

What Theorem 3 shows is that, as far as indifferentiability is concerned, it does not matter if some portion of the distinguisher's state is exported to an oracle. The intuition behind this result is straightforward: distinguishers in indifferentiability maintain state throughout the experiment and so it hardly matters

whether one stores its state in an oracle or locally. But the ability to store data in an oracle obviates security for many multi-stage games. Here are some examples of cryptographic security goals that are not achievable in a storage-augmented primitive model. Note that all these are feasible in the ROM.

Example: CDA security for public-key encryption. Public-key encryption (PKE) and the chosen-distribution attack (CDA) security goal are defined in Section 7. CDA generalizes deterministic PKE security notions [3, 5, 12], and CDA-secure PKE is useful for efficiently search over encrypted data [3] and defense-in-depth against randomness failures [31]. It is easy to see that if one is working in the F_1^* model, this being a storage-augmented primitive model, then the security notion is unachievable. To attack any scheme, a first-stage adversary $\mathcal{A}_1$ picks (m_0, m_1, r) uniformly, and queries $St(0, (m_0, m_1, r))$. The second-stage adversary $\mathcal{A}_2$ queries $St(0, \perp)$ to retrieve (m_0, m_1, r), encrypts both messages under r, compares the results with the challenge ciphertext, and outputs the appropriate bit. This adversary wins with probability one.

In the full version, we give analogous results for nonmalleability of hash functions [11], password-based authenticated key exchange [6], and others. Interestingly, the hash function nonmalleability notion is the only formal notion that captures resistance to length-extension attacks. This is especially troubling because provable resistance to length extension attacks was a primary motivation for building indifferentiable hash constructions [15].

Discussion. The negative results presented in this section rely on augmenting primitives to incorporate a storage procedure. Of course in the context of hash function design, no one would consider using such a primitive (nor would there necessarily be any way to instantiate one!). Rather these results are used to show that indifferentiability cannot imply security in the context of the multi-stage games considered.

6 Indifferentiability with Simulator Resets

We initiate the exploration of strengthenings of indifferentiability that support composition for multi-stage games. The counter-example of Section 4 indicates that typical indifferentiable hash constructions cannot enjoy such a notion. Indeed, no online computable hash function can meet a strengthening whose associated composition theorem covers the CRP game. Nevertheless, we may hope to design new hash functions that do meet stronger notions.

We propose a strengthening of indifferentiability, called reset indifferentiability, that immediately admits a composition theorem covering multi-stage games.

Reset indifferentiability. We define a version of indifferentiability that requires simulators function even under resets. For any simulator $\mathcal{S}$ we define the procedure pair $\widehat{\mathcal{S}} = (\widehat{\mathcal{S}}.\mathcal{S}, \widehat{\mathcal{S}}.Rst)$. The former procedure is simply a renaming of $\mathcal{S}$. The latter procedure that takes no input and when run reinitializes all of $\widehat{\mathcal{S}}.\mathcal{S}$'s internal variables to their initial values. Likewise, let $F = (F.hon, F.adv)$ be any functionality. Let functionality $\overrightarrow{F} = (\overrightarrow{F}.hon, \overrightarrow{F}.adv) =$

$(F.hon, (F.adv, nop))$ where the procedure pair $\overrightarrow{F}.adv = (F.adv, nop)$ includes a procedure *nop* that takes no input and does nothing. Let F_1 and F_2 be functionalities. Let $\mathcal{D}$ be an adversary that outputs a bit (the distinguisher). Let $\mathcal{S}$ be a simulator. Then we define the reset indifferentiability advantage of $\mathcal{D}$ as

$$\mathbf{Adv}^{\text{reset-indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D}) = \Pr\left[\,\text{Real}^{\overrightarrow{F}_1,\mathcal{D}} \Rightarrow y\,\right] - \Pr\left[\,\text{Ideal}^{F_2,\mathcal{D}}_{\widehat{\mathcal{S}}} \Rightarrow y\,\right].$$

For consistency with our definition of the games Real and Ideal (Figure 2), we implicitly assume there is some distinguished symbol that, when received as input by the procedure Prim, causes the execution of *nop* or $\widehat{\mathcal{S}}.Rst$, respectively.

We have the following composition theorem.

Theorem 4. *Let $G \in \mathcal{G}$. Let F_1 and F_2 be functionalities. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be an adversary and let $\mathcal{S}^{F_2.adv}$ be a simulator that exports the same interface as $F_1.adv$. Then there exist an adversary $\mathcal{B}_1, \ldots, \mathcal{B}_m$ and distinguisher $\mathcal{D}$ such that for all values y*

$$\Pr\left[\,G^{F_1,\mathcal{A}_1,\ldots,\mathcal{A}_m} \Rightarrow y\,\right] \leq \Pr\left[\,G^{F_2,\mathcal{B}_1,\ldots,\mathcal{B}_m} \Rightarrow y\,\right] + \mathbf{Adv}^{\text{reset-indiff}}_{F_1,F_2,\mathcal{S}}(\mathcal{D})\,.$$

Moreover: $t_{\mathcal{B}_i} \leq t_{\mathcal{A}_i} + q_{\mathcal{A}_i} t_{\mathcal{S}}$, $q_{\mathcal{B}_i} \leq q_{\mathcal{A}_i} \cdot q_{\mathcal{S}}$, $t_{\mathcal{D}} \leq m + t_G + \sum_{i=1}^{m} q_{G,i} \cdot t_{\mathcal{A}_i}$, and $q_{\mathcal{D}} \leq q_{G,0} + \sum_{i=1}^{m} q_{G,i} \cdot q_{\mathcal{A}_i}$, where $t_{\mathcal{A}}, t_{\mathcal{B}}, t_{\mathcal{D}}$ are the maximum running times of $\mathcal{A}, \mathcal{B}, \mathcal{D}$; $q_{\mathcal{A}}, q_{\mathcal{B}}$ are the maximum number of queries made by $\mathcal{A}$ and $\mathcal{B}$ in a single execution; and $q_{G,0}, q_{G,i}$ are the maximum number of queries made by G to the honest interface and the i^{th} adversarial procedure (respectively). □

The proof of the above is readily established by adapting the proof of Theorem 1. For $1 \leq i \leq m$, let $\mathcal{B}_i^{F_2.adv} = \mathcal{A}_i^{\mathcal{S}^{F_2.adv}}$. This means in particular that a separate instance of $\mathcal{S}$ is used in each procedure $\mathcal{B}_i$. Then define the distinguisher $\mathcal{D}$, for any compatible functionality $F = (F.hon, F.adv)$, by modifying $\mathcal{D}^{F.hon,F.adv} = G^{F,\mathcal{A}_1,\ldots,\mathcal{A}_m}$ so that a reset call immediately precedes each $\mathcal{A}_i$ call.

Reset indifferentiability can be achieved when one establishes (conventional) indifferentiability using a stateless and deterministic simulator. This is because it is clear resetting such a simulator does not affect its subsequent behavior. Unfortunately it seems challenging to achieve reset indifferentiability for all but trivial constructions, and we will show negative results for efficient constructions below. We leave finding non-trivial constructions, even inefficient ones, as an open question.

On practical domain extension under resets. As mentioned above, online computable hash functions cannot be reset indifferentiable. This is because the composition theorem would then imply such a hash function met the CRP property and the results of Section 4 rule this out. But some efficient hash constructions do meet the CRP property, and so the question remains if any efficient construction meets reset indifferentiability. We answer this question in the negative, ruling out a large class of "efficient" constructions from being reset indifferentiable from a RO.

Fix some $p, n, s, r > 0$ such that $p > n$ and let $N = p + s$. Let P be an arbitrary ideal primitive. We restrict our attention to domain-extension

constructions $H^f\colon \{0,1\}^N \to \{0,1\}^r$ that can be written as $H^P(\langle M_1, M_2\rangle) = H_2^P(H_1^P(M_1) \parallel M_2)$ for any $(M_1, M_2) \in \{0,1\}^p \times \{0,1\}^s$. Here $\langle M_1, M_2\rangle$ represents a unique encoding of M_1, M_2 into an N-bit string; $H_1\colon \{0,1\}^p \to \{0,1\}^n$; and $H_2\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^r$. Importantly, that $p > n$ means that H_1 is compressing. We require that the time to compute one each of the encoding, H_1, and H_2 is within a small, absolute constant of the time to compute H^P. As concrete examples, all online computable functions are trivially included by setting $\langle M_1, M_2\rangle = M_1 \parallel M_2$. But the flexibility endowed by the arbitrary encoding also means we encompass a wider range of H that do not allow online computing. For example, any single pass hash function that can be written in the form above. On the other hand, constructions such as the zipper hash [25] (which makes two passes over a message) are not considered.

The following theorem below shows that no construction fitting the form above is reset indifferentiable, no matter what underlying primitive P is used. Its proof appears in the full version.

Theorem 5. *Let integers p, n, s, r, N, functionality P, and construction H^P be as just described. Let functionality* RO *implement a random oracle with range $\{0,1\}^r$. There exists a reset-indifferentiability adversary $\mathcal{D}$ such that for all simulators $\mathcal{S}$ asking at most q queries,*

$$\mathbf{Adv}^{\text{reset-indiff}}_{\mathsf{IP},\mathsf{RO},\mathcal{S}}(\mathcal{D}) \geq 1 - \left(\frac{q}{2^s} + \frac{q}{2^{p-n}} + \frac{1}{2^r}\right). \qquad \square$$

7 Deterministic, Hedged, and Efficiently-Searchable Encryption

The results thus far reveal that schemes proven secure in the ROM may not be secure when using practical hash function constructions, when security is measured by a multi-stage game. As seen in Section 5 this includes numerous important cryptographic tasks. As a first step, we here take one example, that of deterministic, hedged, or efficiently-searchable public-key encryption, and provide a proof of security when using any one of a number of indifferentiable hash constructions. We choose this example due to the extensive use of the ROM in prior results and the practical importance of the schemes [3, 4, 31]. Of course we cannot rely on Theorem 1, so our proof is done directly in the ideal primitive model. Nevertheless, our main result covers a broad mix of PKE and hash functions.

We focus on the hash construction from [17], which composes a preimage-aware function (see below) with a fixed-input-length RO. While we can do analysis without relying on preimage-awareness, doing so simplifies and modularizes our result. Let $h^f\colon \{0,1\}^* \to \{0,1\}^n$ be a function using some underlying primitive f. Let $g\colon \{0,1\}^n \to \{0,1\}^n$ be a function. Let $H^{f,g}\colon \{0,1\}^* \to \{0,1\}^n$ be defined by $H^{f,g}(M) = g(h^f(M))$. We point out that many hash functions fall into this form, including the so-called NMAC construction [15], MCM [30], NIRP [24], and various SHA-3 competitors.

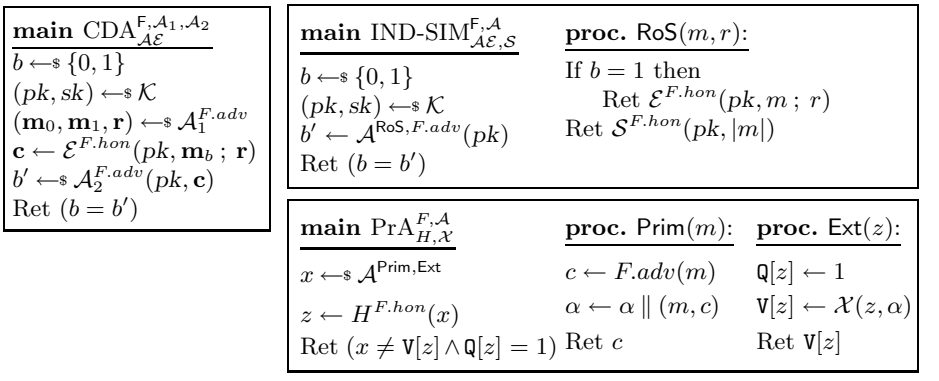


Fig. 4. (Left) The non-adaptive CDA game. **(Right)** The IND-SIM and PrA games

Public-key encryption. Recall that a public-key encryption (PKE) scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms. Key generation $\mathcal{K}$ outputs a public key, secret key pair. Encryption $\mathcal{E}$ takes a public key, a message m, and randomness r and outputs a ciphertext. Decryption $\mathcal{D}$ takes a secret key, a ciphertext, and outputs a plaintext or a distinguished symbol $\perp$. Following [3], we define for any scheme $\mathcal{AE}$ the maximum public-key collision probability by $\mathsf{maxpk}_{\mathcal{AE}} = \max_{w \in \{0,1\}^*} \Pr[\, pk = w \ :\ (pk, sk) \leftarrow\!\!\$\ \mathcal{K}\,]$.

CDA security. In Figure 4 we detail the security game for (non-adaptive) chosen-distribution attacks [4]. This notion, orthogonal to the traditional notion of IND-CPA, captures the security of a PKE scheme when the randomness r used may not be a (sufficiently long) string of uniform bits. For the remainder of this section, fix a randomness length $\rho \geq 0$ and a message length $\omega > 0$. An (μ, ν)-mmr-source $\mathcal{M}$ is a randomized algorithm that outputs a triple of vectors $(\mathbf{m}_0, \mathbf{m}_1, \mathbf{r})$ such that $|\mathbf{m}_0| = |\mathbf{m}_0| = |\mathbf{r}| = \nu$, all components of $\mathbf{m}_0$ and $\mathbf{m}_1$ are bit strings of length ω, all components of $\mathbf{r}$ are bit strings of length ρ, and $(\mathbf{m}_b[i], \mathbf{r}[i]) \neq (\mathbf{m}_b[j], \mathbf{r}[j])$ for all $1 \leq i < j \leq \nu$ and all $b \in \{0,1\}$. Moreover, the source has min-entropy μ, meaning

$$\Pr[\, (\mathbf{m}_b[i], \mathbf{r}[i]) = (m', r') \ \mid\ (\mathbf{m}_0, \mathbf{m}_1, \mathbf{r}) \leftarrow\!\!\$\ \mathcal{M}\,] \leq 2^{-\mu}$$

for all $b \in \{0,1\}$, all $1 \leq i \leq \nu$, and all (m', r').

A CDA adversary $\mathcal{A}_1, \mathcal{A}_2$ is a pair of procedures, the first of which is a (μ, ν)-mmr-source. The CDA advantage for an CDA adversary $\mathcal{A}_1, \mathcal{A}_2$ against scheme $\mathcal{AE}$ is defined by

$$\mathbf{Adv}^{\mathrm{cda}}_{\mathcal{AE},F}(\mathcal{A}_1, \mathcal{A}_2) = 2 \cdot \Pr\left[\, \mathrm{CDA}^{F,\mathcal{A}_1,\mathcal{A}_2}_{\mathcal{AE}} \Rightarrow \mathsf{true}\,\right] - 1\ .$$

Preimage awareness. Dodis, Ristenpart, and Shrimpton's preimage awareness notion [17] generalizes collision resistance to include extractability. Game PrA is defined in Figure 4. We associate to any functionality F, hash construction H, extractor $\mathcal{X}$, and adversary $\mathcal{A}$ the PrA advantage defined by

$$\mathbf{Adv}^{\mathrm{pra}}_{H,F,\mathcal{X}}(\mathcal{A}) = \Pr\left[\, \mathrm{PrA}^{F,\mathcal{A}}_{H,\mathcal{X}} \Rightarrow \mathsf{true}\,\right]\ .$$

We point out that the game PrA does not abide by our convention that only the adversary queries $F.adv$. Thus Theorems 1 and 4 do not apply when $G = \text{PrA}$. This is not a problem for past results or for our results below, both of which do not attempt to conclude PrA via indifferentiability-based composition.

IND-SIM SECURITY. We define a new notion of encryption scheme security that is of technical interest because it is as an intermediate step in proving Theorem 6, shown below. An encryption simulator for a scheme $\mathcal{AE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a procedure $\mathcal{S}$ that takes as input a public key and a message length and outputs a ciphertext. Game $\text{IND-SIM}_{\mathcal{AE},\mathcal{S}}$ is shown in Figure 4. A IND-SIM adversary $\mathcal{A}$ can make multiple queries, but cannot repeat any queries. It measures the ability of an adversary to distinguish between encryptions of a chosen message under chosen randomness and the output of a simulator $\mathcal{S}$. We define the IND-SIM advantage of an adversary $\mathcal{A}$ by

$$\mathbf{Adv}^{\text{ind-sim}}_{\mathcal{AE},\mathcal{S}}(\mathcal{A}) = 2 \cdot \Pr\left[\, \text{IND-SIM}^{\mathcal{A}}_{\mathcal{AE},\mathcal{S}} \Rightarrow \mathsf{true} \,\right] - 1 \ .$$

Note that the adversary can choose the message and also the randomness used to encrypt it. In the standard model this security goal is unachievable if $\mathcal{E}$ uses no further randomness beyond that input. However, we will use IND-SIM security in the ROM when the adversary does not make any RO queries. In the full version we show that for a variety of encryption schemes, IND-SIM security in the ROM against adversaries who do not query the RO is implied by IND-CPA security of an underlying (randomized) scheme.

CDA SECURITY FOR PKE. Theorem 6 below establishes CDA security of PKE schemes that, during encryption, apply $g(h^f(M))$ once to hash an M including an encoding of the public key, as long as the scheme meets the IND-SIM notion above (in the ROM). The ROM schemes for deterministic, hedged, or efficiently-searchable encryption from [3, 4, 31] are of this form and have IND-SIM implied by the IND-CPA security of an underlying randomized encryption scheme. We make no assumptions about f, so the result applies both to hash functions based on an ideal cipher and ideal compression function.

We provide some brief intuition regarding the proof. The PrA security of f^+ means that, to learn anything about the value $g(f^+(M))$, the adversary must query f in order to compute $f^+(M)$. But the inclusion of the public key in the message hashed by $\mathcal{E}$ means that the source $\mathcal{A}_1$ is unlikely to be able to query any of the messages used in computing the challenge ciphertexts. Essentially this means that $\mathcal{E}$ gets randomness via queries to $g(f^+(M))$ that is hidden from the adversary, and this allows one to use the IND-SIM property of $\mathcal{AE}$ to show that ciphertexts leak no information about the challenge message, randomness pairs. This means that $\mathcal{A}_2$ learns nothing about the coins used by $\mathcal{A}_1$, and so the min-entropy of $\mathcal{A}_1$ implies that $\mathcal{A}_2$ has little chance of learning $g(f^+(M))$ outputs for M's used in computing the challenges. The full proof appears in the full version.

Theorem 6. *Let f be a functionality and g be a FIL RO. Let $H^{f,g}(M) = g(h^f(M))$ for some procedure h. Let $\mathcal{AE}$ be a PKE scheme that queries $H^{f,g}$ on a single message per $\mathcal{E}$ invocation, that message including (an encoding of) the*

public key. Let $\mathcal{A}_1, \mathcal{A}_2$ be a CDA adversary making at most q_f queries to f and q_g queries to g and where $\mathcal{A}_1$ is a (μ, ν)-mmr-source. Then for any encryption simulator $\mathcal{S}$ and PrA *extractor $\mathcal{X}$ there exists an* IND-SIM *adversary $\mathcal{B}$ and a* PrA *adversary $\mathcal{C}$ such that.*

$$\mathbf{Adv}^{\mathrm{cda}}_{\mathcal{AE},(f,g)}(\mathcal{A}_1,\mathcal{A}_2) \leq 4{\cdot}\mathbf{Adv}^{\mathrm{ind\text{-}sim}}_{\mathcal{AE},\mathsf{RO},\mathcal{S}}(\mathcal{B}) + 4{\cdot}\mathbf{Adv}^{\mathrm{pra}}_{h,f,\mathcal{X}}(\mathcal{C}) + \frac{2\nu q_g}{2^{\mu}} + 2q_g{\cdot}\mathsf{maxpk}_{\mathcal{AE}}$$

$\mathcal{B}$ makes no random oracle queries, makes ν RoS-queries, and runs in time that of $(\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{C}$ makes at most q_f primitive queries and runs in time at most that of $(\mathcal{A}_1, \mathcal{A}_2)$. □

Acknowledgements

Thomas Ristenpart was supported in part by Mihir Bellare's NSF grant CCF-0915675 and by a UCSD Center for Networked Systems grant. Hovav Shacham was supported by the MURI program under AFOSR Grant No. FA9550-08-1-0352 and (while at the Weizmann Institute) by a Koshland Scholars Program postdoctoral fellowship. Thomas Shrimpton was supported by NSF CAREER grant CNS-0845610.

We thank Mihir Bellare, Mike Dahlin, Daniele Micciancio, and Moni Naor for helpful discussions about this work.

References

1. Andreeva, E., Mennink, B., Preneel, B.: On the indifferentiability of the grøstl hash function. In: Garay, J.A., Prisco, R.D. (eds.) SCN 2010. LNCS, vol. 6280, pp. 88–105. Springer, Heidelberg (2010)
2. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: De Capitani di Vimercati, S., Syverson, P. (eds.) Proceedings of CCS 2007, pp. 598–609. ACM Press, New York (October 2007)
3. 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)
4. Bellare, M., Brakerski, Z., Naor, M., Ristenpart, T., Segev, G., Shacham, H., Yilek, S.: Hedged public-key encryption: How to protect against bad randomness. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 232–249. Springer, Heidelberg (2009)
5. 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)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000)
7. Bellare, M., Ristenpart, T.: Multi-property-preserving hash domain extension and the EMD transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)

8. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993, Fairfax, Virginia, USA, November 3-5, pp. 62–73. ACM Press, New York (1993)
9. 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)
10. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
11. Boldyreva, A., Cash, D., Fischlin, M., Warinschi, B.: Foundations of non-malleable hash and one-way functions. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 524–541. Springer, Heidelberg (2009)
12. 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)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS, Las Vegas, Nevada, USA, October 14-17, pp. 136–145. IEEE Computer Society Press, Los Alamitos (2001)
14. 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)
15. 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)
16. Dodis, Y., Reyzin, L., Rivest, R.L., Shen, E.: Indifferentiability of permutation-based compression functions and tree-based modes of operation, with applications to MD6. In: Dunkelman, O. (ed.) FSE 2009. LNCS, vol. 5665, pp. 104–121. Springer, Heidelberg (2009)
17. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging merkle-damgård for practical applications. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 371–388. Springer, Heidelberg (2009)
18. Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., Stewart, L.: An Extension to HTTP: Digest Access Authentication. RFC 2069 (Proposed Standard) (January 1997), Obsoleted by RFC 2617
19. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
20. 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)
21. 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)
22. Juels, A., Kaliski, B.: PORs: Proofs of retrievability for large files. In: De Capitani di Vimercati, S., Syverson, P. (eds.) Proceedings of CCS 2007, pp. 584–597. ACM Press, New York (October 2007)
23. Kotla, R., Alvisi, L., Dahlin, M.: SafeStore: A durable and practical storage system. In: Chase, J., Seshan, S. (eds.) Proceedings of USENIX Technical 2007, pp. 129–142. USENIX (June 2007)

24. Lehmann, A., Tessaro, S.: A Modular Design for Hash Functions: Towards Making the Mix-Compress-Mix Approach Practical. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 364–381. Springer, Heidelberg (2009)
25. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
26. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Heidelberg (2002)
27. Maurer, U.M., Renner, R., 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)
28. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd ACM STOC, Baltimore, Maryland, USA, May 14-16. ACM Press, New York (1990)
29. Ristenpart, T., Shacham, H., Shrimpton, T.: Careful with composition: Limitations of the indifferentiability framework. Full version of this paper
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. Ristenpart, T., Yilek, S.: When good randomness goes bad: Virtual machine reset vulnerabilities and hedging deployed cryptography. In: Network and Distributed Systems Security – NDSS 2010. ISOC (2010)
32. Tsudik, G.: Message authentication with one-way hash functions. In: Proceedings IEEE INFOCOM 1992, vol. 3, pp. 2055–2059. IEEE, Los Alamitos (1992)
33. Yao, F.F., Yin, Y.L.: Design and analysis of password-based key derivation functions. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 245–261. Springer, Heidelberg (2005)

Efficient Circuit-Size Independent Public Key Encryption with KDM Security

Tal Malkin[1], Isamu Teranishi[1,2], and Moti Yung[1,3]

[1] Columbia University
[2] NEC Japan
[3] Google Inc.
{tal,moti}@cs.columbia.edu, teranisi@ah.jp.nec.com

Abstract. *Key Dependent Message (KDM) secure* encryption is a new area which has attracted much research in recent years. Roughly speaking, a KDM secure scheme w.r.t. a function set $\mathcal{F}$ provides security even if one encrypts a key dependent message $f(sk)$ for any $f \in \mathcal{F}$. We present a construction of an *efficient* public key encryption scheme which is KDM secure with respect to a large function set $\mathcal{F}$. Our function set is a function computable by a polynomial-size *Modular Arithmetic Circuit (MAC)*; we represent the set as *Straight Line Programs* computing multi-variable polynomials (an extended scheme includes all rational functions whose denominator and numerator are functions as above). Unlike previous schemes, our scheme is what we call *flexible*: the size of the ciphertext depends on the degree bound for the polynomials, and beyond this all parameters of the scheme are *completely independent* of the size of the function or the number of secret keys (users). We note that although KDM security has practical applications, all previous works in the standard model are either inefficient feasibility results when dealing with general circuits function sets, or are for a small set of functions such as linear functions. Efficiency of our scheme is dramatically improved compared to the previous feasibility results.

1 Introduction

The design of public key systems that are secure against attackers who are allowed to request ciphertexts that are a function of the system's secret keys is a very active area of research. The initial schemes designed in this area were called "circular" [CL01] and allowed encryption of a secret key or a linear function of a secret key; later, more general functions were considered and the security of these schemes was called *Key Dependent Message (KDM) security* [BRS02]. In particular, we say that a public-key encryption (PKE) scheme is $KDM[\mathcal{F}]$ *secure* (where $\mathcal{F}$ is a class of function), if it is secure even against an adversary who is given public keys $pk_1, \ldots, pk_n$ and has access to encryption of key dependent messages $f(sk_1, \ldots, sk_n)$ for adaptively selected functions $f \in \mathcal{F}$.

Originally motivated by the fact that in some systems keys encrypt other keys (by design or by misuse of protocols), recent research has revealed other

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 507–526, 2011.

important motivations for studying KDM security. On the theoretical side, KDM security can be used to "reconcile" the two fundamental views of security, indistinguishability based security and Dolev-Yao security [AR00,BRS02,ABHS05, BPS08]. This notion also has surprising connection with other fundamental notions, cryptographic agility [ABBC10], and obfuscation [CKVW10]. On the practical side, KDM security is crucial for designing some recent cryptographic protocols. For instance, this notion is used in an anonymous credential system [CL01], where a KDM secure encryption is used to discourage delegation of credentials. Another example is fully homomorphic encryption, where KDM security is used to achieve the full unbounded construction of [G09].

Almost all previous work on KDM security focused on finding KDM[$\mathcal{F}$] (standard model) secure public key encryption schemes for a function class $\mathcal{F}$ that is as large as possible, without much consideration to efficiency. KDM security for the largest set of functions – all functions of bounded Boolean circuit size – was achieved by Barak, Haitner, Hofheinz and Ishai [BHHI10], following previous works such as [BHHO08, BGK09]. However, the schemes in all these works are quite inefficient. For instance, even the most efficient seminal scheme of [BHHO08] requires us to compute $\Theta(\kappa)$ exponentiations over the underlying group $\mathbb{G}$ per each bit of secret key. Here κ is a security parameter. This incurs a factor $\Theta(\kappa^2)$ loss over the standard ElGamal encryption where one can encrypt κ bit by executing $\Theta(1)$ exponentiations. The work of Applebaum, Cash, Peikert, and Sahai [ACPS09] is the only one which explored efficient KDM secure schemes and proposed a much more efficient scheme than others. However, that work is KDM-secure only for linear classes of functions. We discuss previous work in more detail in Section 1.7.

1.1 Our Goals

Efficient Encryption with a Large Set of Queries. In this work we consider the challenge of designing an *efficient* KDM secure scheme that allows the attacker a *large* set of functions $\mathcal{F}$ over the secret key to draw from. The efficiency of the scheme should be comparable with that of the ElGamal encryption (at least for a constant size function family) which is a block-wise semantically secure encryption (and dramatically better than previous KDM-secure works [BHHO08, BGK09, BHHI10] that pay a factor of at least $\Theta(\kappa^2)$ over ElGamal).

Constructing efficient KDM[$\mathcal{F}$] secure scheme for large $\mathcal{F}$ is challenging, and the techniques of previous works seem insufficient. Indeed, all previous works in the standard model are either inefficient feasibility results or possess some noticeable overhead [BHHO08,CCS09, BHHI10,BGK09,BG10], or for a small set of functions such as linear functions [ACPS09].

Flexible Parameters. Another important factor which was ignored in past investigations, is what we call *flexibility parameters* of a scheme, dealing with restrictions on the choice of parameters of functions in $\mathcal{F}$, and when those need to be bounded (determined). For example, consider the number n of keys (users) in $f(sk_1, \ldots, sk_n) \in \mathcal{F}$. Some schemes (e.g., [BG10]) do not allow to select n

flexibly, but rather require the maximum n to be fixed before key generation, and KDM security proof is subject to the scheme being so restricted. Clearly, a flexible scheme that allows unbounded (freely determined) n even after key generation is desirable.

For flexibility determination we will consider the following parameters: *number of keys*, the *size of the circuit* of the function, and its *degree as a polynomial*. For these parameters we will ask whether they need to be determined (bounded) as an input (1) at key generation, (2) at encryption time, or (3) remain unbounded throughout. These are listed in order of increased flexibility, and the more flexible a KDM-scheme is, the more desirable.

1.2 Our Results

We design block-wise encryption (i.e., a scheme encrypting messages as blocks rather than bit by bit) that is efficient, flexible, and provides KDM-security against a large set of functions, based on the DCR assumption.

Roughly, our scheme provides KDM security for polynomials and rational functions (ratios of two polynomials) over a ring of integers (modulo N), with the flexibility of allowing unbounded number of keys, circuit of unbounded size, computing a polynomial whose degree is bounded only at encryption time. This is the first time flexibility is defined, and the first time that KDM security has been achieved with this level of flexibility (no dependence on number of keys, no dependence at key generation time on the circuit, and depending only on the degree, but not the size, of the circuit at encryption time.) We also give a general *triple mode proof framework* for proving KDM security, which was implicitly used in previous works including ours.

We elaborate on these contributions below.

1.3 Function Classes

A function f is called *MAC (Modular Arithmetic Circuit)* if there exists a polynomial-size circuit for f whose inputs are variables $X_1, \ldots, X_n$ and constants of $\mathbb{Z}_K$ and whose gates are $+$, $-$, or $\cdot$ over a ring $\mathbb{Z}_K$. That is, f is MAC computable iff it can be computed from variables and constants of $\mathbb{Z}_K$ by applying $+$, $-$, and $\cdot$ modulo K a polynomial number of times (this is also referred to as a *straight line program* with n variables over $\mathbb{Z}_K$ which can be viewed as computing a polynomial function).

The set of functions $\mathcal{MAC}_d[K]$ contains functions whose total degree (as a polynomial) is not more than $d = \mathsf{poly}(\kappa)$, and which are MAC computable over $\mathbb{Z}_K$. The set of functions $\mathcal{Q}(\mathcal{MAC}_d[K])$ is the set of functions which can be represented by a ratio (division) of two MAC computable functions. These two are the sets of functions that our two schemes (for different K) will be KDM-secure against, respectively.

Richness of Function Classes. These classes are quite large. To start with, $\mathcal{MAC}_d[K]$ includes all functions which are represented by polynomial length formula with total degree $\leq d$, where a *formula* is a (well-formed) word on the

set of alphabets $\{X_1, \ldots, X_n, +, \cdot, {}^m, (,), a \mid m \in \mathbb{Z}, a \in \mathbb{Z}_{N^{s-1}}\}$. $\mathcal{Q}(\mathcal{MAC}_d[K])$ includes all functions that are represented by such a formula that also allows division (or inverse), such as

$$((2X_1+X_2+\cdots+X_n)^{10}+(X_1+4)\cdots(X_n+4)(X_2+4X_3)^{-1})^2+3({X_3}^{-3}-2{X_2}^2X_1)^2 \bmod K.$$

In fact, the $\mathcal{MAC}_d[K]$ class is much richer, as such formulas can be re-interpreted as log depth circuits, while MACs can have polynomial depth. MACs can be simulated by a straight line program with as many variables as the depth of the circuit (i.e., simply traverse the circuit in topological order); the result is a polynomial and we only require its degree to be bounded (at most d). Note that $\mathcal{MAC}_d[K]$ can contain polynomials that have exponentially many terms. The simplest example of such a function is

$$f(X_1, \ldots, X_n) = (X_1 + \cdots + X_n)^d \bmod K \qquad \text{for } n = \mathsf{poly}(\kappa), d = \mathsf{poly}(\kappa)$$

This function can clearly be computed by a MAC (with polynomial number of gates), but it has an exponential number of terms $\{{X_1}^{\varepsilon_1} \cdots {X_n}^{\varepsilon_n} \mid \varepsilon_1+\cdots+\varepsilon_n = d\}$ when expanded.

On the other hand, by definition, these classes cannot compute functions that have an exponential degree (as we need the polynomial degree for our KDM secure construction). For example, $\mathcal{MAC}_d[K]$ does not contain $f(X) = X^{2^\kappa}$ (even though this f can be computed by a polynomial size MAC).

1.4 Properties of Proposed Schemes

We construct two efficient block-wise KDM secure PKE schemes as following:

KDM Security: Our schemes are KDM$[\mathcal{MAC}_d[N^{s-1}]]$ and KDM$[\mathcal{Q}(\mathcal{MAC}_d[N])]$ secure respectively, where N is the product of two safe primes and $s \geq 2$.

Computational Costs: An encryption and a decryption of our first scheme require only $2d+4$ $(d+2)$ exponentiations in $\mathbb{Z}_{N^s}$ when one encrypts (decrypts) a ciphertext of $f(sk_1, \ldots, sk_n)$ with degree d. (The costs double for our second scheme).

Flexibilities of Parameters: In our schemes, for the first time, the number n of keys of a function $f(sk_1, \ldots, sk_n)$, the number ℓ of $\{+, -, \cdot\}$ in a MAC computing f (i.e., circuit size), and the total degree d of f can be selected flexibly by an adversary. Specifically, our schemes are KDM secure even under the condition that an adversary can choose these parameters arbitrarily and adaptively, where n, ℓ are completely unrestricted, and d is needed as input at the encryption stage. (The obvious upper bound for these parameters is the number of steps of the adversary herself, which is some polynomial in κ.) This also means that the party who is encrypting can choose which d to protect against for each encryption, depending on the level of sensitivity or perceived KDM-attack risk for that encryption.

Moreover, the efficiency of our schemes (both in terms of computational cost and ciphertext length) do not depend on n and ℓ. Our schemes therefore remain efficient even if these parameters are quite big. This is in contrast with recent schemes, as will be compared below.

1.5 Triple Mode Proof Framework

For our security proofs, we give a general framework for proving KDM security, the *triple mode proof framework*, which clarifies the structure of the proof, and highlights the crucial parts. Intuitively, the definition of KDM security involves an adversary that can access an encryption oracle, asking for encryptions of $f(sk_1, \ldots, sk_n)$, and getting either the correct key-dependent encryptions, or random bit encryptions; the adversary should not be able to distinguish between these two cases. To prove security, we need to construct a simulator which can use any such distinguishing adversary to break the underlying assumption. However, this is problematic, because without the secret key, it is not clear how the simulator can compute encryptions of key-dependent functions, however, with the secret key, these two cases could be distinguishable, and breaking the underlying assumption is no contradiction.

Our triple mode proof framework solves this issue by preparing three games (or "modes") for the security proof, and using *two* simulators, where the first one knows the secret key but the second one does not. The first and last game correspond to the usual key-dependent vs. random encryption, as above. The key idea is to find an intermediate game where the encryptions are independent of the secret key, yet it is indistinguishable from the first game *even given the secret key.*

The suggested notion unifies security techniques, since it can be shown that known standard model KDM secure schemes [BHHO08, ACPS09,BG10] as well as ours have the structure suitable for the triple mode proof frameworks.

1.6 Techniques

Here we describe our main ideas in a way that is informal and inaccurate, yet hopefully it provides good intuition. We use the following approach.

- Construct an efficient block-wise KDM secure scheme $\mathcal{PKE}$ w.r.t. moderately large and simple set $\mathcal{F}$.
- Reduce the KDM security of $\mathcal{PKE}$ w.r.t. the quite large and complex set $\mathcal{MAC}_d[N^{s-1}]$ into KDM security of it w.r.t. $\mathcal{F}$, by "compressing" the complex structure of MAC into the simple structure of $\mathcal{F}$.

It is important to choose $\mathcal{F}$ carefully, so as it is not too large or too small. We choose $\mathcal{F}$ to be the set of univariate polynomials $f(X) = \sum_{j=0}^{d} a_j X^j$, and construct a KDM scheme w.r.t. $\mathcal{F}$ based on new idea, the *cascaded Paillier ElGamal* and show it satisfies KDM security.

KDM scheme for uni-variate polynomials : cascaded Paillier ElGamal. Our starting point is previous work (in particular [BG10]), which achieved KDM-security for linear functions on bits. Transforming that to block-wise linear functions on entire secret keys is straight forward[1].

[1] [BG10] did not mention this, probably because they also consider other goals such as leakage resilience, for which they focused on bit functions.

An encryption of a message M in the resulting [BG10] scheme is ciphertext of the form $(C_1, A(M)C_2)$, where $A(M)$ is the only part that depends on the message (a la ElGamal encryption); concretely our starting point was [KTY09]. The KDM-security relies on the fact that when the message is a linear function $f(x) = ax + b$ of the secret key x, its encryption $(C_1, A(ax + b)C_2)$ is indistinguishable from the encryption $(A(a)C_1, A(b)C_2)$, which now no longer depends on $(ax+b)$, but only on a, b (and thus can be simulated using the secrecy of the key x).

To extend this to a function in $\mathcal{F}$ that is a degree d polynomial $f(x)$, we write $f(x) = f'(x)x + b$, and can say, similarly to above, that the ciphertext $(C_1, A(f'(x)x + b)C_2)$ is indistinguishable from $(A(f'(x))C_1, A(b)C_2)$. Now the right term is independent of the secret key, and the left term does depend on the secret key, but only as a degree $d - 1$ polynomial $f'(x)$.

Our "cascaded Paillier ElGamal" scheme thus ElGamal encrypts the left element to get $(C'_1, A(f'(x))C'_2)$, and apply the same idea recursively to reduce the degree of $f'(x)$ by one. We may continue recursively constructing these pairs, each time encrypting the left element with a fresh encryption. The final ciphertext we output is a tuple consisting of all right elements, and the last left element.

Reduction from MACs to Univariate Polynomials. In order to achieve our general class, for many keys, we reduce their number by setting secret keys sk_i to the sum $\mu+\alpha_i$ of one "secret" μ and a "difference" α_i from μ[2]. Then a multivariate polynomial $f(sk_1, \ldots, sk_n)$ computed by an MAC can be re-interpret as a univariate polynomial $\phi(\mu) = f(\mu + \alpha_1, \ldots, \mu + \alpha_n)$ of μ. Namely, KDM security w.r.t. $f(sk_1, \ldots, sk_n)$ is reduced to KDM security w.r.t. a univariate polynomial $\phi(\mu)$.

The crucial point of the above reduction is that the coefficients of $\phi(\mu)$ can be computed in polynomial time if f is an element of $\mathcal{MAC}_d(N^{s-1})$. This fact enables simulators to remain polynomial time algorithms. This is why we use $\mathcal{MAC}_d(N^{s-1})$.

The Second Scheme: Security for Quotient of MACs. A ciphertext of our second scheme for a message M is a tuple $(C', C'') = (\mathsf{Enc}(MR), \mathsf{Enc}(R))$, where Enc is the encryption of the first scheme and R is a randomly selected element. Intuitive meanings of C' and C'' are the encryptions of "numerator" and the "denominator" of a key dependent message $f(\overrightarrow{sk}) = f'(\overrightarrow{sk})/f''(\overrightarrow{sk})$ computed by two MAC computable functions f' and f''. Here $\overrightarrow{sk} = (sk_1, \ldots, sk_n)$.

Clearly, encryption (C', C'') of key dependent message $f(\overrightarrow{sk})$ has the same distribution as $(\mathsf{Enc}(f'(\overrightarrow{sk})S), \mathsf{Enc}(f'(\overrightarrow{sk})S))$ for randomly selected S. We therefore succeed in reducing the KDM security of the second scheme to KDM security of the encryptions $\mathsf{Enc}(f'(\overrightarrow{sk})S)$ and $\mathsf{Enc}(f'(\overrightarrow{sk})S)$ of the first scheme.

[2] We note this idea has been used in several previous works to handle multiple keys, but in our case it provides much more powerful results for the class of functions, since we start from a polynomial degree function, rather than a constant degree one.

The only problem of the above idea is that the denominator $f''(\overrightarrow{sk})$ can be 0 and therefore $f = f'/f''$ cannot be defined in this case. But we can overcome this problem by modifying the scheme slightly and proving the security carefully.

1.7 Related Work and Comparison to Our Schemes

Fig.1 shows the comparison among the previous schemes and ours. Here κ is the security parameter. Note that all schemes except for [CCS09] are KDM-CPA secure, while [CCS09] is KDM-CCA2 secure.

Explanation of Fig.1: The "size" ℓ represents the number of gates in a MAC (resp. in a circuit) computing a function $f(sk_1, \ldots, sk_n)$ in the case of our schemes (resp. [BHHI10]). The column "Flexibility of Param." describes the flexibilities of the parameters n, d, and ℓ of a function $f(sk_1, \ldots, sk_n)$. "KeyGen bounded" means that one has to fix the maximum of the parameter before the key generation, KDM security holds only when the parameter is less than the maximum, and efficiency of the scheme depends on this maximum. "Enc bounded" means that we do not have to fix such maximums, and KDM security hold for all values of the parameter, but the parameter is needed for encryption, and efficiency of the scheme depends on its value. "Unbounded" means the scheme (at all stages) is independent from the value of this parameter.

The column "|Ciphertext| per |Message|" represents the ratio between the ciphertext length and the message length.

We note that we can improve properties of known schemes in Fig.1 using known techniques:

- Using the technique of [ACPS09], "|Ciphertext|/|Message|" of [BHHO08] and [BG10] can be reduced to $O(1)$.
- If one restricts the function to polynomials, [BHHI10] can be unbounded in n.

Comparison with [ACPS09]: They deal with linear functions only compared to our larger set of MACs, and they are based on a lattice-based assumption, LWE, while we employ the DCR assumption.

Comparison with [BHHI10, A11]: These schemes achieve KDM secure schemes w.r.t. the largest set of functions (strictly richer than ours), though their schemes are merely a feasibility result, relying on and are application of the inefficient general secure computation. While it is important to know the feasibility of such KDM secure schemes, they are not comparable to schemes that are more efficient then including the encryption of the circuit, or ,as in our scheme, independent of the circuit size. This is especially true given the applications of such encryption schemes. The size ℓ of circuit is encryption bounded in [BHHI10, A11], while it is unbounded in our scheme, which requires only the degree d to be encryption bounded.

Comparison with [BHHO08, BGK09, BG10]: The first to achieve KDM security without a random oracle were [BHHO08]. Their scheme was used as a basis for [BGK09] and [BG10], who achieve KDM secure schemes w.r.t. the

	Functions	|Ciphertext| per |Message|	# of Users n	Flexibility of Parameters		
				max deg d	Size ℓ	Assum -ption
[BHHO08] [CCS09]	Linear of bits	$O(\kappa)$	Un-bounded	-	-	DDH
[BGK09]	Polynomial of Bits with deg= $O(1)$	$O(\kappa^{d+1})$	Un-bounded	KeyGen	-	DDH
						LWE
[BG10] + [BGK09]		$O(n\kappa + \kappa^{d+1})$	KeyGen			QR DCR
[BHHI10] [A11]	Bounded Size Circuit	$O(n\mathsf{poly}(\kappa) + \kappa\ell)$	Enc	-	Enc	DDH LWE QR
[ACPS09]	Linear of block	$O(1)$	Un-bounded	-	-	LWE
First Scheme	$\mathcal{MAC}_d[N^{s-1}]$	$O(d)$		Enc	Un-bounded	DCR
Second Scheme	$\mathcal{Q}(\mathcal{MAC}_d[N])$					

Fig. 1. Parameters of Our Scheme and Previous Work

set of *constant-degree polynomials* of *bits* of secret keys (as we said they can describe also blocks of keys rather than bits). The degrees of polynomials in [BGK09, BG10] have to be bounded by small constant (because the ciphertext lengths in these schemes grow exponentially with this degree), and therefore are KeyGen bounded. In contrast, in our schemes the degree of the polynomials can be polynomial and is encryption bounded, and the number of terms can be super polynomial. For the scheme of [BG10] the number of users is KeyGen bounded, while it is unbounded in our schemes. Finally, the schemes [BGK09, BG10] (and to a lesser extent [BHHO08]) are quite less efficient than ours.

Other Related Works. The notion of KDM security was defined by Black, Rogaway, and Shrimpton [BRS02], although Camenisch and Lysyanskaya [CL01] independently defined a similar notion called *circular encryption* earlier. Earlier works of KDM security were studied in the random oracle model. [BDU08] showed that the well-known OAEP encryption is KDM secure. [HK07] generalize the notion of KDM to pseudorandom functions.

Constructing KDM secure schemes in the standard model was a long-standing open problem. It was partially solved by [HU08] for the case of a symmetric key encryption. The first PKE which is KDM secure in the standard model was proposed in the seminal work of Boneh, Halevi, Hamburg, and Ostrovsky [BHHO08]. The first CCA2 and KDM secure PKE was proposed by [CCS09]. A general transformation starting from KDM secure PKE for a certain class of functions and boosting it to a larger class was shown by [A11]. Examples of PKE which satisfy semantic security but not KDM (specifically, 2-circular) security were shown independently by [GH10] and [ABBC10].

[HH09] showed that KDM[$\mathcal{F}$] security of an encryption scheme for "quite large" $\mathcal{F}$ cannot be proved as long as the reduction's proof of security treats the function $f \in \mathcal{F}$ and the adversary as black boxes.

The connection between the adaptive Dolev-Yao model and generalized versions of KDM security are studied by [BPS08], while further connections of KDM security with agility and obfuscation are shown by [ABBC10] and [CKVW10], respectively.

We refer the reader to [MTY11] for a survey on KDM security results and applications.

2 Preliminaries

Notations and Terminologies: For a natural number n and $m \leq n$, let $[n]$ and $[m..n]$ be the sets $\{1, \ldots, n\}$ and $\{m, \ldots, n\}$ respectively. For a real number x, $\lfloor x \rfloor$ denote the largest integer not greater than x.

Polynomials and Rational Functions: For a polynomial $f(X_1, \ldots, X_n) = \sum_{j_1,\ldots,j_n} a_{j_1,\ldots,j_n} {X_1}^{j_1} \cdots {X_1}^{j_n} \bmod K$, the *(total) degree* $\deg f$ of f is $\max\{\sum_k j_k \mid a_{j_1,\ldots,j_n} \neq 0 \bmod K\}$. A *rational function* over $\mathbb{Z}_K$ is a function which can be written as $f(X_1, \ldots, X_n)/g(X_1, \ldots, X_n)$ using two polynomials f and g over $\mathbb{Z}_K$.

Paillier Group: Let N be the product of two safe primes and T be $1+N$. Define three subsets of $\mathbb{Z}_{N^s}$, sets of Quadratic Residue, Square Composite Residuosity, and Root of the Unity, as follows:

- $\mathcal{QR}[N^s] = \{u^2 \bmod N^s \mid u \in \mathbb{Z}_{N^s}\}$
- $\mathcal{SCR}[N^s] = \{r^{2N} \bmod N^s \mid r \in \mathcal{QR}[N^s]\}$,
- $\mathcal{RU}[N^s] = \{T^M \bmod N^s \mid M \in [0..N^{s-1}]\}$.

Theorem 1 ([P99, DJ01, KTY09]). *There exists a polynomial time computable bijective homomorphism $L : \mathcal{RU}[N^s] \to \mathbb{Z}_N$ satisfying the following property:*

$$\forall M \in \mathbb{Z}_{N^{s-1}} \quad : \quad L(T^M) = M \bmod N^{s-1}.$$

Moreover, the following property holds:

$$\mathcal{QR}[N^s] = \mathcal{SCR}[N^s] \times \mathcal{RU}[N^s].$$

Definition 2. (Decision Composite Residuosity (DCR) Assumption [P99, DJ01]). Let $s \geq 2$ be an integer. There exists a generator Gen of the product N of two safe primes such that the following value is negligible for κ for any polynomial time adversary A:

$$\begin{aligned} &|\Pr[N \leftarrow \mathsf{Gen}(1^\kappa), g \leftarrow \mathcal{SCR}[N^s], b \leftarrow \mathsf{A}(s, N, g) : b = 1] \\ &- \Pr[N \leftarrow \mathsf{Gen}(1^\kappa), g \leftarrow \mathcal{QR}[N^s], b \leftarrow \mathsf{A}(s, N, g) : b = 1]|. \end{aligned}$$

Our DCR assumption is subtly different from the original one [P99, DJ01], where g is taken from $\{r^N \bmod N^2 \mid r \in \mathbb{Z}_{N^s}\}$ (or $\mathbb{Z}_{N^s}$) in the first (or second) game, but ours clearly follows form the original one by squaring g.

2.1 KDM Security

Public Key Encryption Scheme: In this work, a public key encryption scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ has generator Setup of a system parameter prm, such as a group description, and all users commonly use this parameter as inputs of the other three algorithms.

Description of Functions: As in previous works, we implicitly assume that each function f has some polynomial size *description* D. (In the case of our schemes, D is an MAC or MACs computing f.) We let f_D denote the function corresponding to D.

KDM Security: For a public key encryption scheme $\mathcal{PKE}$=$(\mathsf{Setup}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ and its secret key space SkSp and message space MeSp, let

$$\mathcal{F}^{(n)} \subset \{f \ : \ \mathsf{SkSp}^n \to \mathsf{MeSp}\}, \qquad \mathcal{F} = \cup_{n=1}^{\infty} \mathcal{F}^{(n)}.$$

To simplify, we assume that SkSp and MeSp depend only on the system parameter prm. For a natural number n and a bit b, consider the following game:

– $\mathsf{GameKDM}^b_{\mathsf{A}}[\mathcal{F}, n]$:

$$prm \leftarrow \mathsf{Setup}(1^\kappa), b' \leftarrow \mathsf{A}^{\mathcal{O}_{\mathsf{Kg}}, \mathcal{O}^{(b)}_{\mathsf{Enc}}}(prm, (pk_j)_{j\in[n]}), \text{ Output } b'.$$

Above, A is allowed to make polynomial number of queries adaptively:

– If A makes the i-th query new to $\mathcal{O}_{\mathsf{Kg}}$,it generates the i-th key pairs$(pk_i,sk_i) \leftarrow \mathsf{Kg}(prm)$ and sends pk_i as an answer.
– If A makes the i-th query (i, D) to $\mathcal{O}^{(b)}_{\mathsf{Enc}}$ where $i \in [n]$ and D is a description of a function of $\mathcal{F}^{(n)}$, the oracle answers the following C (below, $\mathfrak{o}$ be some fixed element of MeSp and n is the number of keys generated by $\mathcal{O}_{\mathsf{Kg}}$):

$$C \leftarrow \begin{cases} \mathsf{Enc}_{prm}(pk_i, f_D(sk_1, \ldots, sk_n)) & \text{if } b = 1 \\ \mathsf{Enc}_{prm}(pk_i, \mathfrak{o}) & \text{Otherwise.} \end{cases}$$

We say that $\mathcal{PKE}$ is $KDM[\mathcal{F}]$ *secure* if the following advantage is negligible for any n and any polynomial time algorithm A.

$$\mathsf{Adv.KDM}_{\mathsf{A}}[\mathcal{F}] = |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|\,.$$

One can easily check that the above definition is independent of the choice of $\mathfrak{o}$, if $\mathcal{PKE}$ satisfies indistinguishability.

Our definition of KDM security is stronger than that of the previous one [BRS02, BHHO08]: Ours allows an adversary to get new key adaptively while the previous one [BRS02, BHHO08] does not. (In other words, using the terminology of Section 1.4, the number n of keys becomes "unbounded.") Some known schemes (e.g., [BG10]) require the maximum n to be fixed before key generation, and KDM security can be proved only when n is less than the predetermined maximum. Our scheme does not require n to be fixed and therefore can be proved under our stronger KDM security definition.

2.2 Modular Arithmetic Circuit

Definition 3 (Modular Arithmetic Circuit (MAC)). A *Modular Arithmetic Circuit* D (MAC) is a circuit whose inputs are variables $X_1, \ldots, X_n$ and constants of $\mathbb{Z}_K$ and whose gates are $+$, $-$, or $\cdot$ over $\mathbb{Z}_K$. (We stress that the fan-out of each gate is unbounded.) For MAC D, the number of gates in D is called *size* of D.

Note that in our case MAC is equivalent to *straight line program* with unlimited number of registers. Clearly, a function computed by MAC is a polynomial over $\mathbb{Z}_K$. We let f_D denote f when MAC D computes f.

For natural numbers n, d, and ℓ, $\mathcal{MAC}_{n,d,\ell}[K]$ is the set of all rational functions which can be computed using some MAC D with size $\leq \ell$ and $\deg f_D \leq d$[3]. Here n indicates the number of inputs of D.

3 KDM Secure Scheme w.r.t Bounded Degree MAC

Cascaded Paillier ElGamal: Our scheme, called *d-cascaded Paillier ElGamal*, is computed recursively as follows. First, a "Paillier ElGamal" encryption $(e_0, c_0) = ({u_0}^{-1}, T^M v_0) \mod N^s$ of a message M is computed, where $T = 1+N$ and $(u_0, v_0) \leftarrow (g^{r_0}, h^{r_0})$. Next, the left component e_i of the ciphertext is encrypted by "Paillier ElGamal" encryption and $(e_{i+1}, c_{i+1}) = ({u_{i+1}}^{-1}, e_i v_{i+1})$ is obtained for $i = 1, \ldots, d-1$, where $(u_{i+1}, v_{i+1}) \leftarrow (g^{r_{i+1}}, h^{r_{i+1}})$. We finally let c_{d+1} be e_d. (Note that much of the encryption is not message dependent and can be performed off-line given the degree bound expected, as in ElGamal, but with much more performance gain).

The d-cascaded Paillier ElGamal encryption of message M is the tuple

$$C = (c_{d+1}, c_d, c_{d-1}, \ldots, c_0) = ({u_d}^{-1}, {u_{d-1}}^{-1} v_d, {u_{d-2}}^{-1} v_{d-1}, \ldots, T^M v_0).$$

Detailed Scheme: The detail of our scheme is as follows. Bellow, κ and ξ are security parameters and $s \geq 2$ and d are positive integers.

- $\mathsf{Setup}(1^\kappa)$: Generate the product N of two safe primes with $\lfloor \kappa/2 \rfloor$ bit lengths. Select $g \xleftarrow{\$} \mathcal{SCR}[N^s]$ randomly, and output $prm \leftarrow (s, N, g)$. (We will let T denote $1 + N$.)
- $\mathsf{Kg}(prm)$: Select $sk \leftarrow x \xleftarrow{\$} [2^\xi \cdot \lfloor N/4 \rfloor]$ randomly, compute $pk \leftarrow h \leftarrow g^x \bmod N^s$, and output (pk, sk).
- $\mathsf{Enc}_{prm}(pk, M)$ for $M \in \mathbb{Z}_{N^{s-1}}$: Select $r_0, \ldots, r_d \xleftarrow{\$} [\lfloor N/4 \rfloor]$ randomly and output $C \leftarrow (c_{d+1}, \ldots, c_0)$, where

$$c_j \leftarrow \begin{cases} T^M h^{r_0} & \bmod N^s \quad \text{if } j = 0 \\ g^{-r_{j-1}} h^{r_j} & \bmod N^s \quad \text{if } j \in \{1, \ldots, d\} \\ g^{-r_d} & \bmod N^s \quad \text{if } j = d+1. \end{cases}$$

[3] The total degrees of the polynomials may not be computable from D in polynomial time. But this fact does not become a problem in our case, because we can easily compute an upper bound of the total degrees from D.

– $\mathsf{Dec}_{prm}(sk, C)$: Parse C as $(c_{d+1}, \ldots, c_0)$ and output

$$M \leftarrow L(c_0 {c_1}^x \cdots {c_{d+1}}^{x^{d+1}} \bmod N^s).$$

Above, L is the function given in Theorem 1.

Security: We will prove the following theorem in Section 6.

Theorem 4 (KDM Security of Our Scheme w.r.t $\mathcal{MAC}_{n,d,\ell}[N^{s-1}]$). *For any polynomial d, n, and ℓ of the security parameter κ, the proposed scheme is KDM secure with respect to $\mathcal{MAC}_{n,d,\ell}[N^{s-1}]$ under the DCR assumption.*

Specifically, for any polynomials n, d, and ℓ of κ, and any polynomial time adversary A for breaking KDM security of our scheme, there exists an adversary B for breaking the DCR problem in $\mathbb{Z}_{N^s}$ satisfying

$$\begin{cases} \mathsf{Adv.KDM}_{\mathsf{A}}[\mathcal{MAC}_{n,d,\ell}[N^{s-1}], n] \leq 6\mathsf{Adv.DCR}_{\mathsf{B}} + O\left(\frac{qd}{\sqrt{N}}\right) + O\left(\frac{n}{2^{\xi}}\right), \\ t_{\mathsf{B}} \leq t_{\mathsf{A}} + O(qdE) + O(q\ell d^2\kappa^2) \end{cases}$$

Above, q is the number of queries of A, $t_{(\cdot)}$ is the number of steps of machines, and E is a full exponentiation cost in $\mathcal{QR}[N^s]$.

We can show a stronger variant of Theorem 4 where an adversary can select parameter d and ℓ *on the fly* when it makes encryption queries. (Specifically, d becomes encryption bounded and ℓ unbounded as indicated in Section 1.4.) The proof of this stronger security is similar to that of Theorem 4. We therefore omit it.

4 KDM Secure Scheme w.r.t. Fraction of Bounded Degree MACs

In this section, we give a general converter from a KDM[$\mathcal{F}$] secure scheme to a KDM[$\mathcal{Q}(\mathcal{F})$] secure scheme, where $\mathcal{F}$ is a set of polynomials over $\mathbb{Z}_K$ and $\mathcal{Q}(\mathcal{F})$ denote the set of all rational functions $f'(\vec{X})/f''(\vec{X})$ for $f, g \in \mathcal{F}$. By applying this converter to our first scheme, we can get a KDM[$\mathcal{Q}(\mathcal{MAC}_{n,d,\ell}[K])$] secure scheme.

A subtle but difficult problem in designing KDM[$\mathcal{Q}(\mathcal{F})$] secure scheme is that the denominator of $f'(\vec{sk})/f''(\vec{sk})$ can be 0 (or more generally, can be non-invertible): This becomes problem when proving security of the scheme because a simulator in the security proof (which does not know sk) cannot know whether $f''(\vec{sk})$ is invertible or not. We therefore have to design our scheme and prove the security of it such that a simulator can simulate the view of adversary even without knowing whether $f''(\vec{sk})$ is invertible.

We assume the hardness of factoring of the modulus K. Then no one can find value $a \in \mathbb{Z}_K$ which is non zero but is non-invertible. (If one can find such a,

he can factorize K by computing $\mathsf{gcd}(a, K)$.) Therefore, we can assume that the value $f''(\overrightarrow{x})$ is either invertible or 0.

We then define the function value $f'(\overrightarrow{x})/f''(\overrightarrow{x})$ with $f''(\overrightarrow{x}) = 0$ as follows, where $\mathtt{1/0}$ and $\mathtt{0/0}$ are special symbols.

$$f(\overrightarrow{x}) = \begin{cases} \mathtt{1/0} & \text{if } f''(\overrightarrow{x}) = 0 \text{ but } f'(\overrightarrow{x}) \neq 0 \\ \mathtt{0/0} & \text{if } f''(\overrightarrow{x}) = f'(\overrightarrow{x}) = 0. \end{cases}$$

Note that we are not required to consider the other case (that is, $f''(x)$ is not 0 but is not invertible) due to the above discussion.

Let $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme whose secret key and message spaces are $\mathbb{Z}_K$ for some integer K. The scheme $\overline{\mathcal{PKE}} = (\overline{\mathsf{Setup}}, \overline{\mathsf{Kg}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ converted from $\mathcal{PKE}$ is as follows.

- The message space of $\overline{\mathcal{PKE}}$ is $\mathbb{Z}_K \cup \{\mathtt{1/0}, \mathtt{0/0}\}$. Here, "$\mathtt{1/0}$" and "$\mathtt{0/0}$" are special symbols.
- $\overline{\mathsf{Setup}}(1^\kappa)$ and $\overline{\mathsf{Kg}}(prm)$: The same as $\mathsf{Setup}(1^\kappa)$ and $\mathsf{Kg}(prm)$.
- $\overline{\mathsf{Enc}}_{prm}(pk, M)$: Select $R \xleftarrow{\$} \mathbb{Z}_K^*$ randomly and set

$$(M', M'') \leftarrow \begin{cases} (MR, R) \bmod K & \text{if } M \neq \mathtt{1/0}, \mathtt{0/0}, \\ (\ R, \ \ 0) \bmod K & \text{if } M = \mathtt{1/0}. \\ (\ 0, \ \ 0) \bmod K & \text{if } M = \mathtt{0/0}. \end{cases}$$

  Compute and output

$$\bar{C} \leftarrow (C', C'') \leftarrow (\mathsf{Enc}_{prm}(pk, M'), \mathsf{Enc}_{prm}(pk, M'')).$$

- $\overline{\mathsf{Dec}}_{prm}(sk, \bar{C})$: Parse $\bar{C}$ as (C', C''), compute

$$M' \leftarrow \mathsf{Dec}_{prm}(sk, C'); \qquad M'' \leftarrow \mathsf{Dec}_{prm}(sk, C'').$$

  Output $\mathtt{1/0}$ if $M' \neq 0$ and $M'' = 0$ holds. Output $\mathtt{0/0}$ if $M' = M'' = 0$ holds. Output $M \leftarrow M'/M'' \bmod K$ otherwise.

Theorem 5. *Suppose that factoring of K is hard. Suppose the following property also: for any $f(\overrightarrow{X}) \in \mathcal{F}$ and $R \in \mathbb{Z}_K$, the function $R \cdot f(\overrightarrow{X})$ is an element of $\mathcal{F}$. Then, KDM[$\mathcal{F}$] security of $\mathcal{PKE}$ implies KDM[$\mathcal{Q}(\mathcal{F})$] security of $\overline{\mathcal{PKE}}$.*

Proof. (sketch) An adversary B for KDM[$\mathcal{F}$] security of $\mathcal{PKE}$ is constructed from an adversary A for KDM[$\mathcal{Q}(\mathcal{F})$] security of $\overline{\mathcal{PKE}}$ as follows. B takes a public parameter prm and a tuple $(pk_j)_{j\in[n]}$ of public keys as an input and passes it to A. If A makes a query $i \in [n]$ and a pair (D', D'') of descriptions of MACs, B selects $S \xleftarrow{\$} \mathbb{Z}_K^*$ randomly, sets E' and E'' to the descriptions of functions $S \cdot f_{D'}(\overrightarrow{X})$ and $S \cdot f_{D''}(\overrightarrow{X})$ respectively, makes queries (i, E') and (i, E''), gets answers C' and C'' from the challenger, and sends (C', C'') back to A as an answer to the query. If A outputs a bit b', B outputs b'.

From the hardness of the factoring of K, the values $f_{D''}(\overrightarrow{sk})$ is either invertible or equal to 0. Hence, we can consider $1/f_{D''}(\overrightarrow{sk}) \bmod K$ if $f_{D''}(\overrightarrow{sk}) \neq 0$. Therefore,

$$(S \cdot f_{D'}(\overrightarrow{sk}), S \cdot f_{D''}(\overrightarrow{sk})) = \begin{cases} (\frac{f_{D'}(\overrightarrow{sk})}{f_{D''}(\overrightarrow{sk})} \cdot R_0, R_0) & \text{if } f_{D''}(\overrightarrow{sk}) \neq 0 \\ (R_1, 0) & \text{if } f_{D''}(\overrightarrow{sk}) = 0 \text{ but } f_{D'}(\overrightarrow{sk}) \neq 0 \\ (0, 0) & \text{if } f_{D'}(\overrightarrow{sk}) = f_{D''}(\overrightarrow{sk}) = 0, \end{cases}$$

where $R_0 = S \cdot f_{D''}(\overrightarrow{sk})$ and $R_1 = S \cdot f_{D'}(\overrightarrow{sk})$.

The message which B should encrypt in the above three cases is $f_{D'}(\overrightarrow{sk})/f_{D''}(\overrightarrow{sk})$, $1/0$, and $0/0$ respectively. This means that the view of A simulated by B is identical to the actual one.

The above proof does not work well if the factoring of K is easy, because A may make query (D', D'') such that $f_{D''}(sk)$ is not 0 but non-invertible. This means that K cannot be N^{s-1} for $s \geq 3$.

5 Triple Mode Proof Framework

5.1 Overview

A triple mode proof framework is introduced to overcome the dilemma described in Section 1.5. It has three modes called *standard mode*, *fake mode*, and *hiding mode*. The standard mode is the same as the original game of KDM security. Other two modes are as follows. (See Fig.2 also.)

Dependency of Ciphertexts	*sk*	*D*	
Standard Mode	Yes.	Yes.	} Sim. knows *sk*.
Fake Mode	No.	Yes.	} Sim. does not know *sk*.
Hide Mode	No.	No.	

Fig. 2. Triple Mode Proof Framework

Fake Mode: This mode allows us to compute "fake ciphertexts" using queries (i, D) of an adversary but without using the secret keys. The fake ciphertexts should be indistinguishable from the ciphertext of the standard mode, under the condition that a simulator *knows* the secret keys.

This indistinguishability of course cannot be proved based on the hardness related to unavailability of the secret keys. Instead, we are required to prove it based on the secrecy of the *randomness of encryptions*. Showing this indistinguishability is the critical part of the proof of KDM security.

Hiding Mode: This mode enables us to compute "hiding ciphertexts" using neither the queries (i, D) of an adversary nor the secret keys. The hiding ciphertexts should be indistinguishable from the fake ones, under the assumption

that a simulator *does not know* the secret keys. Note that the simulator is not required to know the secret keys in fact, because both the fake ciphertexts and the hiding ones can be computed without using the secret keys. This indistinguishability can be shown using the standard cryptographic arguments based on the secrecy of the secret key. Since the hiding ciphertext does not depend on the query D of an adversary, KDM security, in turn, clearly holds.

5.2 Formal Description

A *triple mode proof framework* for an encryption scheme $\mathcal{PKE} = (\mathsf{Setup}, \mathsf{Kg}, \mathsf{Enc}, \mathsf{Dec})$ is a pair of *fake mode* ($\mathsf{KgFake}, \mathsf{EncFake}$) and *hiding mode* ($\mathsf{KgHide}, \mathsf{EncHide}$). The inputs and outputs of the algorithms are as follows.

- KgFake takes a public parameter prm and a natural number n as inputs and outputs n key pairs (pk_1, sk_1), ..., (pk_n, sk_n) and aux.
- KgHide takes the same inputs as KgFake and outputs keys $pk_1, \ldots, pk_n$ and aux.
- $\mathsf{EncFake}$ takes as inputs a public parameter prm, a tuple of public keys $(pk_j)_{j\in[n]}$, aux, a natural number i, and a description D of some function in $\mathcal{F}$ and outputs C.
- $\mathsf{EncHide}$ takes the same inputs as KgFake except D and outputs C.

A proof of KDM security of $\mathcal{PKE}$ w.r.t. a functions set $\mathcal{F}$ proceeds by showing that the success probability $\Pr[\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{F}, n] = 1]$ of A in $\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{F}, n]$ is the same as those in the following two games $\mathsf{GameFake}_{\mathsf{A}}[\mathcal{F}, n]$ and $\mathsf{GameHide}_{\mathsf{A}}[\mathcal{F}, n]$ but for a negligible differences.

- $\mathsf{GameFake}_{\mathsf{A}}[\mathcal{F}, n]$:

$$prm \leftarrow \mathsf{Setup}(1^\kappa), ((pk_j, sk_j)_{j\in[n]}, aux) \leftarrow \mathsf{KgFake}(prm, n),$$
$$b' \leftarrow \mathsf{A}^{\mathcal{O}'_{\mathsf{Kg}}, \mathcal{O}_{\mathsf{EncFake}_{prm}}}(prm, (pk_j)_{j\in[n]}), \text{ Output } b'$$

- $\mathsf{GameHide}_{\mathsf{A}}[\mathcal{F}, n]$:

$$prm \leftarrow \mathsf{Setup}(1^\kappa), ((pk_j)_{j\in[n]}, aux) \leftarrow \mathsf{KgHide}(prm, n),$$
$$b' \leftarrow \mathsf{A}^{\mathcal{O}'_{\mathsf{Kg}}, \mathcal{O}_{\mathsf{EncHide}_{prm}}}(prm, (pk_j)_{j\in[n]}), \text{ Output } b'$$

Above, A is allowed to make polynomial number of queries adaptively. It can send as queries bit string new to $\mathcal{O}'_{\mathsf{Kg}}$ and a tuple (i, D) to $\mathcal{O}_{\mathsf{EncFake}_{prm}}$ and to $\mathcal{O}_{\mathsf{EncHide}_{prm}}$. Here $i \in [n]$ is an integer and D is a description of some function in $\mathcal{F}$. The answers from the oracles are as follows:

- $\mathcal{O}'_{\mathsf{Kg}}(\mathsf{new})$ returns pk_i generated by KgFake (in $\mathsf{GameFake}$) or KgHide (in $\mathsf{GameHide}$).
- $\mathcal{O}_{\mathsf{EncFake}_{prm}}(i, D)$ returns $\mathsf{EncFake}_{prm}((pk_j)_{j\in[n]}, aux, i, D)$.
- $\mathcal{O}_{\mathsf{EncHide}_{prm}}(i, D)$ returns $\mathsf{EncHide}_{prm}((pk_j)_{j\in[n]}, aux, i)$.

In the final game, $\mathsf{GameHide}_{\mathsf{A}}[\mathcal{F}, n]$, ciphertexts do not depend on f queried by A any more. Hence, the following theorem holds.

Theorem 6. *Suppose that for any polynomial time adversary* A*, there exists a negligible function* $\varepsilon(\kappa)$ *such that the differences among* $\Pr[\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{F}] = 1]$, $\Pr[\mathsf{GameFake}_{\mathsf{A}}[\mathcal{F}, n] = 1]$*, and* $\Pr[\mathsf{GameHide}_{\mathsf{A}}[\mathcal{F}, n] = 1]$ *is less than* $\varepsilon(\kappa)$ *for any* $n = \mathsf{poly}(\kappa)$*, then the scheme is KDM*$[\mathcal{F}]$ *secure.*

As noted, proofs of known KDM secure scheme in the standard model [BHHO08, ACPS09, BG10] can be re-interpreted as above.

6 Security Proof of the First Scheme

6.1 Interactive Vector Lemma [BG10]

We review a lemma of [BG10] which we will use to prove KDM security of our scheme. Let $\mathsf{Gen}(1^\kappa)$ be a generator which outputs the product N of two safe primes with the same bit lengths. Let A be a polynomial time adversary, b be a bit, and $s \geq 2$ be an integer. Define game IV_1 and IV_2 as follows.

- $\mathsf{IV}_1 : N \leftarrow \mathsf{Gen}(1^\kappa), g \xleftarrow{\$} \mathcal{SCR}[N^s], \quad b' \leftarrow \mathsf{A}^{\mathcal{O}_b}(s, N, g), \quad$ Output b'.
- $\mathsf{IV}_2 : N \leftarrow \mathsf{Gen}(1^\kappa),\ g, h \xleftarrow{\$} \mathcal{SCR}[N^s],\ b' \leftarrow \mathsf{A}^{\bar{\mathcal{O}}_b}(s, N, g, h), \quad$ Output b'.

In the above two games, A is allowed to make polynomial number queries. In IV_1, A can send an element δ of $\mathbb{Z}_{N^{s-1}}$ as a query. $\mathcal{O}_b(\delta)$ then selects $r \xleftarrow{\$} [\lfloor N/4 \rfloor]$ randomly and returns

$$u^* \leftarrow \begin{cases} T^\delta g^r \bmod N^s & \text{if } b = 1, \\ g^r \bmod N^s & \text{if } b = 0. \end{cases}$$

On the other hand, A in IV_2 can send an element $(\delta, \bar{\delta})$ of ${\mathbb{Z}_{N^{s-1}}}^2$ as a query. $\bar{\mathcal{O}}_b(\delta, \bar{\delta})$ then selects $r \xleftarrow{\$} [\lfloor N/4 \rfloor]$ randomly and returns

$$(u^*, \bar{u}^*) \leftarrow \begin{cases} (T^\delta g^r, T^{\bar{\delta}} h^r) \bmod N^s & \text{if } b = 1, \\ (g^r, h^r) \bmod N^s & \text{if } b = 0. \end{cases}$$

For $k = 1, 2$, the advantage of A in IV_k is defined to be

$$\mathsf{Adv}.\mathsf{IV}_k[\mathsf{A}] = |\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0]|.$$

Lemma 1 ((DCR-based) Interactive Vector Lemma for $k = 1, 2$, Full paper of [BG10]). *For* $k = 1, 2$*, no polynomial time adversary can have non-negligible advantage in* IV_k *under the DCR assumption.*

Our definition of game IV_k is slightly different from those of [BG10]: The original game takes the randomness r of g^r not from $[\lfloor N/4 \rfloor]$ but $[T^2]$ for some fixed value $T \geq N^s$. This difference is not essential, because the randomness r of the original game is taken from $[T^2]$ in order to be ensured that the distribution of g^r is statistically close to the uniform distribution on $\mathcal{SCR}[N^s]$. It can be shown that the same thing holds even if r is selected from $[\lfloor N/4 \rfloor]$.

6.2 The Proof When the Number n of Keys Is 1

Before proving the security of our scheme, the game $\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{MAC}]$ for our scheme with $n = 1$ is reviewed. (Here we simply write $\mathcal{MAC}$ for $\mathcal{MAC}_{n,d,\ell}[N^{s-1}]$.) An adversary A of this game takes a public parameter $prm = (s, N, g)$ and one public key $pk = h = g^x \bmod N^s$ as inputs. Whenever A sends as a query the "description" of a function in $\mathcal{F}$, namely an MAC D, the challenger sends back

$$C = ({u_d}^{-1}, {u_{d-1}}^{-1} v_d, \ldots, {u_0}^{-1} v_1, T^{f_D(x)} v_0),$$

where f_D is the function corresponding to D and $(u_k, v_k) = (g^{r_k}, h^{r_k})$ for $r_k \leftarrow \lfloor N/4 \rfloor$. Since the number of keys is 1, the polynomial f_D can be written as $f_D(Y) = \sum_j a_j Y^j \bmod N^s$ for some $(a_j)_{j \in [0..d]}$. A finally outputs a bit b'.

The security is proved based on the framework of triple mode proof framework of Section 5. The algorithms KgFake and $\mathsf{EncFake}$ are defined as follows.

- $\mathsf{KgFake}(prm)$: Same algorithm as Kg, except that it sets aux to the null string.
- $\mathsf{EncFake}(prm, pk, aux, D)$: Parse prm and pk as (s, N, g) and h. Take $r_k \xleftarrow{\$} \lfloor N/4 \rfloor$ and compute $(u_k, v_k) \leftarrow (g^{r_k}, h^{r_k})$ for $k \in [d]$. Let $f_D(Y) = \sum_j a_j Y^j \bmod N^s$. Compute and output a "fake ciphertext"

$$C_{\mathsf{Fake}} = ({u_d}^{-1}, T^{a_d} {u_{d-1}}^{-1} v_d, \ldots, T^{a_1} {u_0}^{-1} v_1, T^{a_0} v_0).$$

We show that $|\Pr[\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{MAC}, 1] = 1] - \Pr[\mathsf{GameFake}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]|$ is negligible. To this end, an adversary B for the game IV_k with $k = 1$ is constructed as follows. B takes an input (s, N, g), selects $x \xleftarrow{\$} [2^\xi \cdot \lfloor N/4 \rfloor]$ randomly, and feeds $prm \leftarrow (s, N, g)$ and $pk \leftarrow h \leftarrow g^x$ to A. If A makes a query D, B computes $(a_j)_{j \in [0..d]}$ satisfying $f_D(Y) = \sum_j a_j Y^j \bmod N^s$. Note that B can compute it in polynomial time from D. B sets

$$\delta_j = -\sum_{k=j+1}^{d} a_k x^{k-(j+1)},$$

makes queries $\delta_0, \ldots, \delta_{d-1}$ and gets corresponding answers $u^*_0, \ldots, u^*_{d-1}$ (where u^*_j is $T^{\delta_j} g^{r_j}$ or g^{r_j}). B then selects $r_d \leftarrow [\lfloor N/4 \rfloor]$, computes $u^*_d \leftarrow g^{r_d}$, computes $v^*_j \leftarrow (u^*_j)^x$ for $j = 0, \ldots, d$, and sends back to A

$$C^* = ((u^*_d)^{-1}, (u^*_{d-1})^{-1} v^*_d, \ldots, (u^*_0)^{-1} v^*_1, T^{f_D(x)} v^*_0).$$

If A outputs a bit b', B outputs it and terminates. From the definition of B, the difference between two probabilities $\Pr[\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]$ and $\Pr[\mathsf{GameFake}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]$ is negligible.

The algorithms in $\mathsf{GameHide}$, that is KgHide and $\mathsf{EncHide}$, are defined as follows.

- $\mathsf{KgHide}(prm)$: Take $pk \leftarrow h \xleftarrow{\$} \mathcal{QR}[N^s]$ and outputs it. (It sets aux to the null string.)

– $\mathsf{EncHide}(prm, pk, aux)$: Parse prm and pk as (s, N, g) and $pk = h$. Take $r_k \xleftarrow{\$} \lfloor N/4 \rfloor$ and compute $(u_k, v_k) \leftarrow (g^{r_k}, h^{r_k})$ for $k \in [d]$. Compute and output a "hiding ciphertext"

$$C_{\mathsf{Hide}} = ({u_d}^{-1}, {u_{d-1}}^{-1} v_d, \ldots, {u_0}^{-1} v_1, v_0).$$

Namely, $\mathsf{EncHide}$ outputs $\mathsf{Enc}_{prm}(pk, 0)$.

We show that $|\Pr[\mathsf{GameFake}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1] - \Pr[\mathsf{GameHide}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]|$ is negligible. To this end, an adversary B for IV_k with $k = 2$ is constructed as follows. B takes an input (s, N, g, h), and feeds $prm \leftarrow (s, N, g)$ and $pk \leftarrow h$ to A. If A makes a query D, let $f_D(Y) = \sum_j a_j Y^j \bmod N^s$. B then sets

$$(\delta_j, \bar{\delta}_j) = (0, a_j)$$

makes queries $(\delta_0, \bar{\delta}_0), \ldots, (\delta_d, \bar{\delta}_d)$ and gets answers $(u_1^*, v_1^*), \ldots, (u_d^*, v_d^*)$ (where (u_j^*, v_j^*) is $(T^0 g^{r_j}, T^{a_j} h^{r_j})$ or (g^{r_j}, h^{r_j})) and sends back to A

$$C^* = ((u_d^*)^{-1}, (u_{d-1}^*)^{-1} v_d^*, \ldots, (u_0^*)^{-1} v_1^*, v_0^*).$$

If A outputs a bit b', B outputs it and terminates. From the definition of B, the difference between two probabilities $\Pr[\mathsf{GameFake}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]$ and $\Pr[\mathsf{GameHide}_{\mathsf{A}}[\mathcal{MAC}, 1] = 1]$ is negligible. From Theorem 6, our scheme is KDM secure.

6.3 The Idea Behind the Proof of the General Case

Due to the lack of space, we only present the proof idea. It proceeds in a similar way to the proof of Section 6.2, except that we make KgFake and $\mathsf{EncFake}$ "reduce" n secrets $(sk_j)_{j \in [n]}$ to only one secret μ.

Specifically, KgFake for this proof takes $prm = (s, N, g)$ and the number n of keys as inputs, selects $\mu \xleftarrow{\$} [\lfloor N/4 \rfloor]$ and $\alpha_1, \ldots, \alpha_n \xleftarrow{\$} [2^\xi \cdot \lfloor N/4 \rfloor]$ randomly, and outputs

$$sk_j \leftarrow x_j \leftarrow \mu + \alpha_j, \qquad pk_j \leftarrow h_j \leftarrow g^{x_j} \text{ for } j \in [n], \qquad aux \leftarrow (\alpha_j)_{j \in [n]}$$

In other words, $(sk_j)_{j \in [n]}$ is computed from only one "secret" μ. The proof is therefore reduced to the case where the number of secret is 1, in some sense.

The description of $\mathsf{EncFake}$ is also changed, in order to become consistent with the new KgFake. Specifically, $\mathsf{EncFake}$ takes $prm = (s, N, g)$, $(pk_j)_{j \in [n]} = (h_j)_{j \in [n]}$, and a query (i, D) of an adversary and computes $(a_j)_{j \in [0..d]} \leftarrow \mathsf{Coeff}_{prm}(aux, i, D)$. Here $(a_j)_{j \in [0..d]} = \mathsf{Coeff}_{prm}(aux, i, D)$ is the tuple of $\mathbb{Z}_{N^{s-1}}$ satisfying the following equations about polynomials of a variable Y. Below, d is the total degree of f_D.

$$f_D(Y + \alpha_1, \ldots, Y + \alpha_n) = \sum_{j=0}^{d} a_j (Y + \alpha_i)^j \bmod N^{s-1}.$$

$\mathsf{EncFake}$ then outputs "fake encryption"

$$C_{\mathsf{Fake}} = ({u_d}^{-1}, T^{a_d} {u_{d-1}}^{-1} v_d, \ldots, T^{a_1} {u_0}^{-1} v_1, T^{a_0} v_0),$$

where $(u_k, v_k) = (g^{r_k}, {h_i}^{r_k})$ for $r_k \xleftarrow{\$} \lfloor N/4 \rfloor$. (We stress that v_k is computed using the i-th public key h_i where i is a part of the query (i, D).)

Above, $\mathsf{EncFake}$ is polynomial time algorithm due to the following lemma:

Lemma 2. *Given $aux = (\alpha_j)_{j \in [n]}$, $i \in [n]$, and an MAC D, $\mathsf{Coeff}_{prm}(aux, i, D)$ can be computed in polynomial time.*

KgHide and $\mathsf{EncHide}$ can be constructed similarly. Proofs of indistinguishabilities of $\mathsf{GameKDM}^1_{\mathsf{A}}[\mathcal{MAC}, n]$, $\mathsf{GameFake}_{\mathsf{A}}[\mathcal{MAC}, n]$, and $\mathsf{GameHide}_{\mathsf{A}}[\mathcal{MAC}, n]$ are similar to those of Section 6.2 as well.

Acknowledgments. We thank Zvika Brakerski and Yevgeniy Vahlis for several illuminating discussions. We also thank Zvika, Shafi Goldwasser, Yael Tauman Kalai, and anonymous Eurocrypt reviewers for helpful comments regarding the presentation of the paper.

References

[AR00] Abadi, M., Rogaway, P.: Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption). In: Watanabe, O., Hagiya, M., Ito, T., van Leeuwen, J., Mosses, P.D. (eds.) TCS 2000. LNCS, vol. 1872, pp. 3–22. Springer, Heidelberg (2000); J. Cryptology 15(2), 103–127 (2002), J. Cryptology 20(3), 395 (2007)

[ABBC10] Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic Agility and Its Relation to Circular Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010)

[ABHS05] Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness of Formal Encryption in the Presence of Key-Cycles. In: di Vimercati, S.d.C., Syverson, P.F., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 374–396. Springer, Heidelberg (2005)

[A11] Applebaum, B.: Key-Dependent Message Security: Generic Amplification and Completeness Theorems. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 506–525. Springer, Heidelberg (2011)

[ACPS09] Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast Cryptographic Primitives and Circular-Secure Encryption Based on Hard Learning Problems. In: C 2009, pp. 595–618 (2009)

[BDU08] 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)

[BPS08] Backes, M., Pfitzmann, B., Scedrov, A.: Key-dependent message security under active attacks - BRSIM/UC-soundness of Dolev-Yao-style encryption with key cycles. In: CSF 2007, pp. 112–124 (2008); Journal of Computer Security 16(5), 497–530 (2008)

[BHHI10] Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded Key-Dependent Message Security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010)

[BRS02] Black, J., Rogaway, P., Shrimpton, T.: Encryption-Scheme Security in the Presence of Key-Dependent Messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)

[BG10] Brakerski, Z., Goldwasser, S.: Circular and Leakage Resilient Public-Key Encryption Under Subgroup Indistinguishability (or: Quadratic Residuosity Strikes Back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010), Full paper is available at eprint 2010/226

[BGK09] Brakerski, Z., Goldwasser, S., Kalai, Y.: Circular-Secure Encryption Beyond Affine Functions. e-print. 2009/511

[BHHO08] 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)

[BV98] 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)

[CCS09] Camenisch, J., Chandran, N., Shoup, V.: A Public Key Encryption Scheme Secure against Key Dependent Chosen Plaintext and Adaptive Chosen Ciphertext Attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)

[CL01] Camenisch, J., 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)

[CKVW10] Canetti, R., Tauman Kalai, Y., Varia, M., Wichs, D.: On Symmetric Encryption and Point Obfuscation. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 52–71. Springer, Heidelberg (2010)

[DJ01] Damgård, I., Jurik, M.: A Generalization, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

[G09] Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC 2009, pp. 169–178 (2009)

[GH10] Green, M., Hohenberger, S.: CPA and CCA-Secure Encryption Systems that are not 2-Circular Secure. e-print. 2010/144

[HH09] 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)

[HK07] Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: ACM CCS 2007, pp. 466–475 (2007)

[HU08] 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)

[KTY09] Kiayias, A., Tsiounis, Y., Yung, M.: Group Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)

[MTY11] Malkin, T., Teranishi, I., Yung, M.: Key Dependent Message Security: Recent Results and Applications. In: ACM CODASPY (2011)

[P99] 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)

Key-Dependent Message Security: Generic Amplification and Completeness

Benny Applebaum*

School of Electrical Engineering, Tel-Aviv University
benny.applebaum@gmail.com

Abstract. Key-dependent message (KDM) secure encryption schemes provide secrecy even when the attacker sees encryptions of messages related to the secret-key sk. Namely, the scheme should remain secure even when messages of the form $f(\mathsf{sk})$ are encrypted, where f is taken from some function class $\mathcal{F}$. A KDM *amplification* procedure takes an encryption scheme which satisfies $\mathcal{F}$-KDM security and boost it into a $\mathcal{G}$-KDM secure scheme, where the function class $\mathcal{G}$ should be richer than $\mathcal{F}$. It was recently shown by Brakerski et al. (TCC 2011) and Barak et al. (EUROCRYPT 2010), that a strong form of amplification is possible, provided that the underlying encryption scheme satisfies some special additional properties.

In this work, we prove the first *generic* KDM amplification theorem which relies solely on the KDM security of the underlying scheme without making any other assumptions. Specifically, we show that an elementary form of KDM security against functions in which each output bit either copies or flips a single bit of the key (aka *projections*) can be amplified into KDM security with respect to any function family that can be computed in arbitrary fixed polynomial-time. Furthermore, our amplification theorem and its proof are insensitive to the exact setting of KDM security, and they hold in the presence of multiple-keys and in the symmetric-key/public-key and the CPA/CCA cases. As a result, we can amplify the security of all known KDM constructions, including ones that could not be amplified before.

Finally, we study the minimal conditions under which full-KDM security (with respect to all functions) can be achieved. We show that under strong notion of KDM security, the existence of cyclic-secure fully-homomorphic encryption is not only sufficient for full-KDM security, as shown by Barak et al., but also necessary. On the other hand, we observe that for standard KDM security, this condition can be relaxed by adopting Gentry's bootstrapping technique (STOC 2009) to the KDM setting.

1 Introduction

The study of secure encryption scheme is perhaps the most central subject in cryptography. Since the discovery of semantic security [24] till the formulation

* Work done in part a postdoc at the Weizmann Institute of Science, supported by Alon and Koshland Fellowships.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 527–546, 2011.

of CCA-security [31,33,18], modern cryptography has successfully developed increasingly stronger notions of security providing secrecy in highly adversarial settings. Still, all these strong notions of security guarantee secrecy only as long as the encrypted messages are independent of the secret key. This limitation dates back to the seminal work of Goldwasser and Micali [24] who observed that semantic security may not hold if the adversary gets to see an encryption of the secret key. For many years, such usage scenarios were considered as "security bugs" that should be prevented by system designers.

A decade ago, the assumption of independency between the secret key and the encrypted data was challenged by Camenisch and Lysyanskaya [16] and Black et al. [11]. Specifically, Camenisch and Lysyanskaya considered schemes that remain secure under a "key cycle" usage, where we have t keys organized in a cycle and each key is encrypted under its left neighbor. A generalization of this notion, called *key-dependent message* (KDM) security, was suggested by Black et al. Informally, an encryption is $\mathsf{KDM}^{(t)}$ secure with respect to a function class $\mathcal{F}$ if security holds even when the adversary can ask for an encryption of the message $M = f(\mathsf{sk}_1, \ldots, \mathsf{sk}_t)$ under the i-th public-key, where $\mathsf{sk}_1, \ldots, \mathsf{sk}_t$ are the secret keys present in the system and f is an arbitrary function in $\mathcal{F}$. This notion of security implies cyclic-security if $\mathcal{F}$ is expressive enough (e.g., contains all "selector" functions), and it becomes strictly stronger when the function class $\mathcal{F}$ grows. Hence, one would like to achieve KDM security while making the function class $\mathcal{F}$ as large as possible.

The notion of KDM security was extensively studied in the past few years in several flavors including the symmetric/public-key and the CPA/CCA settings [16,11,26,9,12,15,8,27,25,5,14,2,10,13]. These works were motivated by the fundamental nature of the question as well as by concrete applications including encrypted storage systems (e.g., BitLocker [12]), anonymous credentials [16], and realization of security proofs at the framework of axiomatic security [1,11,3]. (See [12] for more motivations and details.)

Although much is known today about KDM security both on the positive and negative sides, it is still unclear whether a standard encryption scheme can be transformed into a scheme which provides $\mathsf{KDM}^{(t)}$ security, even with respect to a single key (i.e., $t = 1$) and simple non-trivial function families (e.g., selectors)[1]. Hence, it is natural to move forward and explore the possibility of building strong KDM security given a weak form of KDM security as a primitive. This makes sense as today, following the seminal work of Boneh et al. [12] and its follow-ups [15,5,13], it is known that a basic form of KDM security (with respect to the family of "affine functions") can be achieved in several settings under various concrete cryptographic assumptions. Therefore, following [14] we ask:

> Is there a *generic* transformation which *amplifies* KDM security from a weak family of functions $\mathcal{F}$ to a larger family of functions $\mathcal{G}$?

[1] It is known that KDM security with respect to sufficiently rich families of functions cannot be based on standard assumptions via fully black-box reductions [25]. However, this impossibility result (and its extension in [10]) does not hold for simple function class (e.g., projections).

The two main features of such a procedure are *generality* – the transformation should work with any scheme which satisfies $\mathcal{F}$-KDM security without relying on any other additional property – and large *amplification gap* – ideally, $\mathcal{F}$ is a very simple function class whereas $\mathcal{G}$ is as rich as possible. The question of KDM amplification was recently addressed by Brakerski et al. [14] and Barak et al. [10], who made an important progress by showing how to amplify the KDM security of several existing schemes. While these works achieve relatively large amplification gap, they fall short of providing full generality as they strongly rely on additional properties of the underlying scheme (i.e., *simulatable*-KDM security and *entropic*-KDM security – to be defined later). As a concrete example, it is unknown how to use any of these techniques to amplify the KDM-security of the symmetric-key encryption scheme of [5] which is based on the Learning Parity With Noise (LPN) assumption. (See Section 1.3 for more details about these works and their relation to our approach).

1.1 Our Results

We give an affirmative answer to the above question by providing the first generic KDM amplification procedure. In particular, we consider the *projection* function class of all functions $f : (\mathsf{sk}_1, \ldots, \mathsf{sk}_t) \mapsto v$ in which each output bit depends on (at most) a single bit of the input. Namely, each output bit v_j is either fixed to a constant or copies/flips an original bit of one of the keys. We show that this elementary function family is *complete* in the following sense:

Theorem 1 (Completeness of projections, Informal). *Let $\mathcal{G}$ be any function family which can be computed in some fixed polynomial time. Then, any encryption scheme which satisfies $\mathsf{KDM}^{(t)}$ security with respect to projections can be transformed into a new encryption scheme which is $\mathsf{KDM}^{(t)}$-secure with respect to $\mathcal{G}$.*

Generality. Theorem 1 assumes nothing but KDM security regarding the underlying scheme. Furthermore, the theorem (and its surprisingly simple proof) is insensitive to the exact setting of KDM security: it holds for any number of keys (t), and in both symmetric-key/public-key and CPA/CCA settings. In all these cases, the new scheme is proven to be secure exactly in the same setting as the original scheme. This allows us, for example, to amplify the security of the affine-KDM secure scheme of [5], and obtain the first symmetric-key encryption scheme with strong KDM security based on the LPN assumption.

Large gap. Theorem 1 provides a large amplification gap. In fact, this gap can be further expanded as follows. First, we can achieve *length-dependent* KDM security [10], which means that the target family $\mathcal{G}$ can be taken to be the family of all polynomial-size circuits whose size grows with their input and output lengths via a fixed polynomial rate (e.g., the circuit size is quadratic in the input and output lengths). This family is very powerful and it was shown to be rich

enough for most known applications of KDM security [10][2]. (See Section 3 for details.) In addition, in the case of CPA security (both in the public-key and symmetric-key settings), we can weaken the requirement from the underlying scheme and ask for KDM security with respect to projections with a *single output*: namely, all Boolean functions $f(\mathsf{sk}_1, \ldots, \mathsf{sk}_t) \mapsto b$ which output a single bit of one of the keys, or its negation. This can be extended to the CCA setting via the transformations of [9,15] (though in the public-key setting one has to employ, in addition, non-interactive zero-knowledge proofs).

The relaxation to single-output projections also enables a liberal interface to which we can easily plug previous constructions. For example, one can instantiate our reduction with schemes that enjoy KDM security with respect to affine functions, while almost ignoring technical details such as the underlying field and its representation. (These details required some effort in previous works. See the appendices in [14,10,13].) This, together with the simple proof of our main theorem, allows to simplify the proofs of [10,13] for the existence of length-dependent KDM secure encryption scheme under the Decisional Diffie-Hellman (DDH) assumption [12], the Learning With Errors assumptions (LWE) [5], and the Quadratic Residuosity (QR) assumption [13].

Given this completeness theorem, the current status of KDM security resembles the status of other "complete" primitives in cryptography such as one-way functions or oblivious transfer [32,19]: We do not know how to build these primitives from generic weaker assumptions, however, any instantiation of them suffices for an entire world of applications (i.e., all symmetric-key primitives in the case of one-way functions, and generic secure-computation in the case of oblivious transfer, cf. [22,23]).

Beyond length-dependent security. Although length-dependent KDM security seems to suffice for most applications, one can strive for an even stronger notion of security in which the KDM function class contains all functions (or equivalently all functions computable by circuits of *arbitrary* polynomial size). It is somewhat likely that any length-dependent secure scheme actually achieves *full-KDM* security (see the discussion in [10]). Still, one may want to construct such a scheme in a provably secure way. As a basic feasibility result, it was shown in [10] that any fully homomorphic encryption scheme [20] which allows to encrypt the secret-key (i.e., "cyclic-secure") is also full-KDM secure. In light of the small number of FHE candidates [20,17], and our little understanding of this notion, one may ask whether it is possible to relax this requirement and achieve full-KDM security under weaker assumptions.

We make two simple observations regarding this question. First, we consider the case of simulatable KDM security [10], in which it should be possible to simulate an encryption of $f(\mathsf{sk})$ given only the corresponding public-key in a way that remains indistinguishable even to someone who knows the secret-key.

[2] Most of the statements in [10] refer to the slightly weaker notion of *Bounded KDM security* in which the circuit size grows only as a function of the input via a fixed polynomial rate. However, as observed in [10, Sec. 6] their construction actually satisfies the stronger definition of *length-dependent* KDM security.

We show that in this setting the two notions: circular-secure FHE and full-KDM are equivalent. Hence, achieving full-KDM security under a relaxed assumption requires to use non-simulatable constructions.

Our second observation asserts that the bootstrapping technique of Gentry [20] can be used in the KDM setting as well (even for the case of non-simulatable constructions). That is, if one can construct an encryption scheme which guarantees KDM security with respect to circuits whose depth is only slightly larger than the depth of the decryption algorithm, then this scheme is actually fully KDM secure. Unfortunately, all known amplification techniques [10,14] including the ones in this paper, amplify KDM security at the cost of making the decryption algorithm "deeper". Still, we view this observation as an interesting direction for future research.

1.2 Our Techniques

To formalize the question of KDM amplification, we define the notion of *reduction* between KDM function families $\mathcal{G} \leq_{\mathsf{KDM}} \mathcal{F}$ which means that any scheme that provides KDM security with respect to $\mathcal{F}$ can be transformed (via a fully black-box reduction) to a new scheme that satisfies KDM security with respect to $\mathcal{G}$. We describe a novel way to derive such KDM reductions based on the machinery of *randomized encoding* of functions [29,7]. Before we explain this notion, let us start with the simpler case of *deterministic encoding*.

Say that a function f deterministically encodes a function g if for every x the output of $f(x)$ "encodes" the output of $g(x)$ in the sense that $g(x)$ can be efficiently computed based on $f(x)$ and vice versa. That is, there are two efficiently computable mappings S and R such that $S(g(x)) = f(x)$, and $R(f(x)) = g(x)$. Suppose that we are given a scheme which provides KDM security with respect to the encoding f, and we would like to immunize it against the function g. This can be easily achieved by modifying the encryption scheme as follows: to encrypt a message M we first translate it into the f-encoding by computing $S(M)$, and then encrypt the result under the original encryption scheme. Decryption is done by applying the original decryption algorithm, and then applying the recovery algorithm R to translate the result back to its original form. Observe that an encryption of $g(\mathsf{sk})$ in the new scheme is the same as an encryption of $S(g(\mathsf{sk})) = f(\mathsf{sk})$ under the original scheme. Hence, the KDM security of the new scheme with respect to g reduces to the KDM security of the original scheme with respect to f.

This simple idea provides a direct reduction with very nice structure: any KDM query for the new scheme is translated into a single KDM query for the original scheme. This simple single-query-to-single-query translation leads to high level of generality: the transformation is insensitive to the exact KDM setting (symmetric-key/public-key and CPA/CCA), to the number of keys, and it can be used with respect to large function families $\mathcal{G}$ and $\mathcal{F}$ as long as every function in $\mathcal{G}$ is encoded by some function in $\mathcal{F}$ via a pair of universal mappings S and R. On the down side, one may complain that security was not really *amplified*, as the function g and its encoding f are essentially equivalent. It turns out that this drawback can be easily fixed by letting f be a *randomized* encoding of g.

In the case of randomized encoding (RE), the function $f(x;r)$ depends not only on x but also on an additional random input r. For every fixed x, the output of $f(x;r)$ is now viewed as a distribution (induced by a random choice of r) which should encode the value of $g(x)$. Namely, there are two efficiently computable randomized mappings S and R such that for every x: (1) the distribution $S(g(x))$ is indistinguishable from $f(x;r)$, and (2) with high probability over the choice of r (or even with probability one) $R(f(x;r)) = g(x)$. One can view these conditions as saying that $g(x)$ is encoded by a *collection* of functions $\{f_r(x)\}_r$, where $f_r(x) = f(x;r)$.

Now suppose that our scheme is KDM secure with respect to the family $\{f_r(x)\}_r$, then we can apply the above approach and get a scheme which satisfies KDM security with respect to g. The only difference is that now the message preprocessing step is randomized: To encrypt a message M first encode it by the randomized mapping $S(M)$, and then use the original encryption function. The security reduction is essentially the same except that a KDM query for g in the new scheme is emulated by an old KDM query for a *randomly chosen* function f_r. This idea can be easily extended to the case where all functions in $\mathcal{G}$ are encoded by functions in $\mathcal{F}$:

Theorem 2 (Informal). *If $\mathcal{F}$ is an RE of $\mathcal{G}$, then $\mathcal{G} \leq_{\mathsf{KDM}} \mathcal{F}$.*

The crux of this theorem, is that, unlike deterministic encoding, randomized encoding can represent complicated functions by collections of very simple functions [29,30,7,6]. Specifically, by combining the above theorem with the REs of [6], which, in turn, are based on Yao's garbled circuit [34], we obtain our main results (Thm. 1).

1.3 Comparison with BGK and BHHI

Our techniques are inspired by both [14] (BGK) and [10] (BHHI). We believe that our approach inherits the positive features of each of these works, and sheds new light on the way they relate to each other. Let us review the main ideas behind these constructions and explain how they compare to our solution.

The BGK reduction. The starting point in [14] is an encryption scheme which satisfies entropic KDM security with respect to $\mathcal{F}$. Roughly speaking, this means that KDM security should hold not only when sk is chosen uniformly from the key space $\mathcal{K} = \{0,1\}^k$ but also when it is chosen uniformly from a smaller domain $\mathcal{K}'$, e.g., $\mathcal{K}' = \{0,1\}^{k^\varepsilon}$. By relying on this notion, BGK shows that for every efficiently computable injective mapping $\alpha : \mathcal{K}' \to \mathcal{K}$, one can amplify security from $\mathcal{F}$ to the class $\mathcal{F} \circ \alpha$, i.e., with respect to functions $f(\alpha(\mathsf{sk}))$ for every $f \in \mathcal{F}$. The idea is to choose the key sk' from $\mathcal{K}'$ and employ the original scheme with the key $\mathsf{sk} = \alpha(\mathsf{sk}')$. This allows to translate a KDM query $f(\alpha(\mathsf{sk}'))$ for the new scheme into an entropic-KDM query $f(\mathsf{sk})$ for the old scheme.

The deterministic encoding (DE) approach is inspired by the BGK approach, and can be seen as a complementary solution. BGK extends a function $f : \mathcal{K} \to \mathcal{M}$ to $f \circ \alpha : \mathcal{K}' \to \mathcal{M}$ by shrinking the key space (from $\mathcal{K}$ to $\mathcal{K}'$), whereas in the

DE approach $f : \mathcal{K} \to \mathcal{M}$ is extended to $R \circ f : \mathcal{K} \to \mathcal{M}'$ by padding messages which effectively shrinks the message space (from $\mathcal{M}$ to $\mathcal{M}' = R(\mathcal{M})$).

As a result BGK enjoys a similar attractive security reduction with single-query-to-single-query translation. This leads to flexibility with respect to the KDM *setting*. Indeed, although the BGK approach is not fully general due to its use of entropic-KDM security (a notion which seems stronger than standard KDM security), it immediately generalizes to the CCA and the symmetric-key settings, as long as the underlying scheme provides entropic-KDM security.

It should be mentioned that in our approach the amplification is achieved by modifying the encryption algorithm, rather than the key-generation algorithm as in BGK. This minor difference turns to have a considerable effect on the amplification-gap. First, it allows to use fresh randomness in every application of the encryption algorithm, and so the linkage between functions in $\mathcal{G}$ to functions in $\mathcal{F}$ can be *randomized*. Indeed, this is exactly what allows us to exploit the power of randomized encoding. In contrast, the BGK approach tweaks the key-generation algorithm and so the relation between $\mathcal{G}$ to $\mathcal{F}$ is bounded to be deterministic. In addition, since our modification happens in the encryption (and decryption) phases, we can let the function class $\mathcal{G}$ grow not only with the security parameter but also with the size of the messages. This leads to the strong notion of length-dependent security, and in addition allows to achieve $\mathsf{KDM}^{(t)}$ where the number of keys t grows both with the message length and the security parameter.

In contrast, the family $\mathcal{G}$ of BGK cannot grow with the message length, and it can only contain a polynomial number of functions. This limitation prevents it from being used in applications which require KDM security wrt larger functions classes (e.g., secure realization of symbolic protocols with axiomatic proofs of security). Furthermore, amplification for large number of keys can be achieved only at the expense of putting more restrictions on the underlying scheme (i.e., simulatable KDM security). On the other hand, assuming these additional properties, the BGK approach can get $\mathsf{KDM}^{(t)}$ for arbitrary unbounded t with respect to some concrete function families (e.g., constant degree polynomials), whereas in our approach t is always bounded by some fixed polynomial (in the security parameter and message length)[3]. Finally, it is important to mention that the BGK reduction treats $\mathcal{G}$ in a black-box way, while the randomized encoding approach treats this class in a non-black-box way.

The BHHI reduction. The BHHI approach relies on a novel connection between homomorphic encryptions and KDM security. First, it is observed that in order to obtain KDM security with respect to $\mathcal{G}$ it suffices to construct a scheme

[3] In fact, we can achieve a slightly stronger notion. Assuming that the underlying scheme satisfies $\mathsf{KDM}^{(t)}$ security for arbitrary t's (as in [12,5]), we get a $\mathsf{KDM}^{(t)}$ secure scheme where there exists an unbounded number of keys in the system, but the arity of the KDM functions available to the adversary is polynomially bounded (in the security parameter and message length). Still, these functions can be applied to arbitrary subsets of the keys.

which provides both cyclic-security (i.e., KDM security with respect to the identity function) and homomorphism with respect to a function family $\mathcal{G}$, i.e., it should be possible to convert a ciphertext $C = \mathsf{E}_{\mathsf{pk}}(M)$ into $C' = \mathsf{E}_{\mathsf{pk}}(g(M))$ for every $g \in \mathcal{G}$. Indeed, the homomorphism property can be used to convert a ciphertext $\mathsf{E}_{\mathsf{pk}}(\mathsf{sk})$ into the ciphertext $\mathsf{E}_{\mathsf{pk}}(g(\mathsf{sk}))$, and so cyclic-security is amplified to $\mathcal{G}$-KDM security.

BHHI construct such an encryption scheme by combining a two-party secure computation protocol with two messages (i.e., based on Yao's garbled circuit [34]) with a strong version of oblivious transfer which satisfies an additional *cyclic-security* property. The latter primitive is referred to as *targeted encryption* (TE). The basic idea is to view the homomorphic property as a secure-computation task in which the first party holds the message M and the second party holds the function g. The cyclic nature of the TE primitive allows to implement this homomorphism even when the input M is the secret-key. Finally, BHHI show that TE can be constructed based on affine-KDM secure encryption scheme which satisfies a strong notion of simulation: There exists a simulator which given the public-key pk can simulate a ciphertext $\mathsf{E}_{\mathsf{pk}}(g(\mathsf{sk}))$ in a way which is indistinguishable even for someone who holds the secret-key.

The BHHI construction seems conceptually different from our RE approach (i.e., homomorphism vs. encoding). Moreover, the construction itself is not only syntactically different, but also relies on different building blocks (e.g., TE). Still, the RE construction shares an important idea with BHHI: The use of secure-computation techniques. It is well known that REs are closely related to secure multiparty-computation (MPC) protocols, and, indeed, the role of REs in our reduction resembles the role of MPC in BHHI. In both solutions at some point the security reduction applies the RE/MPC to the function g in $\mathcal{G}$. Furthermore, both works achieve strong KDM security by instantiating the RE/MPC with Yao's garbled circuit (GC) — a tool which leads to both stand-alone RE construction [6] and, when equipped with an OT, to a two-party secure-computation protocol.

It should be emphasized, however, that the actual constructions differ in some important aspects. While we essentially encrypt the whole GC-based encoding under the underlying KDM encryption scheme, BHHI tweak the GC protocol with a cyclic-secure OT (i.e., TE). Pictorially, our underlying KDM-secure scheme "wraps" the GC encoding, whereas in BHHI the KDM-secure primitive is "planted inside" the GC protocol. This difference affects both generality and simplicity as follows.

First, BHHI are forced to implement a KDM-secure OT, a primitive which seems much stronger than standard KDM secure encryption schemes. For example, KDM-secure symmetric-key encryption schemes can be constructed at the presence of a random oracle [11] while OT protocols cannot [28][4]. Moreover, as we already mentioned, although TE can be based on several known affine-secure KDM schemes (i.e., ones which enable strong simulation), the LPN assumption

[4] It seems that a similar statement holds even for public-key KDM-secure schemes. See [11,21].

(with constant error-rate) is a concrete example under which symmetric-key encryption scheme with KDM-security wrt affine functions exist, yet OT is not known to exist. Furthermore, since BHHI send the garbled circuit in the clear, it is not hard to show that the resulting scheme is not CCA-secure even if the TE provides CCA security. Finally, the modification of the GC protocol leads to a relatively complicated security proof.

2 Preliminaries

For a positive integer $n \in \mathbb{N}$, let $[n]$ denote the set $\{1, \ldots, n\}$, and U_n denote the uniform distribution over $\{0,1\}^n$. A function $\varepsilon(n)$ is *negligible* if it tends to zero faster than $1/n^c$ for every constant $c > 0$. The term *efficient* refers to probabilistic machines that run in polynomial time in the security parameter.

Efficient functions and randomized functions. A *randomized function* $f : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a function whose second input is treated as a random input. We write $f(x;r)$ to denote the evaluation of f on deterministic input x and random input r, and typically assume length regularity and efficient evaluation as follows: there are efficiently computable polynomials $m(n)$ and $\ell(n)$ and an efficiently computable circuit family $\left\{f_n : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}^{\ell(n)}\right\}$ which computes the restriction of f to n-bit deterministic inputs. If the function is *not* length regular, we assume that the circuit family is indexed by a pair of input and output parameters (n, ℓ), and require evaluation in time $\mathrm{poly}(n, \ell)$. Finally, a *deterministic* function corresponds to the special case where $m(n) = 0$.

Function ensembles. A *function ensemble* is a collection of functions $\{f_z\}_{z \in Z}$ indexed by an index set $Z \subseteq \{0,1\}^*$, where for each z the function f_z has a finite domain $\{0,1\}^{n(z)}$ and a finite range $\{0,1\}^{\ell(z)}$, where $n, \ell : \{0,1\}^* \to \mathbb{N}$. By default, we assume that ensembles are efficiently computable, that is, the functions $n(z), \ell(z)$, as well as the function $F(z,x) = f_z(x)$ are computable in time $\mathrm{poly}(|z|)$. Hence $n(z), \ell(z) < \mathrm{poly}(|z|)$. We also assume that $|z| < \mathrm{poly}(n(z), \ell(z))$.

Randomized encoding of functions. Intuitively, a randomized encoding of a function $g(x)$ is a randomized mapping $f(x;r)$ whose output distribution depends only on the output of g. We formalize this intuition via the notion of *computationally private randomized encoding* of [6], while adopting the original definition from a non-uniform adversarial setting to the uniform setting (i.e., adversaries are modeled by probabilistic polynomial-time Turing machines). Consider a function $g = \left\{g_n : \{0,1\}^n \to \{0,1\}^{\ell(n)}\right\}$ and a randomized function $f = \left\{f_n : \{0,1\}^n \times \{0,1\}^{m(n)} \to \{0,1\}^{s(n)}\right\}$, which are both efficiently computable. We say that f *encodes* g, if there exist an efficient recovery algorithms Rec and an efficient simulator Sim that satisfy the following:

- **perfect correctness.** For every $x \in \{0,1\}^n$, the error probabilities $\Pr[\mathsf{Rec}(1^n, f(x, \mathsf{U}_{m(n)})) \neq g(x)]$ and $\Pr[\mathsf{Rec}(1^n, \mathsf{Sim}(1^n, g(x))) \neq g(x)]$ are both zero[5].
- **computational privacy.** For every efficient adversary $\mathcal{A}$ we have that

$$\Pr[\mathcal{A}^{f(\cdot;\mathsf{U})}(1^n) = 1] - \Pr[\mathcal{A}^{\mathsf{Sim}(g(\cdot))}(1^n) = 1] < \mathrm{neg}(n),$$

  where the oracles are defined as follows: Given x the first oracle returns a sample from $f(x; \mathsf{U}_{m(|x|)})$ and the second oracle returns a sample from $\mathsf{Sim}(1^{|x|}, g(x))$.

This notion is naturally extended to functions $g_{n,\ell}$ which are not length-regular and are indexed by both input and output lengths. However, we always assume that privacy is parameterized *only* with the input length (i.e., the adversary's running-time/distinguishing-probability should be polynomial/negligible in the input length.) Note that, without loss of generality, we can assume that the relevant output length ℓ is always known to the decoder and simulator (i.e., it can be always encoded as part of the output of $f_{n,\ell}$).

Encryption schemes (syntax). An encryption scheme consists of three efficient algorithms $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$, where KG is a key generation algorithm which given a security parameter 1^k outputs a pair $(\mathsf{sk}, \mathsf{pk})$ of decryption and encryption keys; E is an encryption algorithm that takes a message $M \in \{0,1\}^*$ and an encryption key pk and outputs a ciphertext C; and D is a decryption algorithm that takes a ciphertext C and a decryption key sk and outputs a plaintext M'. We also assume that both algorithms take the security parameter 1^k as an additional input, but typically omit this dependency for simplicity. Correctness requires that the decryption error

$$\max_{M \in \{0,1\}^*} \Pr_{(\mathsf{sk},\mathsf{pk}) \stackrel{R}{\leftarrow} \mathsf{KG}(1^k)} [\mathsf{D}_{\mathsf{sk}}(\mathsf{E}_{\mathsf{pk}}(M)) \neq M],$$

should be negligible in k, where the probability is taken over the randomness of KG, E and D. For security parameter k, let $\mathcal{K}_k$ denote the space from which decryption keys are chosen. Without loss of generality, we always assume that $\mathcal{K}_k = \{0,1\}^k$.

Following Goldreich [23], we note that the above definition corresponds to both public-key and symmetric-key encryption schemes where the latter correspond to the special case in which the decryption key sk and encryption key pk are equal. As we will see, the difference between the two settings will be part of the security definitions.

3 KDM-Security

Let $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ be an encryption scheme with key space $\mathcal{K} = \{\mathcal{K}_k\}$. Let $t : \mathbb{N} \to \mathbb{N}$ be a function. A t-ary KDM function ensemble is an efficient

[5] Previous definitions require only that the first quantity is zero, however, all known constructions (of perfectly-correct randomized encoding) satisfy the current (stronger) definition.

ensemble of functions $\mathcal{F} = \left\{ f_{k,z} : \mathcal{K}_k^{t(k)} \to \{0,1\}^* \right\}_{(k,z)}$. We let $\mathcal{F}_k$ denote the set $\left\{ f_{k,z} : \mathcal{K}_k^{t(k)} \to \{0,1\}^* \right\}_z$. An $\mathcal{F}$-KDM Chosen-Ciphertext Attack (CCA) in the public-key setting is defined in Fig. 1 as a game that takes place between a challenger and an adversary $\mathcal{A}$. The advantage of $\mathcal{A}$ when attacking a scheme $\mathcal{E}$ is $\alpha(k) = \Pr[\mathcal{A} \text{ wins the KDM game}] - \frac{1}{2}$.

- **Initialization.** The challenger randomly chooses a bit $b \xleftarrow{R} \{0,1\}$ and $t = t(k)$ key-pairs $(\mathsf{sk}_1, \mathsf{pk}_1) \ldots, (\mathsf{sk}_t, \mathsf{pk}_t)$ by invoking $\mathsf{KG}(1^k)$ for t times. The adversary $\mathcal{A}$ can send a "public-key" query and get to see all the encryption keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_t)$.
- **Queries.** The adversary $\mathcal{A}$ may adaptively make polynomially-many queries of the following types:
  - **Encryption queries** of the form (i, M) where $i \in [t]$ and $M \in \{0,1\}^*$. The challenger responds with $C \xleftarrow{R} \mathsf{E}(\mathsf{pk}_i, M)$ if $b = 1$, and $C \xleftarrow{R} \mathsf{E}(\mathsf{pk}_i, 0^{|M|})$ if $b = 0$.
  - **KDM queries** of the form (i, f) where $i \in [t]$ and $f \in \mathcal{F}_k$. The challenger computes $M = f(\mathsf{sk}_1, \ldots, \mathsf{sk}_t)$ and responds with $C \xleftarrow{R} \mathsf{E}(\mathsf{pk}_i, M)$ if $b = 1$, and $C \xleftarrow{R} \mathsf{E}(\mathsf{pk}_i, 0^{|M|})$ if $b = 0$.
  - **Decryption queries** of the form (i, C) where $i \in [t]$ and the string C was not given as an answer of a previous encryption/KDM query. The challenger responds with $M = \mathsf{D}_{\mathsf{sk}_i}(C)$ regardless of the value of b.
- **Final phase.** The adversary outputs a bit $b' \in \{0,1\}$ and wins if $b = b'$.

Fig. 1. The $\mathcal{F}$-KDM game is defined with respect to the function ensemble $\mathcal{F} = \{\mathcal{F}_k\}$ and is indexed by the security parameter k. The presence (resp., absence) of public-key query captures the public-key (resp., symmetric-key) setting.

By restricting the power of the adversary in the KDM game (Fig. 1) we get other KDM settings. Specifically, the symmetric-key setting corresponds to adversaries of type sym who do not ask public-key queries, and the CPA setting corresponds to adversaries of type CPA who do not make decryption queries. Hence, we can classify KDM adversaries into one of the following four *types*: $(\mathsf{pub}, \mathsf{CCA})$, $(\mathsf{pub}, \mathsf{CPA})$, $(\mathsf{sym}, \mathsf{CCA})$, and $(\mathsf{sym}, \mathsf{CPA})$. An adversary of type T that conducts an $\mathcal{F}$-KDM attack is denoted as $(T, \mathcal{F})$-adversary.

Definition 1. (KDM-secure encryption) *Let T be a type, and $\mathcal{F}$ be a function ensemble. An encryption scheme is* $(T, \mathcal{F})$-KDM secure *if every efficient $(T, \mathcal{F})$ adversary has at most negligible advantage when attacking the scheme.*

Interesting KDM functions ensembles. For every $t = t(k)$ and for every type T we consider the following ensembles:

- **Selectors and projections**. If the ensemble $\mathcal{F}_k$ contains all selector functions $\{f_j : (\mathsf{sk}_1, \ldots, \mathsf{sk}_t) \mapsto \mathsf{sk}_j\}_{j \in [t]}$, we get the notion of *clique* security [12]

(which is stronger than *circular security* [16]), that is, the scheme is secure even if the adversary sees encryptions of the form $\mathsf{E}_{\mathsf{pk}_i}(\mathsf{sk}_j)$ for every $i, j \in [t]$. Another elementary class that slightly generalizes the previous ones is the class of all functions $f : (\mathbf{sk}) \mapsto v$ in which each output bit depends on (at most) a single bit of the input $\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_t)$. Namely, the j-th output bit v_j is either fixed to a constant or copies/flips an original bit of one of the keys, i.e., $v_j \in \{0, 1, \mathsf{sk}_{i,q}, 1 - \mathsf{sk}_{i,q}\}$, where $\mathsf{sk}_{i,q}$ is the q-th bit of the i-th secret key. We refer to this class as the class of *projections* and let $\Pi^t_{k,\ell}$ denote the restriction of this class to functions of input length kt and output length $\ell(k)$. Projections is a proper subclass of the class of affine functions $L : \mathbb{F}_2^{kt} \to \mathbb{F}_2^{\ell(k)}$.

– **Polynomial-size circuits [10].** For polynomials $p(\cdot)$ and $\ell(\cdot)$, let $\mathcal{C}^t_{k,\ell,p}$ denote the class of all circuits $C : \{0,1\}^{kt} \to \{0,1\}^{\ell(k)}$ of size at most $p(k) + p(\ell)$. Security with respect to this class is denoted by (p, ℓ)-*bounded circuit-size* KDM security. A slightly stronger notion of security is p-*length-dependent* KDM security which means that the scheme is KDM secure with respect to $\mathcal{C}^t_{k,\ell,p}$ for *every* polynomial ℓ. While, ultimately one would like to have KDM security with respect to all polynomial-size circuits (for arbitrary polynomial), it seems that p-length-dependent security, say for quadratic p, may be considered to be almost as powerful since it allows the adversary to use larger circuits by encrypting longer messages. In particular, one can represent essentially any polynomial-time computable function via padding. That is, if a function f is not in the class since its circuit is too large, then a "padded" version f' of f in which the output is padded with zeroes does fall into the ensemble. Furthermore, in [10] it was shown that if p is sufficiently large (e.g., the quadratic polynomial) then length-dependent security is sufficient for axiomatic-security applications (i.e., it gives the ability to securely instantiate symbolic protocols with axiomatic proofs of security).

The above definitions become stronger when the arity t grows. At one extreme, one may consider a single scheme which satisfies any of the above definitions for an arbitrary polynomial $t(k)$, and at the other extreme one may consider the case of $t = 1$, which is still non-trivial even for projection functions.

Reductions among KDM-ensembles. We say that a KDM function ensemble $\mathcal{G}$ KDM-reduces to another KDM function ensemble $\mathcal{F}$ (in symbols $\mathcal{G} \leq_{\mathsf{KDM}} \mathcal{F}$) if there exists a transformation which converts an encryption scheme $\mathcal{E}$ that is $\mathcal{F}$-KDM secure to an encryption scheme $\widehat{\mathcal{E}}$ which is $\mathcal{G}$-KDM secure. Formally, such a (black-box) reduction is composed of (1) (construction) an encryption scheme $\widehat{\mathcal{E}}$ which is given an oracle access to the scheme $\mathcal{E}$; and (2) (security reduction) an efficient algorithm $\mathcal{B}$ such that for any $\mathcal{F}$-adversary $\mathcal{A}$ which attacks $\mathcal{E}$ with advantage α, the $\mathcal{G}$-adversary $\mathcal{B}^{\mathcal{A},\mathcal{E}}$ attacks the scheme $\widehat{\mathcal{E}}$ with a similar advantage (up to a negligible loss). This definition can be instantiated with respect to all four different types. We say that the reduction is *type-preserving* if $\mathcal{B}^{\mathcal{A},\mathcal{E}}$ is always of the same type as $\mathcal{A}$ (i.e., $\mathcal{B}$ always ask the same type of queries that $\mathcal{A}$ asks in the KDM game.) Type preserving reduction extends KDM-security while being insensitive to the concrete setting which is being used. Formally,

Lemma 1 (KDM-reductions). *Suppose that the KDM function ensemble $\mathcal{G}$ KDM-reduces to the ensemble $\mathcal{F}$ via a type-preserving reduction $(\widehat{\mathcal{E}}, \mathcal{B})$. For every $T \in \{\mathsf{pub}, \mathsf{sym}\} \times \{\mathsf{CCA}, \mathsf{CPA}\}$, if the encryption scheme $\mathcal{E}$ is $(T, \mathcal{F})$-KDM secure then the scheme $\widehat{\mathcal{E}}^{\mathcal{E}}$ is $(T, \mathcal{G})$-KDM secure.*

4 Reductions and Completeness Results

4.1 KDM Reductions via Randomized Encoding

Let $\mathcal{F} = \{f_{k,z}\}$ and $\mathcal{G} = \{g_{k,w}\}$ be a pair of KDM function ensembles with the same arity $t = t(k)$. We say that $\mathcal{F}$ *encodes* $\mathcal{G}$ if every function $g(x)$ in $\mathcal{G}$ has a randomized encoding $f(x; r)$ such that for every fixing of the random string r, the resulting function $f_r(x)$ is in $\mathcal{F}$. More formally, the evaluation function $G_k(z, x)$ of $\mathcal{G}$ should have a randomized encoding $F_k((z, x); r)$ such that for every fixing of r and index z, the function $F_{k,z,r}(x) = F(k, z, x; r)$ corresponds to a function $f_{k,w}$ in $\mathcal{F}$, where the mapping from (z, r) to w should be efficiently computable in $\mathrm{poly}(k)$ time. Note that this means that the simulator and decoder are *universal* for all indices z, and depend only on the value of k.

Theorem 3 (main theorem). *Suppose that the KDM function ensemble $\mathcal{F}$ encodes the KDM function ensemble $\mathcal{G}$. Then, $\mathcal{G}$ KDM-reduces to $\mathcal{F}$ via a type-preserving reduction.*

To prove the theorem we need to describe a construction and a security reduction. From now on, let Sim and Rec be the universal simulator and recovery algorithm which establish the encoding of $\mathcal{G}$ by $\mathcal{F}$.

Construction 4. *Given oracle access to the encryption scheme $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$, we define the scheme $\widehat{\mathcal{E}}$ as follows*

$$\widehat{\mathsf{KG}}(1^k) = \mathsf{KG}(1^k) \qquad \widehat{\mathsf{E}}_{\mathsf{pk}}(M) = \mathsf{E}_{\mathsf{pk}}(\mathsf{Sim}(M)) \qquad \widehat{\mathsf{D}}_{\mathsf{sk}}(C) = \mathsf{Rec}(\mathsf{D}_{\mathsf{sk}}(C)),$$

where all algorithms (i.e., encryption, decryption, simulator and recovery) get the security parameter 1^k as an additional input.

It is not hard to see that the decryption error of the scheme $\widehat{\mathcal{E}}$ is the same as the decryption error of $\mathcal{E}$, as an improper decryption of $\widehat{\mathsf{E}}_{\mathsf{pk}}(M)$ happens only if $\mathsf{E}_{\mathsf{pk}}(M')$ is improperly decrypted where $M' \stackrel{R}{\leftarrow} \mathsf{Sim}(M)$.

We show that the security of $\widehat{\mathcal{E}}$ can be based on that of $\mathcal{E}$. Given an oracle access to a $(T, \mathcal{G})$ adversary $\mathcal{A}$ that attacks $\widehat{\mathcal{E}}$, we define a $(T, \mathcal{F})$ adversary $\mathcal{B}$ that attacks $\mathcal{E}$ by randomly choosing one of two strategies $\mathcal{B}_0$ and $\mathcal{B}_1$.

Reduction 5 (The adversary $\mathcal{B}^{\mathcal{A},\mathcal{E}}$). *Toss a coin $\sigma \stackrel{R}{\leftarrow} \{0, 1\}$. If $\sigma = 1$ invoke the following adversary $\mathcal{B}_1$:*

- ***Initialization:*** *$\mathcal{B}$ invokes $\mathcal{A}$. If $\mathcal{A}$ asks for the encryption keys then $\mathcal{B}$ makes a similar query and passes the answer to $\mathcal{A}$.*

- ***Encryption query:*** *If $\mathcal{A}$ makes an encryption query (i, M), for $i \in [t]$ and $M \in \{0,1\}^*$, then $\mathcal{B}$ samples $M' = \mathsf{Sim}(M)$, sends (i, M') as an encryption query (wrt to $\mathcal{E}$) and passes the answer of the challenger to $\mathcal{A}$.*
- ***KDM query:*** *If $\mathcal{A}$ makes a KDM query (i, g), for $i \in [t]$ and $g \in \mathcal{G}$, then the adversary $\mathcal{B}$ does the following: She uniformly chooses randomness r for the randomized encoding $f(\cdot; r)$ of $g(\cdot)$, and asks the KDM query (i, f_r) where $f_r(\cdot) = f(\cdot; r)$ which, by our assumption, is in $\mathcal{F}$. The answer of the challenger is being sent to $\mathcal{A}$.*
- ***Decryption query:*** *If $\mathcal{A}$ makes a decryption query (i, C), then $\mathcal{B}$ checks that it is legal (by inspecting all previous encryption/KDM queries), and if so, (1) passes the same decryption query to the challenger, (2) applies the recovery algorithm Rec to the result, and (3) sends it back to $\mathcal{A}$.*
- ***Termination:*** *$\mathcal{B}$ terminates with the same output of $\mathcal{A}$.*

If $\sigma = 0$ then invoke the adversary $\mathcal{B}_0$. This adversary is similar to $\mathcal{B}_1$ except that encryption and KDM queries of $\mathcal{A}$ are both translated into encryption queries as follows: given an encryption query of $\mathcal{A}$ of the form (i, M) (resp., KDM query of the form (i, g)), the adversary $\mathcal{B}_0$ samples $M' = \mathsf{Sim}(0^\ell)$ and asks for the ciphertext $\mathsf{E}_{\mathsf{pk}_i}(M')$, where ℓ is the length of M (resp., output length of g)[6]*. At the end, $\mathcal{B}_0$ flips the output of $\mathcal{A}$ and terminates.*

Note that the above reduction is indeed type-preserving. Let us first focus on the adversary $\mathcal{B}_1$. If the challenge bit b is 1 (i.e., when the challenger is in the "real-mode"), then the difference between the emulated view of $\mathcal{A}$ and the view of $\mathcal{A}$ in the actual KDM game, is only due to the difference in the way KDM queries are answered. In the real game answers to KDM queries are computed properly as $\widehat{\mathsf{E}}_{\mathsf{pk}_i}(g(\mathbf{sk})) = \mathsf{E}_{\mathsf{pk}_i}(\mathsf{Sim}(g(\mathbf{sk})))$, whereas in the emulated game they are computed by $\mathsf{E}_{\mathsf{pk}_i}(f(\mathbf{sk}; \mathsf{U}))$. However, this difference should not be noticeable due to the privacy of the randomized encoding. Formally, let $\alpha_b(k)$ (resp., $\beta_{\sigma,b}(k)$) denote the probability that $\mathcal{A}$ (resp., $\mathcal{B}_\sigma$) guesses the challenge bit when it takes the value b. Then,

Lemma 2. $|\beta_{1,1}(k) - \alpha_1(k)| \leq \mathrm{neg}(k)$.

Proof. We define the following distinguisher $\mathcal{D}$ which, given an oracle access to either $F(\cdot; \mathsf{U})$ or to $\mathsf{Sim}(G(\cdot))$, attempts to distinguish between the two. The adversary $\mathcal{D}$ emulates the challenger with challenge bit $b = 1$. It generates a key vector $(\mathsf{sk}_i, \mathsf{pk}_i)_{i \in [t]}$ by executing the key-generation algorithm $\mathsf{KG}(1^k)$ for t times. Then $\mathcal{D}$ invokes $\mathcal{A}$. If $\mathcal{A}$ asks a KDM query (i, g_z) then $\mathcal{D}$ calls its oracle with the value $G(z, \mathsf{sk}_1, \ldots, \mathsf{sk}_t)$. Let M denote the answer of the oracle. The distinguisher computes the ciphertext $C = \mathsf{E}_{\mathsf{pk}_i}(M)$ and sends the ciphertext C to $\mathcal{A}$. If $\mathcal{A}$ asks other types of queries such as public-key queries, encryption queries, and decryption queries, the distinguisher $\mathcal{D}$ answers them properly exactly as the real challenger does when it's in the real mode $b = 1$. (For the case of a decryption query (i, C), the distinguisher checks that it is legal by inspecting all

[6] Recall that the output length of $g \in \mathcal{G}$ is given as part of its description.

previous KDM/encryption queries, and if so, sends $\mathsf{D}_{\mathsf{sk}_i}(C)$.) The distinguisher halts with output 1 if and only if $\mathcal{A}$ outputs 1.

Note that: (1) If $\mathcal{D}$ gets an oracle access to $\mathsf{Sim}(G(\cdot))$ then the view of $\mathcal{A}$ is distributed exactly as in the real game and so in this case $\mathcal{D}$ outputs 1 with probability $\alpha_1(k)$; (2) If $\mathcal{D}$ gets an oracle access to $F(\cdot;\mathsf{U})$ then the view of $\mathcal{A}$ is distributed exactly as in the above reduction when $\mathcal{B}_1$ emulates the game with $b = 1$, and so in this case $\mathcal{D}$ outputs 1 with probability $\beta_{1,1}(k)$. Hence, by the privacy of the encoding, it follows that $|\beta_{1,1}(k) - \alpha_1(k)| \leq \mathrm{neg}(k)$. □

We would like to argue now that a similar thing happens in the "fake" mode when $b = 0$; namely, that $\beta_{1,0}$ is close to α_0. However, in this case real-game KDM queries are answered with $\widehat{\mathsf{E}}_{\mathsf{pk}_i}(0^\ell) = \mathsf{E}_{\mathsf{pk}_i}(\mathsf{Sim}(0^\ell))$, whereas in the game emulated by $\mathcal{B}_1$ these queries are answered by $\mathsf{E}_{\mathsf{pk}_i}(0^s)$, where $\ell = |g(\mathsf{sk}_1, \ldots, \mathsf{sk}_t)|$ and $s = |f(\mathsf{sk}_1, \ldots, \mathsf{sk}_t; \mathsf{U})|$. Although the privacy of the encoding ensures that the plaintexts are of the same length, i.e., $s = |\mathsf{Sim}(0^\ell)|$, the actual distributions of the plaintexts may differ, and so it may be the case that the two views are distinguishable. For this reason we need the adversary $\mathcal{B}_0$ which breaks the standard (non-KDM) security of $\mathcal{E}$ whenever such a gap exists. Formally, we will show that the average success probability of $\mathcal{B}_1$ and $\mathcal{B}_0$ is roughly half the success probability of $\mathcal{A}$. To this aim we prove the following

Lemma 3. *$\beta_{0,1}(k) = \alpha_0(k)$ and $\beta_{0,0}(k) + \beta_{1,0}(k) = 1$.*

Proof. First, we note that when the challenge bit $b = 1$, the view of $\mathcal{A}$ as emulated by $\mathcal{B}_0$ is identical to the view of $\mathcal{A}$ in the fake mode of the real game ($b = 0$). Indeed, in both cases a KDM query (i, g) (resp., an encryption query (i, M)) is answered with $\widehat{\mathsf{E}}_{\mathsf{pk}_i}(0^{|\ell|}) = \mathsf{E}_{\mathsf{pk}_i}(\mathsf{Sim}(0^\ell))$ where ℓ is the output length of g (resp., $\ell = |M|$). Hence, $\beta_{0,1}$, the probability that $\mathcal{B}_0$ outputs 1 when the challenger is in the real mode, is exactly the probability that $\mathcal{A}$ outputs 0 in the real game when the challenger is in the fake mode. (Recall that $\mathcal{B}$ flips the output of $\mathcal{A}$). The first equation follows.

To prove the second equality we first claim that when the challenge bit b is 0, the view of $\mathcal{A}$ when emulated by $\mathcal{B}_0$ is identical to the view of $\mathcal{A}$ as emulated by $\mathcal{B}_1$. Indeed, the only difference is that in the first case KDM queries (i, g) are answered by $\mathsf{E}(0^{|\mathsf{Sim}(g(\mathsf{sk}))|})$, while in the second case the answer is $\mathsf{E}(0^{|f(\mathsf{sk};r)|})$. The output lengths of f and $\mathsf{Sim}(g(\cdot))$ are fixed (for any $g \in \mathcal{G}$) and therefore should be equal (otherwise the privacy of the encoding is violated), and so the claim follows. The claim implies that $\beta_{0,0}(k) + \beta_{1,0}(k) = 1$, as $\mathcal{B}_1$ outputs the outcome of $\mathcal{A}$, and $\mathcal{B}_0$ flips it. □

By combining the two lemmas (2 and 3), it follows that the advantage $\beta = (\beta_{1,1}+\beta_{1,0}+\beta_{0,0}+\beta_{0,1})/4-\frac{1}{2}$ of $\mathcal{B}$ is at least $\frac{1}{2}\alpha-\mathrm{neg}(k)$ where $\alpha = \frac{1}{2}(\alpha_1+\alpha_0)-\frac{1}{2}$ is the advantage of $\mathcal{A}$. Hence, we established the correctness of the reduction.

Remark 1. Thm. 3 holds even if the encoding itself makes use of the underlying encryption scheme $\mathcal{E}$ as long as this usage is done in a fully black-box way (the same holds for any cryptographic primitive which can be based on $\mathcal{E}$ via a

black-box reduction e.g., one-way function). More precisely, our results hold (i.e., lead to black-box KDM reduction/construction) as long as the security of the encoding reduces to the security of the underlying primitive (i.e., $\mathcal{E}$) via a black-box reduction, and as long as the simulator and decoder can be implemented given a black-box access to the underlying primitive. Similarly, such a black box access can be given to the algorithm which maps fixed index/randomness pairs (z, r) to the index w of the function $g_{k,w} = G_{k,z,r}(x)$.

4.2 Completeness of Projections

In [6] it is shown that Yao's garbled circuit technique allows to encode any efficiently computable function by a decomposable encoding in which every bit depends on at most a single bit of the deterministic input. By combining this fact with Thm. 3 we get the following:

Proposition 1 (Completeness of projections). *For every polynomials $p(\cdot)$, $t(\cdot)$ and $\ell(\cdot)$, there exists a polynomial $q(\cdot)$ for which*

$$\mathcal{C}^t_{k,\ell,p} \leq_{\mathsf{KDM}} \Pi^t_{k,q}, \qquad \mathcal{C}^t_{k,p} \leq_{\mathsf{KDM}} \Pi^t_k, \tag{1}$$

where $\mathcal{C}^t_{k,\ell,p}$ is the t-ary ensemble of p-bounded circuits of output length ℓ, $\Pi^t_{k,q}$ is the t-ary ensemble of projections of output length q, $\mathcal{C}^t_{k,p} = \bigcup_{a\in\mathbb{N}} \mathcal{C}^t_{k,k^a,p}$, and $\Pi^t_k = \bigcup_{a\in\mathbb{N}} \Pi^t_{k,k^a}$. Moreover, the reductions are type preserving.

Hence, one can upgrade KDM security from (almost) the weakest KDM function ensemble to the very powerful notion of p-length-dependent KDM security.

Proof. By [6] any efficiently computable circuit family $\{g_k(x)\}$ of circuit complexity $a(k)$ can be encoded by a uniform computationally-private perfectly-correct encoding $\{\hat{g}_k(x; r)\}$ with the following properties: (1) The simulator and decoder use a black-box access to a symmetric encryption (equivalently, to a one-way function); (2) For every fixed randomness r, the resulting function $\hat{g}_{k,r}(x) = \hat{g}_k(x; r)$ is a projection function of output length $a(k)^{1+\varepsilon}$, where $\varepsilon > 0$ is an arbitrary small constant. (3) The mapping from the circuit of g_k to the circuit of $\hat{g}_{k,r}$ is efficiently computable given a black-box access to the symmetric encryption scheme.

Let $\{F_k\}$ be the universal (and uniform) circuit family for the mapping $(x, z) \mapsto y$ where $x \in (\{0,1\}^k)^t$, the string z is a description of a circuit $C_z : (\{0,1\}^k)^t \to \{0,1\}^{\ell(k)}$ of size $p(k)+p(\ell(k))$, and the string $y \in \{0,1\}^{\ell(k)}$ is $C_z(x)$. By applying the encoding from [6] to $\{F_k\}$ it follows that $\mathcal{C}^t_{k,\ell,p}$ is encoded by $\Pi^t_{k,q}$ where q is polynomial in the circuit size of F_k. The first part of the proposition now follows from Thm. 3.

The second part follows similarly, except that now we consider the (non-regular) function $\{G_{k,\ell}\}$ which computes the same mapping of F_k but for circuits C_z whose output length ℓ is given as an additional index, and not as a fixed polynomial in k. Again, by applying the encoding from [6] to $\{G_k\}$ it follows that $\mathcal{C}^t_{k,p}$ is encoded by Π^t_k, and the claim follows from Thm. 3. □

In the case of CPA KDM security, one can actually derive KDM-security with respect to projections of arbitrary output length (i.e., Π_k^t) from single-output projections $\Pi_{k,1}^t$.

Lemma 4 (Completeness of single-output projections for CPA-KDM). *For every polynomial $t(\cdot)$, we have $\Pi_k^t \leq_{\mathsf{KDM}} \Pi_{k,1}^t$, where the reduction holds for both* $(\mathsf{sym}, \mathsf{CPA})$ *and* $(\mathsf{pub}, \mathsf{CPA})$ *types.*

Proof. The proof follows by simple concatenation: the new encryption/decryption algorithms encrypts/decrypts the message/ciphertext by applying the original encryption/decryption algorithm in a bit by bit manner. Hence, a KDM query in Π_{k,k^a}^t for the new scheme can be emulated by k^a KDM queries in $\Pi_{k,1}^t$ for the original scheme. □

As shown in [9], we can use the standard encrypt-then-MAC transformation to upgrade the security of a scheme that satisfies $(\mathsf{sym}, \mathsf{CPA})$-KDM security into a scheme that satisfies $(\mathsf{sym}, \mathsf{CCA})$-security with respect to the same KDM class. A similar result was proven for the public-key setting by [15] via the Naor-Yung double-encryption paradigm (which relies on the existence of NIZK). Hence, by Proposition 1 and Lemma 4, we have:

Corollary 1 (KDM Collapse). *For every polynomials t and p, there exists a $\Pi_{k,1}^t$-KDM secure scheme if and only if there exists a t-ary p-length-dependent KDM secure encryption scheme. This holds unconditionally for the KDM types* $(\mathsf{sym}, \mathsf{CPA})$, $(\mathsf{sym}, \mathsf{CCA})$, *and* $(\mathsf{pub}, \mathsf{CPA})\}$, *and it holds for* $(\mathsf{pub}, \mathsf{CCA})$ *assuming the existence of non-interactive zero-knowledge proof system for* **NP**.

We remark that all the known constructions of affine-KDM secure encryption schemes [12,5,13] can be adapted to yield KDM security with respect to single-output projections (see the Appendix of the full version of this paper [4]). Hence, we get p-length-dependent $(\mathsf{pub}, \mathsf{CPA})$-KDM (resp., $(\mathsf{sym}, \mathsf{CCA})$) based on the DDH, LWE, or QR assumptions (resp., LPN assumption), which can be boosted into $(\mathsf{pub}, \mathsf{CCA})\}$-KDM assuming the existence of NIZK for **NP**.

5 On Full KDM Security

In this section, we study the possibility of constructing a scheme which satisfies KDM security for the class of all functions. In [10] it was shown that such a scheme can be constructed based on the existence of cyclic-secure fully homomorphic encryption (FHE) [20]. We show that a similar assumption is inherently required for full KDM security which is also *simulatable*. For simplicity, we focus on the case of arity $t = 1$ and single-query adversaries.

A public-key encryption scheme $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ is simulatable $\mathcal{F}$-KDM secure if there exists a polynomial-time simulator S such that for every $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KG}(1^k)$, and every circuit family $f_k \in \mathcal{F}_k$ of size $\mathrm{poly}(k)$, the ensemble $S(\mathsf{pk}, f_k)$ is indistinguishable from $\mathsf{E}_{\mathsf{pk}}(f_k(\mathsf{sk}))$. (Note that this means that the distinguisher holds the secret-key sk.) The notions of *simulatable circular-security* and

simulatable full-KDM security correspond to the two extreme cases where $\mathcal{F}$ contains only the identity function, and $\mathcal{F}$ contains all functions.

An FHE allows to translate encryptions of a message M into an encryption of a related message $h(M)$ for any polynomial-size circuit h. More formally, we say that $\mathcal{E}$ is *fully homomorphic* if there exists an efficient algorithm Eval such that for every $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KG}(1^k)$, every circuit family $\{h_k\}$ of size $\mathrm{poly}(k)$, and every sequence of messages $M_k \in \{0,1\}^{\mathrm{poly}(k)}$, the ensemble $\mathsf{Eval}(\mathsf{pk}, h_k, \mathsf{E}_{\mathsf{pk}}(M_k))$ is computationally indistinguishable from the ensemble $\mathsf{E}_{\mathsf{pk}}(h_k(M_k))$.

In [10], it was shown that if an encryption scheme is both simulatable circular-secure and fully-homomorphic then it is also simulatable fully-KDM secure. We show that the other direction holds as well, and so the two notions are equivalent.

Proposition 2. *Any simulatable fully-KDM secure encryption scheme is also fully-homomorphic circular-secure.*

Proof. Given a simulatable fully-KDM secure encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ with simulator S, we define $\mathsf{Eval}(\mathsf{pk}, h, C)$ by invoking S on the pair $(\mathsf{pk}, f_{h,C})$ where $f_{h,C}$ is the mapping $\mathsf{sk} \mapsto h(\mathsf{D}_{\mathsf{sk}}(C))$. Note that the circuit size of $f_{h,C}$ is polynomial in the circuit size of h (since D is efficient). Also, by definition, we have for every $(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KG}(1^k)$, sequence $\{M_k\}$ and sequence $\{h_k\}$,

$$\begin{aligned}\mathsf{Eval}(\mathsf{pk}, h_k, \mathsf{E}_{\mathsf{pk}}(M_k)) &\equiv S(\mathsf{pk}, f_{h_k, \mathsf{E}_{\mathsf{pk}}(M_k)}) \\ &\stackrel{c}{\equiv} \mathsf{E}_{\mathsf{pk}}(h_k(\mathsf{D}_{\mathsf{sk}}(\mathsf{E}_{\mathsf{pk}}(M_k)))) \\ &\equiv \mathsf{E}_{\mathsf{pk}}(h_k(M_k)),\end{aligned}$$

where $\equiv$ ($\stackrel{c}{\equiv}$) denotes statistical (computational) indistinguishability. □

Next, we show that if one removes the simultability requirement then any encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ which provides KDM security with respect to a function which is slightly stronger than its decryption algorithm D, is actually fully-KDM secure. This is done by observing that Gentry's "bootstrapping technique" can be adapted to the KDM setting.

Proposition 3. *Let* $T \in \{(\mathsf{pub}, \mathsf{CPA}), (\mathsf{sym}, \mathsf{CPA})\}$*, and let* $\mathcal{E} = (\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *be* T*-KDM secure encryption with respect to single-output projections and with respect to the function family* $\mathcal{F}_k = \{f_{C_1,C_2} : \mathsf{sk} \mapsto \mathrm{NAND}(\mathsf{D}_{\mathsf{sk}}(C_1), \mathsf{D}_{\mathsf{sk}}(C_2))\}$*, where* C_1, C_2 *ranges over* $\{0,1\}^{p(k)}$ *and* $p(k)$ *is the length of an encryption of one-bit message under secret-key of length* k*. Then,* $\mathcal{E}$ *is fully KDM secure of type* T*.*

Proof (Sketch). In the CPA setting it suffices to prove full KDM security with respect to all circuits of single output. We show how to convert an attacker which sends arbitrary KDM queries into one which uses only queries from $\mathcal{F}_k$. Let h be a circuit of size t, which is wlog composed of NAND gates, and let h_i denote the function computed by the i-th gate of h, where gates are ordered under some topological ordering. We translate a KDM query for h into t KDM calls to $\mathcal{F}_k$ by traversing the circuit from bottom to top in a gate by gate manner preserving the following invariant: The i-th query will be answered by a ciphertext C_i such

that, if the oracle is in the real mode $C_i = \mathsf{E}_{\mathsf{pk}}(h_i(\mathsf{sk}))$ and if it is in the fake mode $C_i = \mathsf{E}_{\mathsf{pk}}(0)$. For an input gate, this can be achieved directly by making a single KDM query with a single-output projection. To do this for an internal gate h_ℓ whose input wires are connected to h_i and h_j for some $i, j < \ell$, we use a KDM query to f_{C_i,C_j}. □

Acknowledgement. We thank Iftach Haitner, Yuval Ishai, and the anonymous referees for their helpful comments.

References

1. Abadi, M., Rogaway, P.: Reconciling two views of cryptography (the computational soundness of formal encryption). Journal of Cryptology 20(3), 395 (2007)
2. Acar, T., Belenkiy, M., Bellare, M., Cash, D.: Cryptographic agility and its relation to circular encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 403–422. Springer, Heidelberg (2010)
3. Adão, P., Bana, G., Herzog, J., Scedrov, A.: Soundness and completeness of formal encryption: The cases of key cycles and partial information leakage. Journal of Computer Security 17(5), 737–797 (2009)
4. Applebaum, B.: Key-dependent message security: Generic amplification and completeness theorems. Cryptology ePrint Archive, Report 2010/513 (2010)
5. Applebaum, B., Cash, D., Peikert, C., Sahai, A.: Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 595–618. Springer, Heidelberg (2009)
6. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Journal of Computional Complexity 15(2), 115–162 (2006)
7. Applebaum, B., Ishai, Y., Kushilevitz, E.: Cryptography in NC^0. SIAM Journal on Computing 36(4), 845–888 (2006)
8. 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)
9. Backes, M., Pfitzmann, B., Scedrov, A.: Key-dependent message security under active attacks - BRSIM/UC-soundness of symbolic encryption with key cycles. In: CSF 2007 (2007)
10. Barak, B., Haitner, I., Hofheinz, D., Ishai, Y.: Bounded key-dependent message security. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 423–444. Springer, Heidelberg (2010)
11. Black, J., Rogaway, P., Shrimpton, T.: Encryption-scheme security in the presence of key-dependent messages. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 62–75. Springer, Heidelberg (2003)
12. 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)
13. Brakerski, Z., Goldwasser, S.: Circular and leakage resilient public-key encryption under subgroup indistinguishability (or: Quadratic residuosity strikes back). In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 1–20. Springer, Heidelberg (2010)
14. Brakerski, Z., Goldwasser, S., Kalai, Y.: Circular-secure encryption beyond affine functions. In: TCC 2011 (2011)

15. Camenisch, J., Chandran, N., Shoup, V.: A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 351–368. Springer, Heidelberg (2009)
16. Camenisch, J., 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)
17. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully homomorphic encryption over the integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
18. Dolev, D., Dwork, C., Naor, M.: Non-malleable cryptography (extended abstract). In: Proc. of STOC, pp. 542–552 (1991)
19. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the Association for Computing Machinery 28 (1985)
20. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: Proc. of STOC, pp. 169–178 (2009)
21. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: Proc. of FOCS (2000)
22. Goldreich, O.: Foundations of Cryptography: Basic Tools. Cambridge University Press, Cambridge (2001)
23. Goldreich, O.: Foundations of Cryptography: Basic Applications. Cambridge University Press, Cambridge (2004)
24. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984)
25. 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)
26. Halevi, S., Krawczyk, H.: Security under key-dependent inputs. In: ACM CCS 2007, pp. 466–475 (2007)
27. 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)
28. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 8–26. Springer, Heidelberg (1990)
29. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: A new representation with applications to round-efficient secure computation. In: Proc. of FOCS, pp. 294–304 (2000)
30. Ishai, Y., Kushilevitz, E.: Perfect constant-round secure computation via perfect randomizing polynomials. In: Widmayer, P., Triguero, F., Morales, R., Hennessy, M., Eidenbenz, S., Conejo, R. (eds.) ICALP 2002. LNCS, vol. 2380, pp. 244–256. Springer, Heidelberg (2002)
31. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: Proc. of STOC, pp. 427–437 (1990)
32. Rabin, M.: Digitalized signatures and public key functions as intractable as factoring. Tech. Rep. 212, LCS, MIT (1979)
33. 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)
34. Yao, A.C.: How to generate and exchange secrets. In: Proc. of FOCS, pp. 162–167 (1986)

Unbounded HIBE and Attribute-Based Encryption

Allison Lewko* and Brent Waters**

University of Texas Austin
{alewko,bwaters}@cs.utexas.edu

Abstract. In this work, we present HIBE and ABE schemes which are "unbounded" in the sense that the public parameters do not impose additional limitations on the functionality of the systems. In all previous constructions of HIBE in the standard model, a maximum hierarchy depth had to be fixed at setup. In all previous constructions of ABE in the standard model, either a small universe size or a bound on the size of attribute sets had to be fixed at setup. Our constructions avoid these limitations. We use a nested dual system encryption argument to prove full security for our HIBE scheme and selective security for our ABE scheme, both in the standard model and relying on static assumptions. Our ABE scheme supports LSSS matrices as access structures and also provides delegation capabilities to users.

1 Introduction

Hierarchical Identity-Based Encryption (HIBE) systems [29,26] and Attribute-Based Encryption (ABE) systems [40] offer users more levels of flexibility in sharing and managing sensitive data than are provided by Identity-Based and Public Key Encryption systems. In a hierarchical identity-based encryption scheme, user identities are arranged in an organizational hierarchy. Anyone can encrypt a message to any identity in the system using the public parameters. An identity at level k in the hierarchy can use its secret key to delegate secret keys to its subordinates, but cannot decrypt any messages which are intended for recipients other than itself and its subordinates. In a Key-Policy Attribute-Based Encryption (KP-ABE) system [28], users have secret keys which are associated with access policies over a universe of attributes and ciphertexts are associated with sets of attributes. A user can decrypt a message encrypted to a set of attributes S only if S satisfies the access policy of the user's key.

Both HIBE and ABE systems are designed to accommodate certain changes in the needs of users over time, but current constructions have some inherent limitations. For instance, new users can enter an HIBE system and collect

* Supported by National Defense Science and Engineering Graduate Fellowship.
** Supported by NSF CNS-0915361, and CNS-0952692, the MURI program under AFOSR Grant No: FA9550-08-1-0352. Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), a Google Faculty Research award, and the Alfred P. Sloan Foundation.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 547–567, 2011.

secret keys without requiring any change to the public parameters or the keys of users already present. However, for all previous constructions in the standard model, the identities of new users must fit within the hierarchy depth specified by the public parameters. More precisely, the size of the public parameters grows linearly with the maximum depth of the hierarchy, and it is impossible to add new levels to the hierarchy once the public parameters are fixed. In the ABE setting, the particular access policies and attribute sets employed by users may change over time, but current constructions in the standard model do not allow complete versatility in the choice of attributes and policies once the public parameters have been set. In "small universe" constructions (e.g. [28,31]), a polynomially sized universe of attributes must be fixed at setup, and the size of the public parameters grows linearly with the size of the chosen attribute universe. In "large universe" constructions (e.g. [28]), the attribute universe is exponentially large, but the size of a set S used for encryption is bounded by a parameter n which is fixed at setup. The size of the public parameters grows linearly with n.

This places an undesirable burden on someone wishing to deploy an HIBE or ABE system to be used in practice. If the setup parameters are chosen to be too small, the system will not achieve the desired longevity and will need to be completely re-initialized when users exhaust its overly restrictive structure. If the setup parameters are chosen to be too large, then the public parameters of the system will be needlessly large and this will cause unnecessary inefficiency.

Removing these restrictions from previous approaches appears to be quite challenging. For example, many standard model HIBE constructions employ structures similar to the Boneh-Boyen HIBE in [9] (e.g. [11,10,45,34] fall roughly into this framework). At a high level, these systems all rely on hash functions H which map identity vectors to group elements in a particular way. More specifically, we suppose that a user at level j in the hierarchy is associated with an identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$. The hash function H uses d fixed group elements $u_1, \ldots, u_d$ in a bilinear group G of order p (for example). Upon receiving an identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ as input, H somehow chooses k vectors $\boldsymbol{v}^1 = (v_1^1, \ldots, v_d^1), \ldots, \boldsymbol{v}^k = (v_1^k, \ldots, v_d^k) \in \mathbb{Z}_p^d$, where k is a function of j and the maximum depth of the hierarchy. In particular, k will be strictly less than d. It then outputs group elements of the form

$$\left(u_1^{v_1^1} \cdot u_2^{v_2^1} \cdots u_d^{v_d^1}\right), \ldots, \left(u_1^{v_1^k} \cdot u_2^{v_2^k} \cdots u_d^{v_d^k}\right).$$

In forming the secret keys or ciphertexts, these group elements are typically each raised to the same random exponent in $\mathbb{Z}_p$.

If we try to apply this approach without bounding the maximum depth of the hierarchy, then for some identity vectors, we will need to produce $\geq d$ samples of the form above, and each will be raised to the same exponent $s \in \mathbb{Z}_p$. This causes insecurity - since our vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^k$ reside in a d-dimensional space, most collections of d of them will be linearly independent, and will span $\mathbb{Z}_p^d$. This will allow an attacker to create a new sample

$$\left(u_1^{v_1^*} \cdot u_2^{v_2^*} \cdots u_d^{v_d^*}\right)^s$$

for any vector $(v_1^*, \ldots, v_d^*)$ that it wants, by taking its received samples, raising them to appropriate powers, and multiplying the results. For this reason, achieving unbounded HIBE systems by relying on these sorts of hash functions seems unlikely.

Our Contribution. Using new techniques, we obtain "unbounded" HIBE and ABE schemes. Our HIBE scheme can accommodate arbitrary hierarchy depths from public parameters which consist of only a constant number of group elements. This eliminates the need to decide maximum hierarchy depth at setup and reduces the size of the public parameters. We prove our scheme fully secure in the standard model, relying on static, generically secure assumptions in composite order bilinear groups. Our ABE scheme has a large attribute universe and imposes no bound on the size of attribute sets used for encryption. It also has public parameters which are a constant number of group elements. It supports LSSS matrices as access structures, and additionally provides delegation capabilities to users. Our ABE scheme is proven selectively secure[1] from the same static, generically secure assumptions in composite order bilinear groups.

Our Techniques. We overcome the limitations of previous constructions by employing a secret-sharing technique and introducing fresh "local" randomness at each level of the keys and ciphertexts. Thus, instead of needing to create too many samples from a bounded dimensional vector space with the same randomness, we will be creating many samples which each have *new randomness*. This avoids the insecurity of the previous approach described above.

To create a secret key for a user in our HIBE and ABE systems, we first split the master secret into shares that will be associated with the components of the user's identity vector or the rows of its access matrix. Each share is then blinded by randomness which is freshly chosen for each share and links the share to its corresponding identity or attribute.

The main obstacle to proving the security of our schemes is the low amount of entropy provided by the short public parameters. This poses a challenge for both partitioning proof techniques and the more recently introduced technique of dual system encryption [45]. To successfully execute a partitioning proof, we would need to program the public parameters to allow cancelations when the simulator attempts to make a certain key or keys. However, the small number of degrees of freedom available in the public parameters make it difficult to program in keys of arbitrary depth. To use a dual system encryption proof, we must execute an information-theoretic argument in a low entropy context - this is a challenge, but a surmountable one. We ultimately accomplish this by introducing a *nested dual system encryption approach* which allows us to make our information-theoretic

[1] This is a weaker model of security where the attacker must specify what it will be challenged on before seeing the public parameters.

argument in a very localized context, where the limited entropy of the public parameters is sufficient.

In a dual system encryption scheme, ciphertexts and keys can take two forms: normal and semi-functional. Normal keys can decrypt both normal and semi-functional ciphertexts, while semi-functional keys can only decrypt normal ciphertexts. Security is proven through a hybrid argument over a sequence of games, where first the challenge ciphertext is changed to semi-functional, and then the keys are changed to semi-functional one by one. At the end of this process, the simulator does not need to produce keys and ciphertexts which decrypt properly, and now security can be proven directly. However, we must avoid a potential paradox: at the point in the game sequence where a key is being changed to semi-functional, the simulator should not be able to test the nature of the key for itself by testing decryption on a semi-functional ciphertext. This can be enforced with nominal semi-functionality, meaning that if the simulator tries to make a semi-functional ciphertext which can be decrypted by the key of unknown type, then the key, ciphertext pair will actually be correlated so that decryption will succeed regardless of semi-functionality. In other words, even if semi-functional terms are present, they will cancel out upon decryption with the semi-functional ciphertext and hence be undetectable to the simulator. This nominal semi-functionality should be hidden from an attacker who cannot request keys capable of decrypting the ciphertext it receives.

The limited entropy of the public parameters in our systems does not enable us to hide nominal semi-functionality from the attacker if we try to change a key from normal to semi-functional in a single step. To overcome this, we introduce the concept of *ephemeral semi-functionality* for keys and ciphertexts. Ephemeral semi-functionality for keys is a temporary state which serves as an intermediate step between normalcy and semi-functionality. Ephemeral semi-functionality for ciphertexts is a temporary state of enhanced semi-functionality - ephemeral semi-functional keys can still decrypt semi-functional ciphertexts, but ephemeral semi-functional ciphertexts can only be decrypted by normal keys. Our proof employs a nested hybrid structure, where first the ciphertext is changed to semi-functional, then one key at a time is first changed to ephemeral semi-functional, then the ciphertext is changed to ephemeral semi-functional, and then the single key and ciphertext are both changed to semi-functional.

We note that a key first becomes incapable of decrypting ciphertexts when both are ephemeral semi-functional, and there is only *one* ephemeral semi-functional key at a time. This allows us to employ a information-theoretic argument to hide nominality in a more local context, where we need only be concerned with a single key. Even with this nested approach, accomplishing the game transitions with low entropy is still an intricate process - we employ additional inner hybrid steps to gradually change the distributions of keys and ciphertexts. In the KP-ABE setting, we also change to the selective security model.

Related Work. Identity-Based Encryption was conceived by Shamir in [41] and first constructed by Boneh and Franklin [12] and Cocks [23]. These were proven secure in the random oracle model. Canetti, Halevi, and Katz [16] and Boneh

and Boyen [9] then provided systems which were proven selectively secure in the standard model. Fully secure solutions in the standard model were later provided by Boneh and Boyen [10] and Waters [44]. The Waters system was efficient and proven from the well-established decisional Bilinear Diffie-Hellman assumption, but had public parameters consisting of $O(\lambda)$ group elements, where λ is the security parameter. The system provided by Gentry [24] had short public parameters and was proven secure in the standard model, but relied on a "q-type" assumption (meaning that the number of terms in the assumption depends on the number of queries q made by an attacker). Using dual system encryption, Waters [45] provided an efficient IBE system with short public parameters proven fully secure under the decisional linear and decisional bilinear Diffie-Hellman assumptions. In the random oracle model, additional schemes were provided by Boneh, Gentry, and Hamburg [13] under the quadratic residuosity assumption and by Gentry, Peikert, and Vaikuntanathan [25] under lattice-based assumptions.

Hierarchical Identity-Based Encryption was first introduced by Horwitz and Lynn [29] and constructed by Gentry and Silverberg [26] in the random oracle model. Selectively-secure constructions in the standard model were then provided by Boneh and Boyen [9] and Boneh, Boyen, and Goh [11]. The scheme of Boneh, Boyen, and Goh achieved short ciphertexts (ciphertext size independent of the hierarchy depth). Gentry and Halevi gave a fully secure construction for polynomial depth, relying on a complex assumption. Waters [45] provided a fully secure scheme from the decisional linear and decisional bilinear Diffie-Hellman assumptions. Lewko and Waters [34] provided a construction with short ciphertext, also achieving full security from static assumptions. Lattice-based HIBE systems were constructed by Cash, Hofheinz, Kiltz, and Peikert [17] and Agrawal, Boneh, and Boyen [1]. Agrawal, Boneh, and Boyen [2] constructed a lattice HIBE scheme where the dimension of the delegated lattices does not grow with the levels of the hierarchy. The lattice systems are proven either secure in the random oracle model or selectively secure in the standard model. Chatterjee and Sarkar [20] defined a couple of new security models for HIBE, and also suggested an HIBE system in a new, much weaker security model which can support arbitrary depths (i.e. a maximum depth is not fixed at setup). However, this system does not achieve even selective security - the authors point out that there is a simple attack against it in the standard selective security model.

Attribute-Based Encryption was introduced by Sahai and Waters [40]. Subsequently, Goyal, Pandey, Sahai, and Waters [28] defined two forms of ABE: Key-Policy ABE (where keys are associated with access policies and ciphertexts are associated with sets of attributes) and Ciphertext-Policy ABE (where ciphertexts are associated with access policies and keys are associated with sets of attributes). Several constructions of selectively secure KP-ABE and CP-ABE systems followed (e.g. [8,21,27,28,38,39,46]). Fully secure constructions were recently provided by Lewko, Okamoto, Sahai, Takashima, and Waters [31] and Okamoto and Takashima [37]. The works of Chase [18] and Chase and Chow [19] considered the problem of ABE in a setting with multiple authorities. The related concept of Predicate Encryption was introduced by Katz, Sahai and

Waters [30] and further studied in [31,36,37,42]. Other works have considered related problems without addressing collusion resistance [3,4,5,15,35,43].

The methodology of dual system encryption was introduced by Waters [45] and later used in [34,31,37,22,32] to obtain adaptive security (and also leakage resilience in [22,32]) for IBE, HIBE, and ABE systems. The abstractions we provide for dual system encryption in the HIBE and ABE settings are similar to the abstractions provided in [32], except that we do not consider leakage resilience and also provide only selective security in the ABE case.

2 Dual System Encryption HIBE

We now define a Dual System Encryption HIBE scheme. (This is similar to the abstraction given in [32], but things are simpler in our case because we do not consider leakage resilience.) In addition to the five algorithms of a regular HIBE scheme (Setup, Encrypt, KeyGen, Decrypt, and Delegate), a Dual System Encryption HIBE scheme also has algorithms KeyGenSF and EncryptSF, which produce semi-functional keys and ciphertexts, respectively. Unlike the Setup, Encrypt, KeyGen, Decrypt, and Delegate algorithms, the KeyGenSF and EncryptSF algorithms need not run in polynomial time (given only their input parameters), since they are used only for the proof of security and are not used in the normal operation of the system. Notice that decryption will work as before unless both the secret key and ciphertext are semi-functional, in which case decryption will always fail.

Setup$(\lambda) \rightarrow \mathrm{PP}, \mathrm{MSK}$. The setup algorithm takes the security parameter λ as input and outputs the public parameters PP and the master secret key MSK.

Encrypt$(M, \boldsymbol{\mathcal{I}}, \mathrm{PP}) \rightarrow \mathrm{CT}$. The encryption algorithm takes a message M, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters PP as input and outputs the ciphertext CT.

EncryptSF$(M, \boldsymbol{\mathcal{I}}, \mathrm{PP}) \rightarrow \widetilde{\mathrm{CT}}$. The semi-functional encryption algorithm takes a message M, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters PP as input. It produces a semi-functional ciphertext $\widetilde{\mathrm{CT}}$.

KeyGen$(\mathrm{MSK}, \boldsymbol{\mathcal{I}}, \mathrm{PP}) \rightarrow \mathrm{SK}_{\boldsymbol{\mathcal{I}}}$. The key generation algorithm takes the master secret key MSK, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters as input and outputs a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ for that identity vector.

KeyGenSF$(\mathrm{MSK}, \boldsymbol{\mathcal{I}}, \mathrm{PP}) \rightarrow \widetilde{\mathrm{SK}}_{\boldsymbol{\mathcal{I}}}$. The semi-functional key generation algorithm takes the master secret key MSK, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters as input. It produces a semi-functional secret key $\widetilde{SK}_{\boldsymbol{\mathcal{I}}}$ for $\boldsymbol{\mathcal{I}}$.

Decrypt$(\mathrm{CT}, \mathrm{PP}, \mathrm{SK}_{\boldsymbol{\mathcal{I}}}) \rightarrow M$. The decryption algorithm takes a ciphertext CT, the public parameters PP, and a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ as input. If the identity vector of the secret key $\boldsymbol{\mathcal{I}}$ is a prefix of the identity vector used to encrypt the ciphertext and the key and ciphertext are *not both semi-functional*, the decryption algorithm outputs the message M.

Delegate$(\mathrm{SK}_{\boldsymbol{\mathcal{I}}}, \mathcal{I}', \mathrm{PP}) \rightarrow \mathrm{SK}_{\boldsymbol{\mathcal{I}}:\mathcal{I}'}$. The delegation algorithm takes a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ for identity vector $\boldsymbol{\mathcal{I}}$, an identity $\mathcal{I}'$, and the public parameters PP as input. It outputs a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}:\mathcal{I}'}$ for the identity vector $\boldsymbol{\mathcal{I}} : \mathcal{I}'$, which denotes the concatenation of $\boldsymbol{\mathcal{I}}$ and $\mathcal{I}'$.

2.1 Security Properties for Dual System Encryption HIBE

We define four security properties for a dual system encryption HIBE. We will show that a system which has these four properties is a secure HIBE. To define these properties, we define the following variations of the security game for HIBE, which we call Game HIBE. In this game, the attacker may make Create, Delegate, and Reveal queries. In response to Create queries, the challenger creates the specified key. In response to Delegate queries, the challenger applies the delegation algorithm to produce the requested key from a specified superior key. In response to Reveal queries, the challenger gives the requested key to the attacker. For background on HIBE and its security definition and proofs of the theorems in this section, see the full version of this paper [33].

We first define Game HIBE_{WD} to be the same as Game HIBE, except without delegation. More precisely, instead of making Create, Delegate, and Reveal queries, the attacker simply makes KeyGen queries - i.e. it provides the challenger with an identity vector, the challenger creates a secret key for this identity vector by calling KeyGen, and then gives the secret key to the attacker. The only restriction is that no queried identity vectors can be prefixes of the challenge identity vector provided for the challenge ciphertext.

We next define Game HIBE_C to be the same as Game HIBE_{WD}, except that the challenge ciphertext is generated by a call to EncryptSF instead of Encrypt (i.e. a semi-functional ciphertext is given to the attacker). We also define Game HIBE_{SF} to be the same as Game HIBE_C, except that the challenger replaces all KeyGen calls with calls to KeyGenSF. In other words, the challenge ciphertext and all the secret keys given to the attacker will be semi-functional.

Delegation Invariance. We say a dual system encryption HIBE scheme $\Pi_D =$ (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *delegation invariance* if for any PPT algorithm $\mathcal{A}$, there exists another PPT algorithm $\mathcal{A}'$ such that the advantage of $\mathcal{A}$ in Game HIBE is negligibly close to the advantage of $\mathcal{A}'$ in Game HIBE_{WD}. (Here, $\mathcal{A}$ makes Create, Delegate, and Reveal queries, while $\mathcal{A}'$ makes KeyGen queries.) We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE}(\lambda) - Adv_{\mathcal{A}'}^{HIBE_{WD}}(\lambda) \right| = negl(\lambda).$$

Semi-functional Ciphertext Invariance. We say a dual system encryption HIBE scheme $\Pi_D =$ (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *semi-functional ciphertext invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_{WD} is negligibly close to its advantage in Game HIBE_C. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_{WD}}(\lambda) - Adv_{\mathcal{A}}^{HIBE_C}(\lambda) \right| = negl(\lambda).$$

Semi-functional Key Invariance. We say a dual system encryption HIBE scheme $\Pi_D = (\text{Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate})$ has *semi-functional key invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_C is negligibly close to its advantage in Game HIBE_{SF}. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_C}(\lambda) - Adv_{\mathcal{A}}^{HIBE_{SF}}(\lambda) \right| = negl(\lambda).$$

Semi-functional Security. We say a dual system encryption HIBE scheme $\Pi_D = (\text{Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate})$ has *semi-functional security* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_{SF} is negligible. We denote this by:

$$Adv_{\mathcal{A}}^{HIBE_{SF}}(\lambda) = negl(\lambda).$$

Theorem 1. *If a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has delegation invariance, semi-functional ciphertext invariance, semi-functional key invariance, and semi-functional security, then Π = (Setup, Encrypt, KeyGen, Decrypt, Delegate) is a secure HIBE scheme.*

2.2 An Alternative Security Property

The semi-functional key invariance property can be difficult to prove directly. For this reason, we define an alternative property, *one semi-functional key invariance*, which is more convenient to work with and which implies semi-functional key invariance through a hybrid argument.

To define one semi-functional key invariance, we must define an additional game, Game HIBE_b (where b represents a bit that can take value 0 or 1). In this game, when the attacker requests a key, it specifies whether it wants a normal or semi-functional key. If the attacker requests a normal key, the challenger makes a call to KeyGen to generate the key and returns it to the attacker. If the attacker requests a semi-functional key, the challenger makes a call to KeyGenSF to generate the key and returns it to the attacker. At some point, the attacker specifies a *challenge key.* In response, the challenger provides a normal key if $b = 0$ and a semi-functional key if $b = 1$. When the attacker requests the challenge ciphertext, it is given a semi-functional ciphertext (under the usual restriction that no key given to the attacker can be for an identity vector which is a prefix of the identity vector of the ciphertext). Note that the only difference between Game HIBE_0 and Game HIBE_1 is the nature of a single key specified by the attacker.

One Semi-functional Key Invariance We say a dual system encryption HIBE scheme $\Pi_D = (\text{Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate})$ has *one semi-functional key invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_0 is negligibly close to its advantage in Game HIBE_1. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_0}(\lambda) - Adv_{\mathcal{A}}^{HIBE_1}(\lambda) \right| = negl(\lambda).$$

Theorem 2. *If a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has one semi-functional key invariance, then it has semi-functional key invariance.*

3 Complexity Assumptions

Our construction will use composite order bilinear groups, first introduced in [14]. Additional background about these groups can be found in the full version. We let G denote a bilinear group order $N = p_1 p_2 p_3$, which is a product of three distinct primes, and we let $e : G \times G \to G_T$ denote the bilinear map.

In the assumptions below, we let G_{p_1} denote the subgroup of order p_1 in G, for example. We note that if $g_i \in G_{p_i}$ and $g_j \in G_{p_j}$ for $i \neq j$, then $e(g_i, g_j) = 1$. We use the notation $X \xleftarrow{R} S$ to express that X is chosen uniformly randomly from the finite set S. We note that except for Assumption 2, all of these assumptions are special cases of the General Subgroup Decision Assumption defined in [7]. Informally, the General Subgroup Decision Assumption can be described as follows: in a bilinear group of order $N = p_1 p_2 \ldots p_n$, there is a subgroup of order $\prod_{i \in S} p_i$ for each subset $S \subseteq \{1, \ldots, n\}$. We let S_0, S_1 denote two such subsets. It should be hard to distinguish a random element from the subgroup corresponding to S_0 from a random element of the subgroup corresponding to S_1, even if one is given random elements from subgroups corresponding to other sets S_i which satisfy either that $S_0 \cap S_i = \emptyset = S_1 \cap S_i$ or $S_0 \cap S_i \neq \emptyset \neq S_1 \cap S_i$. The formal statements of our precise assumptions are below. Assumption 1 here is a slightly weaker form of Assumption 1 in [34], and Assumptions 2 and 4 here also appeared in [34]. In our proofs, we will also invoke Assumption 4 with the roles of p_2 and p_3 reversed.

Assumption 1. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g \xleftarrow{R} G_{p_1},$$

$$D = (\mathbb{G}, g),$$

$$T_1 \xleftarrow{R} G_{p_1 p_2},\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := \left| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \right|.$$

We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

Assumption 2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g \xleftarrow{R} G_{p_1},\ g_2, X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ \alpha, s \xleftarrow{R} \mathbb{Z}_N$$

$$D = (\mathbb{G}, g, g_2, g_3, g^{\alpha} X_2, g^{s} Y_2),$$

$$T_1 = e(g,g)^{\alpha s},\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

Assumption 3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ X_3 \xleftarrow{R} G_{p_3}$$

$$D = (\mathbb{G}, g, g_2, X_1 X_3),$$

$$T_1 \xleftarrow{R} G_{p_1},\ T_2 \xleftarrow{R} G_{p_1 p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

Assumption 4. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} G_{p_1},\ X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3, Y_3 \xleftarrow{R} G_{p_3}$$

$$D = (\mathbb{G}, g, g_3, X_1 X_2, Y_2 Y_3),$$

$$T_1 \xleftarrow{R} G_{p_1 p_3},\ T_2 \xleftarrow{R} G.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be:

$$Adv4_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

We say that $\mathcal{G}$ satisfies Assumption 4 if $Adv4_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

4 Our HIBE Construction

We now present our dual system encryption HIBE scheme. Our system is constructed in a composite order bilinear group whose order N is the product of three distinct primes. We assume that our identity vectors have components which are elements of $\mathbb{Z}_N$. In the semi-functional algorithms below, we let g_2 denote a generator of G_{p_2} and g_3 denote a generator of G_{p_3}.

We will assume that identity vectors are encoded such that if identity vector $\boldsymbol{\mathcal{I}}$ is not a prefix of identity vector $\boldsymbol{\mathcal{I}}^*$, then the *last* component of $\boldsymbol{\mathcal{I}}$ is not equal to *any* component of $\boldsymbol{I}^*$. In other words, when $\boldsymbol{\mathcal{I}} = (\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is not a prefix of $\boldsymbol{\mathcal{I}}^* = (\mathcal{I}_1^*, \ldots, \mathcal{I}_\ell^*)$, we assume that $\mathcal{I}_j \neq \mathcal{I}_k^*$ for all $k \in \{1, \ldots, \ell\}$. A simple scheme to achieve this encoding is to replace an arbitrary vector of component identities, $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ by concatenating in each entry with all the previous entries: $(\mathcal{I}_1, \mathcal{I}_1||\mathcal{I}_2, \ldots, \mathcal{I}_1||\mathcal{I}_2||\cdots||\mathcal{I}_j)$. This creates entries which grow in length, but we can avoid this by applying a collision-resistant hash function to each of them.

The main idea of our construction is to employ a secret-sharing approach across the levels of our secret keys. A user's secret key involves a sharing of the master secret key α as a sum of exponents, where each piece of the sum is additionally blinded by a random term which is unique to that piece. In other words, each share of α is blinded by randomness which is "local" to that share. To successfully decrypt, a user must effectively unblind each share, which can only be accomplished by a user with a j^{th} level identity vector which matches the ciphertext identity vector in *all of the components one through* j. If a user's identity vector fails to match in component $k \leq j$, then the user will fail to recover the k^{th} share needed, thus preventing successful decryption. In essence, each level of the key and ciphertext closely resembles an instance of the Boneh-Boyen IBE scheme [9] with an added layer of local randomness between the shares of the master secret key and the terms involving the identities. These instances share the same public parameters, which we are able to accommodate by using fresh local randomness in the levels of the key and ciphertext.

4.1 Construction

Setup$(\lambda) \rightarrow \mathrm{PP}, \mathrm{MSK}$ The setup algorithm takes in the security parameter λ and chooses a bilinear group G of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes. We let G_{p_i} denote the subgroup of order p_i in G. The algorithm then chooses g, u, h, v, w uniformly randomly from G_{p_1}, and α uniformly randomly from $\mathbb{Z}_N$. It sets the public parameters as:

$$PP := \{N, G, g, u, h, v, w, e(g,g)^\alpha\}.$$

The master secret key is α.

Encrypt$(M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}), \rightarrow \mathrm{CT}$ The encryption algorithm chooses s, $t_1, \ldots, t_j$ uniformly randomly from $\mathbb{Z}_N$. It creates the ciphertext as:

$$C := Me(g,g)^{\alpha s},\ C_0 := g^s,$$

$$C_{i,1} := w^s v^{t_i},\ C_{i,2} := g^{t_i},\ C_{i,3} := (u^{\mathcal{I}_i} h)^{t_i} \quad \forall i \in \{1, \ldots, j\}.$$

EncryptSF($M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}$*)* $\rightarrow \widetilde{\mathrm{CT}}$. The semi-functional encryption algorithm first calls the Encrypt algorithm to obtain a normal ciphertext, $\mathrm{CT} = \{C', C'_0, C'_{i,1}, C'_{i,2}, C'_{i,3}\ \forall i\}$. It then chooses random values $\gamma, \delta \in \mathbb{Z}_N$. It forms the semi-functional ciphertext $\widetilde{\mathrm{CT}}$ as:

$$C := C',\ C_0 := C'_0 \cdot g_2^{\gamma},$$

$$C_{i,1} := C'_{i,1} \cdot g_2^{\delta},\ C_{i,2} := C'_{i,2},\ C_{i,3} := C'_{i,3} \quad \forall i \in \{1, \ldots, j\}.$$

Notice that the additional term g_2^{δ} on $C_{i,1}$ is the *same for each value of i*.

KeyGen($(\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}$*)* $\rightarrow \mathrm{SK}_{\boldsymbol{\mathcal{I}}}$. The key generation algorithm chooses uniformly random values $r_1, \ldots, r_j, y_1, \ldots, y_j$ from $\mathbb{Z}_N$. It also chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ subject to the constraint that $\alpha = \lambda_1 + \lambda_2 + \cdots + \lambda_j$. The secret key is created as:

$$K_{i,0} := g^{\lambda_i} w^{y_i},\ K_{i,1} := g^{y_i},\ K_{i,2} := v^{y_i} (u^{\mathcal{I}_i} h)^{r_i},\ K_{i,3} := g^{r_i} \quad \forall i \in \{1, \ldots, j\}.$$

KeyGenSF($(\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}$*)* $\rightarrow \widetilde{\mathrm{SK}}_{\boldsymbol{\mathcal{I}}}$. The first time this algorithm is called, it chooses random values $\sigma, \psi \in \mathbb{Z}_N$. These values will be stored and used on each invocation of the algorithm.

To create a semi-functional key, the semi-functional key generation algorithm first calls the KeyGen algorithm to obtain a normal key, $\mathrm{SK}_{\boldsymbol{\mathcal{I}}} = \{K'_{i,0}, K'_{i,1}, K'_{i,2}, K'_{i,3}\ \forall i\}$. It then chooses a random value $\tilde{y}_j \in \mathbb{Z}_N$ and creates the semi-functional key as:

$$K_{i,0} := K'_{i,0},\ K_{i,1} := K'_{i,1},\ K_{i,2} := K'_{i,2},\ K_{i,3} := K'_{i,3} \quad \forall i \in \{1, \ldots, j-1\},$$

$$K_{j,0} := K'_{j,0} \cdot (g_2 g_3)^{\psi \tilde{y}_j},\ K_{j,1} := K'_{j,1} \cdot (g_2 g_3)^{\tilde{y}_j},$$

$$K_{j,2} := K'_{j,2} \cdot (g_2 g_3)^{\sigma \tilde{y}_j},\ K_{j,3} := K'_{j,3}.$$

We note that the $\tilde{y}_j$ terms are chosen to be freshly random for each key, while the values σ, ψ are shared by all semi-functional keys. We also note that the exponents modulo p_3 here are uncorrelated from the exponents modulo p_2 by the Chinese Remainder Theorem. It is also important to observe that the semi-functional components (the added terms in G_{p_2} and G_{p_3}) only appear in the *last level* of the key.

Delegate($\mathrm{PP}, \mathrm{SK}, \mathcal{I}_{j+1}$*)* $\rightarrow \mathrm{SK}'$. The delegation algorithm takes in a secret key $\mathrm{SK} = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}\ \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a level $j+1$ identity $\mathcal{I}_{j+1}$. It produces a secret key SK' for $(\mathcal{I}_1, \ldots, \mathcal{I}_{j+1})$ as follows. It chooses $y'_1, \ldots, y'_{j+1}$ and $r'_1, \ldots, r'_{j+1} \in \mathbb{Z}_N$ uniformly at random, $\lambda'_1, \ldots, \lambda'_{j+1} \in \mathbb{Z}_N$ randomly up to the constraint that $\lambda'_1 + \cdots + \lambda'_{j+1} = 0$ and computes:

$$K'_{i,0} := K_{i,0} \cdot g^{\lambda'_i} \cdot w^{y'_i},\ K'_{i,1} := K_{i,1} \cdot g^{y'_i},\ K'_{i,2} := K_{i,2} \cdot v^{y'_i} (u^{\mathcal{I}_i} h)^{r'_i},$$

$$K'_{i,3} := K_{i,3} \cdot g^{r'_i} \; \forall i \in \{1, \ldots, j+1\},$$

where $K_{j+1,1}, K_{j+1,2}, K_{j+1,3}$ are defined to be the identity element in G.

Decryption$(CT, \mathrm{SK}) \rightarrow M$. The decryption algorithm takes in a secret key $\mathrm{SK} = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3} \; \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a ciphertext CT encrypted to $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$. Assuming $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is a prefix of $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$, the message is decrypted as follows. The decryption algorithm computes:

$$B := \prod_{i=1}^{j} \frac{e(C_0, K_{i,0}) e(C_{i,2}, K_{i,2})}{e(C_{i,1}, K_{i,1}) e(C_{i,3}, K_{i,3})}.$$

The message is then computed as $M = C/B$.

Correctness We observe that:

$$B = \prod_{i=1}^{j} \frac{e(g,g)^{s\lambda_i} e(g,w)^{sy_i} e(g,v)^{t_i y_i} e(g, u^{\mathcal{I}_i} h)^{t_i r_i}}{e(w,g)^{sy_i} e(v,g)^{t_i y_i} e(u^{\mathcal{I}_i} h, g)^{t_i r_i}},$$

which is equal to:

$$= \prod_{i=1}^{j} e(g,g)^{s\lambda_i} = e(g,g)^{s\alpha},$$

since $\sum_{i=1}^{j} \lambda_i = \alpha$. Thus, $M = C/B$.

5 Security

We prove that our dual system encryption HIBE scheme has delegation invariance, semi-functional ciphertext invariance, one semi-functional key invariance, and semi-functional security. By theorems 1 and 2, this implies that our HIBE system is secure. More formally, we prove the following theorem:

Theorem 3. *Under Assumptions 1-4, our HIBE system is fully secure.*

Proving delegation invariance, semi-functional ciphertext invariance, and semi-functional security is relatively straightforward, and these proofs can be found in the full version. The truly challenging part of the proof will be proving one semi-functional key invariance, and this is where we introduce our key technical innovations.

5.1 One Semi-functional Key Invariance

The primary challenge in proving one semi-functional key invariance for our system is that the repetition of the same public parameters for each level of the keys and ciphertexts severely limits our ability to simulate properly distributed semi-functional keys and ciphertexts as we are changing the form of the challenge

key. In a typical dual system encryption argument, we must ensure as we are changing the form of one key that the simulator cannot determine the nature of the key for itself. Since the simulator must be prepared to make a semi-functional ciphertext for any identity vector and also must be prepared to use any identity vector for the challenge key, it seems that a simulator could learn for itself whether or not the challenge key is semi-functional by trying to decrypt a semi-functional ciphertext. This potential paradox can be avoided by ensuring that a simulator can only make a nominally semi-functional key, meaning that even if semi-functional terms are present on the challenge key, they will be correlated with the semi-functional ciphertext and cancel out upon decryption. We would then argue that nominality is hidden from an attacker who cannot request keys capable of decrypting.

If we attempt to change the challenge key in our system from normal to semi-functional in a single or very small number of steps using generalized subgroup decision assumptions, then the very limited entropy available in the public parameters seems to prevent us from maintaining the proper distributions of the semi-functional keys and semi-functional ciphertext without revealing nominality. In other words, it appears to be difficult for the simulator to prevent information-theoretic exposure of unwanted correlations between the semi-functional components of the keys and ciphertext it creates.

To overcome this difficulty, we employ a *nested dual system encryption approach* and introduce the concept of *ephemeral semi-functionality*[2]. Instead of trying to directly change the challenge key from normal to semi-functional, we will first change it from normal to *ephemeral semi-functional*. An ephemeral semi-functional key will come from a new distribution which serves as an intermediary stage between the normal and semi-functional distributions. We note that an ephemeral semi-functional key can still correctly decrypt semi-functional ciphertexts, and that its form only differs from a normal key on its last level.

After changing the challenge key from normal to ephemeral semi-functional, we will then change the ciphertext to also be ephemeral semi-functional. Ephemeral semi-functional ciphertexts will come from a new distribution of ciphertexts, and will *not* be decryptable by ephemeral semi-functional keys. This is where we confront the potential paradox of dual system encryption: we will make sure that the simulator can only make challenge key and ciphertext pairs which are *nominally ephemeral semi-functional*, meaning that the distributions of the challenge key and ciphertext will be correlated so that even if the ephemeral semi-functional terms are present in both the key and ciphertext, they will cancel out upon decryption. This correlation will be hidden from an attacker who cannot request a key capable of decrypting the ciphertext.

To accomplish this information-theoretic hiding with such low entropy in our public parameters, we will make a hybrid argument in which we change the ciphertext form one level at a time. Since there are only ephemeral semi-functional

[2] We choose not to include this concept in our abstraction for dual system encryption HIBE, because its use here is motivated by the particular challenge of short public parameters and we imagine dual system encryption HIBE as a broader framework.

terms on one level of one key, it is now sufficient to hide a correlation between one level of the ciphertext and one level of one key: this can be accomplished with the use of a pairwise independent function. Once we have obtained an ephemeral semi-functional challenge key and an ephemeral semi-functional ciphertext, we are able to change the challenge key to be semi-functional in the usual sense and also return the ciphertext to its usual semi-functional state.

Essentially, using ephemeral semi-functionality helps us overcome the challenge presented by low entropy in the public parameters because it allows us to move the information-theoretic argument that nominality is hidden from the attacker to a setting where we are really only concerned with *one key*. Since the other semi-functional keys come from a different distribution, we can prevent them from leaking information about the simulated ephemeral distribution that would break the information-theoretic argument.

We now define the distributions of ephemeral semi-functional keys and ciphertexts. We do this by defining two new algorithms, EncryptESF and KeyGenESF. Like the algorithms EncryptSF and KeyGenSF, these do not need to run in polynomial time (given only their input parameters). We note that the EncryptESF algorithm takes in an additional parameter σ: this is because the ciphertexts it produces will share the value σ with the semi-functional keys created by KeyGenSF. As in the original semi-functional algorithms, we let g_2 denote a generator of G_{p_2} and g_3 denote a generator of G_{p_3}.

EncryptESF$(M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}, \sigma) \rightarrow \widetilde{\mathrm{CT}}_E$. The ephemeral semi-functional encryption algorithm first calls the Encrypt algorithm to obtain a normal ciphertext $\mathrm{CT} = \{C', C'_0, C'_{i,1}, C'_{i,2}, C'_{i,3}\ \forall i \in \{1, \ldots, j\}\}$. It then chooses random values $\gamma, \delta, a', b', t_1, \ldots, t_j \in \mathbb{Z}_N$ and forms the ephemeral semi-functional ciphertext $\widetilde{\mathrm{CT}}_E$ as:

$$C := C', C_0 := C'_0 \cdot g_2^{\gamma},$$

$$C_{i,1} := C'_{i,1} \cdot g_2^{\delta} \cdot g_2^{\sigma t_i},\ C_{i,2} := C'_{i,2} \cdot g_2^{t_i},\ C_{i,3} := C'_{i,3} \cdot g_2^{(a'\mathcal{I}_i + b')t_i} \quad \forall i \in \{1, \ldots, j\}.$$

KeyGenESF$((\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}, \sigma) \rightarrow \widetilde{\mathrm{SK}}_E$. The ephemeral semi-functional key generation algorithm first calls the KeyGen algorithm to obtain a normal key $\mathrm{SK} = \{K'_{i,0}, K'_{i,1}, K'_{i,2}, K'_{i,3}\ \forall i \in \{1, \ldots, j\}\}$. It chooses random values $\tilde{r}_1, \tilde{r}_2 \in \mathbb{Z}_N$ and forms the ephemeral semi-functional key $\widetilde{\mathrm{SK}}_E$ as:

$$K_{i,0} := K'_{i,0},\ K_{i,1} := K'_{i,1},\ K_{i,2} := K'_{i,2},\ K_{i,3} := K'_{i,3}\ \forall i \in \{1, \ldots, j-1\},$$

$$K_{j,0} := K'_{j,0},\ K_{j,1} = K'_{j,1},\ K_{j,2} := K'_{j,2} \cdot (g_2 g_3)^{\tilde{r}_1},\ K_{j,3} := K'_{j,3} \cdot (g_2 g_3)^{\tilde{r}_2}.$$

We note that an ephemeral semi-functional key can decrypt a semi-functional ciphertext, but cannot decrypt an ephemeral semi-functional ciphertext, while an ephemeral semi-functional ciphertext can only be decrypted by normal keys.

Sequence of Games. We prove one semi-functional key invariance of our dual system encryption HIBE scheme via a hybrid argument over the following sequence of games. We begin with Game HIBE_0, where the ciphertext is semi-functional and the challenge key is normal. We will end with Game HIBE_1, where the ciphertext is semi-functional and the challenge key is semi-functional. We define the following intermediary games. In these games, the distributions of the challenge key and ciphertext vary, while the distribution of the requested normal and semi-functional keys are the same as in Games HIBE_0 and HIBE_1.

Game HIBE$'_0$. This game is exactly like Game HIBE_0, except for the added restriction that the last component of the challenge key identity vector cannot be equal to any of the components of the challenge ciphertext identity vector modulo p_3 (note that we were already requiring this modulo N - now we make the stronger requirement that the identities must remain unequal when we reduce modulo p_3). (This added restriction will be needed to apply pairwise independence arguments in $\mathbb{Z}_{p_3}$.)

Game EK. In Game EK, the ciphertext is still semi-functional, and the challenge key is now ephemeral semi-functional. We retain the added restriction on the identities modulo p_3.

Game EC. In Game EC, *both* the ciphertext and challenge are ephemeral semi-functional. We retain the added restriction on the identities modulo p_3.

Game HIBE$'_1$. This game is exactly like the Game HIBE_1, but with the added restriction on the identities modulo p_3.

In the full version, we prove that we can transition from Game HIBE_0 to Game HIBE'_0, to Game EK, to Game EC, to Game HIBE'_1, and finally to Game HIBE_1 without the attacker's advantage changing by a non-negligible amount.

6 Key-Policy Attribute-Based Encryption

We now present our construction for KP-ABE. Our public parameters consist of a constant number of elements from a bilinear group of composite order N, while our attribute universe is $\mathbb{Z}_N$. Ciphertexts in our system are associated with sets of attributes, while secret keys are associated with LSSS access matrices. Our construction is closely related to our HIBE construction. The main changes are that attributes have now replaced identities, and the master secret key α is now shared according to the LSSS matrix, instead of as a sum. We follow the convention that to share a value α, one employs a vector $\boldsymbol{\alpha}$ with first coordinate equal to α, and the shares are obtained by multiplying the rows of the LSSS matrix by the sharing vector $\boldsymbol{\alpha}$. A subset of rows is capable of reconstructing the shared secret if and only if their span includes the vector $(1, 0, \ldots, 0)$.

6.1 Construction

Setup$(\lambda) \to \text{PP}, \text{MSK}$. The setup algorithm takes in the security parameter λ and chooses a suitable bilinear group G of order $N = p_1p_2p_3$, a product of three distinct primes. It chooses $\alpha \in \mathbb{Z}_N$ uniformly randomly, and also chooses uniformly random elements g, u, h, v, w from the subgroup G_{p_1}. It sets the public parameters as:

$$\text{PP} := \{N, G, g, u, h, v, w, e(g,g)^\alpha\}.$$

The MSK is α, and the universe U of attributes is $\mathbb{Z}_N$.

Encrypt$(M, S \subseteq U, \text{PP}) \to \text{CT}$. The encryption algorithm takes in a message M, a set of attributes S, and the public parameters. We let ℓ denote the size of the set S, and we let $s_1, \ldots, s_\ell \in \mathbb{Z}_N$ denote the elements of S. The encryption algorithm chooses uniformly random values $s, t_1, \ldots, t_\ell \in \mathbb{Z}_N$ and computes the ciphertext as:

$$C := Me(g,g)^{\alpha s},\ C_0 := g^s,$$

$$C_{s_i,1} := w^s v^{t_i},\ C_{s_i,2} := g^{t_i},\ C_{s_i,3} := (u^{s_i}h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

(We also assume the set of S is given as part of the ciphertext.)

KeyGen$(\text{MSK}, \text{PP}, (A, \rho)) \to \text{SK}$. The key generation algorithm takes in the master secret key α, the public parameters, and a LSSS matrix (A, ρ), where A is an $n \times m$ matrix over $\mathbb{Z}_N$, and ρ maps each row of A to an attribute in $\mathbb{Z}_N$. The key generation algorithm chooses a random vector $\boldsymbol{\alpha} \in \mathbb{Z}_N^m$ with first coordinate equal to α and random values $r_1, \ldots, r_n, y_1, \ldots, y_n \in \mathbb{Z}_N$. For each $x \in \{1, \ldots, n\}$, we let A_x denote the x^{th} row of A, and we let $\rho(x)$ denote that attribute associated with this row by the mapping ρ. We let $\lambda_x := A_x \cdot \boldsymbol{\alpha}$ denote the share associated with the row A_x of A. The secret key is formed as[3]:

$$K_{x,0} := g^{\lambda_x} w^{y_x}, K_{x,1} := g^{y_x}, K_{x,2} := v^{y_x}(u^{\rho(x)}h)^{r_x}, K_{x,3} := g^{r_x}\ \forall x \in \{1, \ldots, n\}.$$

Decrypt$(\text{SK}, \text{CT}) \to M$. The decryption algorithm takes in a ciphertext CT for attribute set S and a secret key SK for access matrix (A, ρ). If the attributes of the ciphertext satisfy the policy of the secret key, then it will compute the message M as follows. First, it computes constants ω_x such that $\sum_{\rho(x) \in S} \omega_x A_x = (1, 0, \ldots, 0)$. It then computes:

$$B = \prod_{\rho(x) \in S} \left(\frac{e(C_0, K_{x,0}) e(C_{\rho(x),2}, K_{x,2})}{e(C_{\rho(x),1}, K_{x,1}) e(C_{\rho(x),3}, K_{x,3})} \right)^{\omega_x},\ M = C/B.$$

[3] We also assume the access matrix (A, ρ) is given as part of the key.

Correctness. We observe that:

$$B = \prod_{\rho(x)\in S} \left(\frac{e(g,g)^{s\lambda_x} e(g,w)^{sy_x} e(g,v)^{t_{\rho(x)} y_x} e(g, u^{\rho(x)} h)^{t_{\rho(x)} r_x}}{e(w,g)^{sy_x} e(v,g)^{t_{\rho(x)} y_x} e(u^{\rho(x)} h, g)^{t_\rho(x) r_x}} \right)^{\omega_x},$$

$$= \prod_{\rho(x)\in S} \left(e(g,g)^{s\lambda_x} \right)^{\omega_x} = e(g,g)^{s\alpha}.$$

This shows that $M = C/B$.

6.2 Security

We prove that our system is selectively secure using a similar strategy to our proof of adaptive security for our HIBE system (the formal definition for selective security in the KP-ABE setting can be found in the full version). We were able to achieve adaptive security in the HIBE setting because we could assume that, regardless of what identity vector was chosen for the challenge ciphertext, each requested key would have a final identity component which would not match any components of the challenge ciphertext. This allowed us to put our semi-functional components only on the last level of the key, which was crucial to preserving the appearance of randomness via our pairwise independence argument in the middle stages of the proof. In the adaptive KP-ABE setting, we only know that the policy of a requested key will fail to be satisfied by the attribute set of the ciphertext, but we do not know *how* it will fail to be satisfied. In other words, for keys which are requested by the attacker before the challenge ciphertext, we do not know which rows of the keys will correspond to attributes which are not in the challenge ciphertext. This leaves us in a bind - we do not know where to put the semi-functional terms. If we try to put semi-functional terms on each row, we will not be able to make the semi-functional terms appear suitably random in the attacker's view. If we put the semi-functional terms on too few rows, we will not achieve a meaningful kind of semi-functionality.

This problem is solved by moving to the selective security model, which forces the attacker to reveal the attribute set of the challenge ciphertext at the very start of the game. This means that when the simulator is faced with a key request, it already knows which rows of the key correspond to attributes which are absent from the ciphertext, and it can place the semi-functional terms exactly on these rows. We must add an additional hybrid to our proof strategy here so that we can change the rows of a key from normal to semi-functional one at a time. The proof of the following theorem, as well as discussion of delegation capabilities for our ABE scheme, can be found in the full version.

Theorem 4. *Under Assumptions 1-4, our KP-ABE system is selectively secure.*

References

1. Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

2. Agrawal, S., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 98–115. Springer, Heidelberg (2010)
3. Al-Riyami, S., Malone-Lee, J., Smart, N.: Escrow-free encryption supporting cryptographic workflow. Int. J. Inf. Sec. 5, 217–229 (2006)
4. Bagga, W., Molva, R., Crosta, S.: Policy-based encryption schemes from bilinear pairings. In: ASIACCS, p. 368 (2006)
5. Barbosa, M., Farshim, P.: Secure cryptographic workflow in the standard model. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 379–393. Springer, Heidelberg (2006)
6. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
7. Bellare, M., Waters, B., Yilek, S.: Identity-based encryption secure against selective opening attack. In: TCC 2011 (2011)
8. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: Proceedings of the IEEE Symposium on Security and Privacy, pp. 321–334
9. 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)
10. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
11. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
12. 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)
13. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: FOCS, pp. 647–657 (2007)
14. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
15. Bradshaw, R., Holt, J., Seamons, K.: Concealing complex policies with hidden credentials. In: ACM Conference on Computer and Communications Security, pp. 146–157 (2004)
16. 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)
17. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
18. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
19. Chase, M., Chow, S.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM Conference on Computer and Communications Security, pp. 121–130 (2009)
20. Chatterjee, S., Sarkar, P.: Generalization of the selective-ID security model for HIBE protocols. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 241–256. Springer, Heidelberg (2006)
21. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: ACM Conference on Computer and Communications Security, pp. 456–465 (2007)

22. Chow, S., Dodis, Y., Rouselakis, Y., Waters, B.: Practical leakage-resilient identity-based encryption from simple assumptions
23. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 26–28. Springer, Heidelberg (2001)
24. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
25. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 197–206 (2008)
26. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
27. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
28. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute based encryption for fine-grained access control of encrypted data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
29. 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)
30. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
31. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (Hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
32. Lewko, A., Rouselakis, Y., Waters, B.: Achieving leakage resilience through dual system encryption. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 70–88. Springer, Heidelberg (2011)
33. Lewko, A., Waters, B.: Unbounded HIBE and attribute-based encryption. Cryptology ePrint Archive, Report 2011/049 (2011), http://eprint.iacr.org/
34. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
35. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: VLDB, pp. 898–909 (2003)
36. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
37. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
38. Ostrovksy, R., Sahai, A., Waters, B.: Attribute based encryption with non-monotonic access structures. In: ACM Conference on Computer and Communications Security, pp. 195–203 (2007)

39. Pirretti, M., Traynor, P., McDaniel, P., Waters, B.: Secure attribute-based systems. In: ACM Conference on Computer and Communications Security, pp. 99–112 (2006)
40. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
41. 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)
42. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
43. Smart, N.: Access control using pairing based cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 111–121. Springer, Heidelberg (2003)
44. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
45. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
46. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

Decentralizing Attribute-Based Encryption

Allison Lewko* and Brent Waters**

University of Texas Austin
{alewko,bwaters}@cs.utexas.edu

Abstract. We propose a Multi-Authority Attribute-Based Encryption (ABE) system. In our system, any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. A party can simply act as an ABE authority by creating a public key and issuing private keys to different users that reflect their attributes. A user can encrypt data in terms of any boolean formula over attributes issued from any chosen set of authorities. Finally, our system does not require any central authority.

In constructing our system, our largest technical hurdle is to make it collusion resistant. Prior Attribute-Based Encryption systems achieved collusion resistance when the ABE system authority "tied" together different components (representing different attributes) of a user's private key by randomizing the key. However, in our system each component will come from a potentially different authority, where we assume no coordination between such authorities. We create new techniques to tie key components together and prevent collusion attacks between users with different global identifiers.

We prove our system secure using the recent dual system encryption methodology where the security proof works by first converting the challenge ciphertext and private keys to a semi-functional form and then arguing security. We follow a recent variant of the dual system proof technique due to Lewko and Waters and build our system using bilinear groups of composite order. We prove security under similar static assumptions to the LW paper in the random oracle model.

1 Introduction

Traditionally, we view encryption as a mechanism for a user, Alice, to confidentially encode data to a target recipient, Bob. Alice encrypts the data under the recipient's public key such that only Bob, with knowledge of his private key, can decrypt it.

* Supported by National Defense Science and Engineering Graduate Fellowship.
** Supported by NSF CNS-0915361, and CNS-0952692, the MURI program under AFOSR Grant No: FA9550-08-1-0352. Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), a Google Faculty Research award, and the Alfred P. Sloan Foundation.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 568–588, 2011.

However, in many applications, we find we need to share data according to an encryption policy without prior knowledge of who will be receiving the data. Suppose an administrator needs to encrypt a junior faculty member's performance review for all senior members of the computer science department or anyone in the dean's office. The administrator will want to encrypt the review with the access policy ("Computer Science" **AND** "Tenured") **OR** "Dean's Office". In this system, only users with attributes (credentials) that match this policy should be able to decrypt the document. The key challenge in building such systems is to realize security against *colluding* users. For instance, the encrypted records should not be accessible to a pair of unauthorized users, where one has the two credentials of "Tenured" and "Chemistry" and the other one has the credential of "Computer Science". Neither user is actually a tenured faculty member of the Computer Science Department.

Sahai and Waters [45] proposed a solution to the above problem that they called Attribute-Based Encryption (ABE). In an ABE system, a party encrypting data can specify access to the data as a boolean formula over a set of attributes. Each user in the system will be issued a private key from an authority that reflects their attributes (or credentials). A user will be able to decrypt a ciphertext if the attributes associated with their private key satisfy the boolean formula ascribed to the ciphertext. A crucial property of ABE systems is that they resist collusion attacks as described above.

Since the introduction of Attribute-Based Encryption, several works [8,30,44,29,23,54,21,22,37] have proposed different ABE systems and applications. In almost all ABE proposals, private keys were issued by one central authority that would need to be in a position to verify all the attributes or credentials it issued for each user in the system. These systems can be utilized to share information according a policy over attributes issued within a domain or organization, however, in many applications a party will want to share data according to a policy written over attributes or credentials issued across different trust domains and organizations. For instance, a party might want to share medical data only with a user who has the attribute of "Doctor" issued by a medical organization and the attribute "Researcher" issued by the administrators of a clinical trial. On a commercial application, two corporations such as Boeing and General Electric might both issue attributes as part of a joint project. Using current ABE systems for these applications can be problematic since one needs a *single* authority that is both able to verify attributes across different organizations and issue private keys to every user in the system.

A Simple Approach and Its Limitations. We would like to realize an encryption system where a party can encrypt data for a policy written over attributes issued by different authorities. A user in the system should be able to decrypt if their attributes (possibly issued by multiple authorities) satisfy the policy specified by the ciphertext. In addition, the system should be able to express complex policies and not require coordination amongst the authorities.

An initial step towards this goal is to simply "engineer" a system by utilizing existing (Ciphertext-Policy) Attribute-Based Encryption schemes along with standard signature schemes. In this proposal, a designated "central authority" will first create a set of public parameters. Then any party wishing to become an "authority" will create a signature verification key VK that will be associated with them. A user in the system with a globally verifiable identifier GID will collect private keys for attributes that it has from different authorities.

Suppose that a user GID can demonstrate attributes X_1, X_2 to the authority with verification key VK and attribute Y to the authority with verification key VK$'$. The user will obtain his secret key as follows. First, he will obtain a signature of GID, (X_1, X_2) that verifies under VK and a signature of GID, Y under VK$'$ from the two respective authorities (and any other authorities). Next, the user will present these signature and verification key pairs to the central authority. The central authority will first check that each signature verifies under the claimed verification key and that each signature is on the *same* global identifier. Using an existing ABE algorithm, it will then issue an attribute for each verification key and attribute pair. In the above example, the user will get a key with attributes "VK, X_1", "VK, X_2", and "VK$'$, Y". We note that the operation of the central authority is agnostic to the meaning of these verification keys and attributes; indeed, it will not need to have any a priori relationship with any of the authorities.

This simple system enjoys multiple benefits. Since encryption simply uses a prior ABE system, we can achieve the same level of expressiveness and write a policy in terms of any boolean formula. The system also requires minimum coordination between separate authorities. Any party can choose to be an authority by creating and publishing a verification key coupled with a list of attributes it will manage. Different authorities will not need to coordinate or even be aware of each other. There are several issues that will need to be dealt with in any larger system, such as the choice of an appropriate global identifier[1] or a party's decision as to which authority it trusts to issue private keys related to certain attributes. For instance, one might encrypt a policy using Experian's verification key to attest for the attribute of a good FICO (credit) score.

The major drawback of this simple engineered approach is that it requires a designated central authority. This authority must be globally trustworthy, since its failure will compromise the entire system. If we aim to build a large or even global scale system, this authority will become a common bottleneck. Spreading a central authority's keys over several machines to alleviate performance pressures might simultaneously increase the risk of key exposure.

A few works have attempted to create new cryptographic solutions to the multi-authority ABE problem. Chase [21] proposed an interesting solution that introduced the concept of using a global identifier as a "linchpin" for tying

[1] The idea of applying a global identifier in the context of multi-authority ABE was first proposed by Chase [21]. Chase adapted the concept from its use in anonymous credential systems [19]. One previously suggested candidate for a global identifier is a user's social security number.

users' keys together. Her system relied on a central authority and was limited to expressing a strict "AND" policy over a *pre-determined* set of authorities. Therefore a party encrypting would be much more limited than in the simple engineering approach outlined above. Müller, Katzenbeisser, and Eckert [42,43] give a different system with a centralized authority that realizes any LSSS access structure. Their construction builds on the Waters system [54]; their proof is limited to non-adaptive queries. The system achieves roughly the same functionality as the engineering approach above, except one can still acquire attributes from additional authorities without revisiting the central authority. Chase and Chow [22] showed how to remove the central authority using a distributed PRF; however, the same limitations of an AND policy of a determined set of authorities remained. Lin et. al. [40] give a threshold based scheme that is also somewhat decentralized. The set of authorities is fixed ahead of time, and they must interact during the system setup. The system is only secure up to collusions of m users, where m is a system parameter chosen at setup such that the cost of operations and key storage scales with m.

Our Contribution. We propose a new multi-authority Attribute-Based Encryption system. In our system, any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. (These will be created during a trusted setup.) A party can simply act as an authority by creating a public key and issuing private keys to different users that reflect their attributes. Different authorities need not even be aware of each other. We use the Chase [21] concept of global identifiers to "link" private keys together that were issued to the same user by different authorities. A user can encrypt data in terms of any boolean formula[2] over attributes issued from any chosen set of authorities.

Finally, our system does not require any central authority. We thus avoid the performance bottleneck incurred by relying on a central authority, which makes our system more scalable. We also avoid placing absolute trust in a single designated entity which must remain active and uncorrupted throughout the lifetime of the system. This is a crucial improvement for efficiency as well as security, since even a central authority that remains uncorrupted may occasionally fail for benign reasons, and a system that constantly relies on its participation will be forced to remain stagnant until it can be restored. In our system, authorities can function entirely independently, and the failure or corruption of some authorities will not affect the operation of functioning, uncorrupted authorities. This makes our system more robust then the other approaches outlined above.

Challenges and Our Techniques. In constructing our system, our central technical hurdle is to make it collusion resistant. Prior Attribute-Based Encryption systems achieved collusion resistance when the ABE system authority "tied" together different components (representing different attributes) of a user's private key by randomizing the key. Such randomization would make the different key

[2] Our system actually generalizes to handle any policy that can be expressed as a Linear Secret Sharing Scheme (LSSS) or equivalently a monotone span program.

components compatible with each other, but not with the parts of a key issued to another user.

In our setting, we want to satisfy the simultaneous goals of autonomous key generation and collusion resistance. The requirement of autonomous key generation means that established techniques for key randomization cannot be applied since there is no one party to compile all the pieces together. Furthermore, in our system each component may come from a different authority, where such authorities have no coordination and are possibly not even aware of each other and there is no preset access structure[3].

To overcome this, we develop a novel technique for tying a user's key components together and preventing collusion attacks between users with different global identifiers. At a high level, instead of relying on one key generation call to tie all key components together, we will use a hash function on the user's global identity, GID to manage collusion resistance across multiple key generations issued by different authorities.

In our system, we define a hash function H (modeled as a random oracle) that hashes each identity to a (bilinear) group element. We will use the group element output from the hash function $H(\text{GID})$ as the linchpin to tie keys together. Tying keys together in this manner is more challenging than in the single authority case. Our main idea is to structure the decryption mechanism at each satisfied node 'x' in the access tree such that a user will recover a target group element of the form $e(g,g)^{\lambda_x} \cdot e(g, H(\text{GID}))^{w_x}$. This group element first contains a secret share λ_x of a secret s in the exponent, and these shares can be combined to recover the message. However, these will each be "blinded" by a share w_x which is a share of 0 in the exponent with base $e(g, H(\text{GID}))$. This structure allows for the decryption algorithm to both reconstruct the main secret and to unblind it in parallel. If a user with a particular identifier GID satisfies the access tree, he can reconstruct s in the exponent by raising the group elements to the proper exponents. However, this operation will simultaneously reconstruct the share of 0 and thus the $e(g, H(\text{GID}))$ terms will cancel out. Intuitively, if two users with different global identifiers GID, GID' attempt to collude, the cancelation will not work since the w_x shares will have different bases.

We prove our system secure using the recent dual system encryption methodology [53], where the security proof works by first converting the challenge ciphertexts and private keys to a semi-functional form and then arguing security. We follow a recent variant of the dual system proof technique due to Lewko and Waters [39] and build our system using bilinear groups of composite order. The absence of coordination between the authorities also introduces a new technical challenge in applying the dual system encryption methodology. Due to the decentralized nature of user's keys, the techniques employed in [37] to achieve full security for single authority ABE using dual system encryption are insufficient. We overcome this by using two semi-functional subgroups instead of one, and switching between these allows us to defeat the information-theoretic problem

[3] Prior works [21,22] assumed coordination ahead of time between different authorities and required a limited access structure.

which is naturally encountered if one simply tries to apply the previous techniques. We prove security under similar assumptions to the LW paper in the random oracle model.

Related Work. Several of the roots of Attribute-Based Encryption can be traced back to Identity Based Encryption (IBE), proposed by Shamir [46]. The first IBE schemes were constructed by Boneh and Franklin [13] and Cocks [24]. These initial systems were proven secure in the random oracle model. Other standard model solutions followed [20,9,10,52,27], along with extensions to the hierarchical IBE setting [34,28,11].

Attribute-based encryption was introduced by Sahai and Waters [45]. Subsequently, Goyal, Pandey, Sahai, and Waters [30] formulated two complimentary forms of ABE: Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Key-Policy Attribute-Based Encryption (KP-ABE). In a CP-ABE system, keys are associated with sets of attributes and ciphertexts are associated with access policies. In a KP-ABE system, the situation is reversed: keys are associated with access policies and ciphertexts are associated with sets of attributes. Since then, several different ABE systems have been proposed [8,21,23,29,44,54,22], as well as related systems [14,2]. The problem of building ABE systems with multiple authorities was proposed by Sahai and Waters and first considered by Chase [21] and Chase and Chow [22]. Another interesting direction is the construction of "anonymous" or predicate encryption systems [36,49,17,12,1,47,37] where in addition to the data the encryption policy or other properties are hidden. Other works have discussed similar problems without addressing collusion resistance [3,4,5,18,41,51]. In these systems, the data encryptor specifies an access policy such that a set of users can decrypt the data only if the union of their credentials satisfies the access policy.

Until recently, all ABE systems were proven secure in the selective model where an attacker needed to declare the structure of the challenge ciphertext *before* seeing the public parameters. Recently, Lewko, Okamoto, Sahai, Takashima and Waters [37] solved the open problem by giving the first fully secure Attribute-Based Encryption systems. Their system applied the dual system encryption methodology introduced by Waters [53] and techniques used by Lewko and Waters [39]. Our proof uses some techniques from Lewko et. al. [37], but faces new challenges from the multi-authority setting.

Organization. In Section 2, we formally define multi-authority CP-ABE systems and their security. In Section 3, we give our complexity assumptions. In Section 4, we present our multi-authority CP-ABE system and outline the proof of its security. In Section 5, we discuss possible extensions of our results.

2 Multi-authority CP-ABE

Here we give the necessary background on multi-authority CP-ABE schemes and their security definition. For background on access structures, linear secret-sharing schemes, and composite order bilinear groups, see the full version of this paper [38].

A multi-authority Ciphertext-Policy Attribute-Based Encryption system is comprised of the following five algorithms:

Global Setup(λ*)* $\rightarrow$ GP. The global setup algorithm takes in the security parameter λ and outputs global parameters GP for the system.

*Authority Setup(*GP*)* $\rightarrow$ SK, PK. Each authority runs the authority setup algorithm with GP as input to produce its own secret key and public key pair, SK, PK.

Encrypt($M, (A, \rho)$, GP, $\{\text{PK}\}$*)* $\rightarrow$ CT. The encryption algorithm takes in a message M, an access matrix (A, ρ), the set of public keys for relevant authorities, and the global parameters. It outputs a ciphertext CT.

*KeyGen(*GID, GP, i, SK*)* $\rightarrow$ $\text{K}_{i,\text{GID}}$. The key generation algorithm takes in an identity GID, the global parameters, an attribute i belonging to some authority, and the secret key SK for this authority. It produces a key $\text{K}_{i,\text{GID}}$ for this attribute, identity pair.

*Decrypt(*CT, GP, $\{\text{K}_{i,\text{GID}}\}$*)* $\rightarrow$ M. The decryption algorithm takes in the global parameters, the ciphertext, and a collection of keys corresponding to attribute, identity pairs all with the same fixed identity GID. It outputs either the message M when the collection of attributes i satisfies the access matrix corresponding to the ciphertext. Otherwise, decryption fails.

Definition 1. *A multi-authority CP-ABE system is said to be correct if whenever* GP *is obtained from the global setup algorithm,* CT *is obtained from the encryption algorithm on the message* M*, and* $\{\text{K}_{i,\text{GID}}\}$ *is a set of keys obtained from the key generation algorithm for the same identity* GID *and for a set of attributes satisfying the access structure of the ciphertext,* $\mathit{Decrypt}(\text{CT}, \text{GP}, \{\text{K}_{i,\text{GID}}\}) = M$.

2.1 Security Definition

We define security for multi-authority Ciphertext-Policy Attribute-Based Encryption systems by the following game between a challenger and an attacker. We assume that adversaries can corrupt authorities only statically, but key queries are made adaptively. A static corruption model is also used by Chase [21] and Chase and Chow [22], but we note that our model additionally allows the adversary to choose the public keys of the corrupted authorities for itself, instead of having these initially generated by the challenger.

We let S denote the set of authorities and U denote the universe of attributes. We assume each attribute is assigned to one authority (though each authority may control multiple attributes). In practice, we can think of an attribute as being the concatenation of an authority's public key and a string attribute. This will ensure that if multiple authorities choose the same string attribute, these will still correspond to distinct attributes in the system.

Setup. The global setup algorithm is run. The attacker specifies a set $S' \subseteq S$ of corrupt authorities. For good (non-corrupt) authorities in $S - S'$, the challenger obtains public key, private key pairs by running the authority setup algorithm, and gives the public keys to the attacker.

Key Query Phase 1. The attacker makes key queries by submitting pairs (i, GID) to the challenger, where i is an attribute belonging to a good authority and GID is an identity. The challenger responds by giving the attacker the corresponding key, $\text{K}_{i,\text{GID}}$.

Challenge Phase. The attacker must specify two messages, M_0, M_1, and an access matrix (A, ρ). The access matrix must satisfy the following constraint. We let V denote the subset of rows of A labeled by attributes controlled by corrupt authorities. For each identity GID, we let V_{GID} denote the subset of rows of A labeled by attributes i for which the attacker has queried (i, GID). For each GID, we require that the subspace spanned by $V \cup V_{\text{GID}}$ must not include $(1, 0, \ldots, 0)$. (In other words, the attacker cannot ask for a set of keys that allow decryption, in combination with any keys that can obtained from corrupt authorities.) The attacker must also give the challenger the public keys for any corrupt authorities whose attributes appear in the labeling ρ. The challenger flips a random coin $\beta \in \{0, 1\}$ and sends the attacker an encryption of M_β under access matrix (A, ρ).

Key Query Phase 2. The attacker may submit additional key queries (i, GID), as long as they do not violate the constraint on the challenge matrix (A, ρ).

Guess. The attacker must submit a guess β' for β. The attacker wins if $\beta = \beta'$. The attacker's advantage in this game is defined to be $Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 2. *A multi-authority Ciphertext-Policy Attribute-Based Encryption system is secure (against static corruption of authorities) if all polynomial time attackers have at most a negligible advantage in this security game.*

2.2 Transformation from One-Use Multi-authority CP-ABE

In the full version of this paper, we show how to construct a fully secure multi-authority CP-ABE system where attributes are used multiple times in an access matrix from a fully secure multi-authority CP-ABE system where attributes are used only once. We do this with a simple encoding technique. This same transformation was employed by [37] for (single authority) CP-ABE.

3 Our Assumptions

We now state the complexity assumptions that we will rely on to prove security for our system. These assumptions are formulated for a bilinear group G of order $N = p_1 p_2 p_3$, a product of 3 primes. We let $e : G \times G \to G_T$ denote the bilinear

map. For background on these groups, see the full version. We note that these are similar to the assumptions used in [39,37]. While the fourth assumption is new, the first three are instances of the class of General Subgroup Decision Assumptions described in [7]. This class is defined as follows: in a bilinear group of order $N = p_1 p_2 \ldots p_n$, there is a subgroup of order $\prod_{i \in S} p_i$ for each subset $S \subseteq \{1, \ldots, n\}$. We let S_0, S_1 denote two distinct subsets. We then assume it is hard to distinguish a random element from the subgroup associated with S_0 from a random element of the subgroup associated with S_1, even if one is given random elements from subgroups associated with several subsets Z_i which each satisfy either that $S_0 \cap Z_i = \emptyset = S_1 \cap Z_i$ or $S_0 \cap Z_i \neq \emptyset \neq S_1 \cap Z_i$. We prove our four specific assumptions are generically secure in the full version, under the assumption that it is hard to find a nontrivial factor of the group order N.

In the assumptions below, we let G_{p_1}, e.g., denote the subgroup of order p_1 in G. We note that if $g_i \in G_{p_i}$ and $g_j \in G_{p_j}$ for $i \neq j$, then $e(g_i, g_j) = 1$. When we write $g_1 \xleftarrow{R} G_{p_1}$, we mean that g_1 is chosen to be a random generator of G_{p_1} (so it is not the identity element). Similarly, when we write $T_1 \xleftarrow{R} G$, we mean that T_1 is chosen to be a random generator of G (this is not quite the same as a uniformly random element, but the distributions are negligibly close).

Assumption 1 (Subgroup decision problem for 3 primes). Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1},$$
$$D = (\mathbb{G}, g_1),$$
$$T_1 \xleftarrow{R} G,\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

We note that T_1 can be written (uniquely) as the product of an element of G_{p_1}, an element of G_{p_2}, and an element of G_{p_3}. We refer to these elements as the "G_{p_1} part of T_1", the "G_{p_2} part of T_1", and the "G_{p_3} part of T_1" respectively. We will use this terminology in our proofs.

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1, X_1 \xleftarrow{R} G_{p_1}, X_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3},$$

$$D = (\mathbb{G}, g_1, g_3, X_1 X_2),$$

$$T_1 \xleftarrow{R} G_{p_1},\ T_2 \xleftarrow{R} G_{p_1 p_2}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 4. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e), \xleftarrow{R} \mathcal{G},$$

$$g_1, X_1 \xleftarrow{R} G_{p_1}, Y_2 \xleftarrow{R} G_{p_2}, X_3, Y_3 \xleftarrow{R} G_{p_3},$$

$$D = (\mathbb{G}, g_1, X_1 X_3, Y_2 Y_3),$$

$$T_1 \xleftarrow{R} G_{p_1 p_2}, T_2 \xleftarrow{R} G_{p_1 p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 5. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 4. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e), \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1}, g_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3}, a, b, c, d \xleftarrow{R} \mathbb{Z}_N,$$

$$D = (\mathbb{G}, g_1, g_2, g_3, g_1^a, g_1^b g_3^b, g_1^c, g_1^{ac} g_3^d),$$

$$T_1 = e(g_1, g_1)^{abc}, T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be:

$$Adv4_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 6. *We say that $\mathcal{G}$ satisfies Assumption 4 if $Adv4_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

4 Our Multi-authority CP-ABE System

We now present our one-use multi-authority ciphertext-policy attribute based encryption system. We use a composite order bilinear group G, where the group order is a product of three primes: $N = p_1p_2p_3$. Except for the random oracle H which maps identities to random group elements, the entire system is confined to the subgroup G_{p_1} in G. The subgroups G_{p_2} and G_{p_3} are used in our security proof, which employs the dual system encryption technique. In a dual system, keys and ciphertexts can be either normal or semi-functional. Normal keys and ciphertexts in our system will be contained in the subgroup G_{p_1}, while semi-functional keys and ciphertexts will involve elements of the subgroups G_{p_2} and G_{p_3}. In other words, the subgroups G_{p_2} and G_{p_3} form the semi-functional space, which is orthogonal to the subgroup G_{p_1} where the normal keys and ciphertexts reside.

To prevent collusion attacks, our system uses the global identity to "tie" together the various attributes belonging to a specific user so that they cannot be successfully combined with another's user's attributes in decryption. More specifically, the encryption algorithm blinds the message M with $e(g_1, g_1)^s$, where g_1 is a generator of the subgroup G_{p_1}, and s is a randomly chosen value in $\mathbb{Z}_N$. The value s is then split into shares λ_x according to the LSSS matrix, and the value 0 is split into shares ω_x. The decryptor must recover the blinding factor $e(g_1, g_1)^s$ by pairing their keys for attribute, identity pairs (i, GID) with the ciphertext elements to obtain the shares of s. In doing so, the decryptor will introduce terms of the form $e(g_1, H(\mathrm{GID}))^{\omega_x}$. If the decryptor has a satisfying set of keys with the same identity GID, these additional terms will cancel from the final result, since the ω_x's are shares of 0. If two users with different identities GID and GID$'$ attempt to collude and combine their keys, then there will be some terms of the form $e(g_1, H(\mathrm{GID}))^{\omega_x}$ and some terms of the form $e(g_1, H(\mathrm{GID}'))^{\omega_{x'}}$, and these will not cancel with each other, thereby preventing the recovery of $e(g_1, g_1)^s$.

4.1 Construction

Global Setup$(\lambda) \rightarrow \mathrm{GP}$. In the global setup, a bilinear group G of order $N = p_1p_2p_3$ is chosen. The global public parameters, GP, are N and a generator g_1 of G_{p_1}. In addition, the description of a hash function $H : \{0,1\}^* \rightarrow G$ that maps global identities GID to elements of G is published. We will model H as a random oracle.

Authority Setup$(\mathrm{GP}) \rightarrow \mathrm{PK}, \mathrm{SK}$. For each attribute i belonging to the authority, the authority chooses two random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and publishes $\mathrm{PK}_j = \{e(g_1, g_1)^{\alpha_i}, g_1^{y_i} \forall i\}$ as its public key. It keeps $\mathrm{SK} = \{\alpha_i, y_i \forall i\}$ as its secret key.

Encrypt$(M, (A, \rho), \mathrm{GP}, \{\mathrm{PK}\}) \rightarrow \mathrm{CT}$. The encryption algorithm takes in a message M, an $n \times \ell$ access matrix A with ρ mapping its rows to attributes, the global parameters, and the public keys of the relevant authorities. It chooses a random $s \in \mathbb{Z}_N$ and a random vector $v \in \mathbb{Z}_N^\ell$ with s as its first entry. We let λ_x

denote $A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $w \in \mathbb{Z}_N^\ell$ with 0 as its first entry. We let ω_x denote $A_x \cdot w$. For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_N$. The ciphertext is computed as:

$$C_0 = Me(g_1, g_1)^s,\ C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},$$
$$C_{2,x} = g_1^{r_x},\ C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{\omega_x}\ \forall x.$$

KeyGen($\mathrm{GID}, i, \mathrm{SK}, \mathrm{GP}) \rightarrow \mathrm{K}_{i,\mathrm{GID}}$. To create a key for GID for attribute i belonging to an authority, the authority computes:

$$\mathrm{K}_{i,\mathrm{GID}} = g_1^{\alpha_i} H(\mathrm{GID})^{y_i}.$$

Decrypt($\mathrm{CT}, \{\mathrm{K}_{i,\mathrm{GID}}\}, \mathrm{GP}) \rightarrow M$. We assume the ciphertext is encrypted under an access matrix (A, ρ). To decrypt, the decryptor first obtains $H(\mathrm{GID})$ from the random oracle. If the decryptor has the secret keys $\{\mathrm{K}_{\rho(x),\mathrm{GID}}\}$ for a subset of rows A_x of A such that $(1, 0, \ldots, 0)$ is in the span of these rows, then the decryptor proceeds as follows. For each such x, the decryptor computes:

$$C_{1,x} \cdot e(H(\mathrm{GID}), C_{3,x}) / e(\mathrm{K}_{\rho(x),\mathrm{GID}}, C_{2,x}) = e(g_1, g_1)^{\lambda_x} e(H(\mathrm{GID}), g_1)^{\omega_x}.$$

The decryptor then chooses constants $c_x \in \mathbb{Z}_N$ such that

$\sum_x c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$\prod_x \left(e(g_1, g_1)^{\lambda_x} e(H(\mathrm{GID}), g_1)^{\omega_x}\right)^{c_x} = e(g_1, g_1)^s.$$

(We recall that $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot w$, where $v \cdot (1, 0, \ldots, 0) = s$ and $w \cdot (1, 0, \ldots, 0) = 0$.) The message can then be obtained as:

$$M = C_0 / e(g_1, g_1)^s.$$

4.2 Security

We apply a form of the dual system encryption technique to prove security; overcoming the new challenges that arise in the multi-authority setting. In a dual system, keys and ciphertexts can either be normal or semi-functional: normal keys can decrypt semi-functional ciphertexts, semi-functional keys can decrypt normal ciphertexts, but semi-functional keys cannot decrypt semi-functional ciphertexts. The proof proceeds by a hybrid argument over a sequence of games, where we first change the challenge ciphertext to be semi-functional, and then change the keys to be semi-functional one by one. To prove that these games are indistinguishable, we must ensure that the simulator cannot test the form of the key being turned from normal to semi-functional for itself by test decrypting a semi-functional ciphertext. We avoid this problem employing the approach of [39,37], where the simulator can only make a challenge ciphertext and key pair which is *nominally semi-functional*, meaning that both the key and ciphertext have semi-functional components, but these cancel out upon decryption. Thus, if the simulator attempts to test the form of the key for itself, decryption will succeed unconditionally.

New Challenges. The existence of multiple authorities who do not coordinate with each other introduces additional technical challenges in our case. Nominal semi-functionality must be hidden from the attacker's view, which is accomplished in [37] by using temporary "blinding factors" in the semi-functional space that are active for one key at a time. Leaving these blinding factors off for the other keys prevents leakage of information that would information-theoretically reveal nominal semi-functionality in the attacker's view. However, what allows these blinding factors to be turned on and off is the stable presence of a semi-functional term attached to a single element in each key derived from the master secret key. In the multi-authority case, we do not have this sort of structural linchpin to rely on. We still need the blinding factors to hide nominal semi-functionality, but we cannot simply excise them from the other semi-functional keys to prevent their leakage. To overcome this, we use two subgroups for the semi-functional space, and instead of removing the blinding factors from the other keys, we "switch" them from one semi-functional subgroup to the other. This switch preserves semi-functionality of the keys while avoiding leakage of information about the subgroup the semi-functional components have been switched out of.

Hybrid Organization. We now formally define our sequence of games. We will assume a one-use restriction on attributes throughout the proof: this means that the row labeling ρ of the challenge ciphertext access matrix (A, ρ) must be injective.

The first game, Game_{Real}, is the real security game. We next define $\text{Game}_{Real'}$, which is like the real security game, except that the random oracle maps identities GID to random elements of G_{p_1} instead of G. We now define semi-functional ciphertexts and keys, which are used only in the proof - not in the real system.

Semi-functional ciphertexts will contain terms from subgroups G_{p_2} and G_{p_3}. Semi-functional keys will be of two types: semi-functional keys of Type 1 will have terms in G_{p_2}, while semi-functional keys of Type 2 will have terms in G_{p_3}. When a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the extra terms from G_{p_2} in the key will be paired with the extra G_{p_2} terms in the ciphertext, which will cause decryption to fail. When a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the extra terms from G_{p_3} in the key will be paired with the extra G_{p_3} terms in the ciphertext, which will cause decryption to fail.

To more precisely describe semi-functional ciphertexts and keys, we first fix random values $z_i, t_i \in \mathbb{Z}_N$ for each attribute i which will be common to semi-functional ciphertexts and keys. These values are fixed per attribute, and do not vary for different users.

Semi-functional Ciphertexts. To create a semi-functional ciphertext, we first run the encryption algorithm to obtain a normal ciphertext,

$$C'_0,\ C'_{1,x},\ C'_{2,x},\ C'_{3,x}\ \forall x.$$

We let g_2, g_3 denote generators of G_{p_2} and G_{p_3} respectively. We choose two random vectors $u_2, u_3 \in \mathbb{Z}_N^\ell$ and set $\delta_x = A_x \cdot u_2$, $\sigma_x = A_x \cdot u_3$ for each row A_x of the access matrix A. We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row $A_x \in \overline{B}$, we also choose random exponents $\gamma_x, \psi_x \in \mathbb{Z}_N$. The semi-functional ciphertext is formed as:

$$C_0 = C'_0,\ C_{1,x} = C'_{1,x},\ C_{2,x} = C'_{2,x} g_2^{\gamma_x} g_3^{\psi_x},$$

$$C_{3,x} = C'_{3,x} g_2^{\delta_x + \gamma_x z_{\rho(x)}} g_3^{\sigma_x + \psi_x t_{\rho(x)}}\ \forall x \text{ s.t. } A_x \in \overline{B},$$

$$C_{1,x} = C'_{1,x},\ C_{2,x} = C'_{2,x},\ C_{3,x} = C'_{3,x} g_2^{\delta_x} g_3^{\sigma_x}\ \forall x \text{ s.t. } A_x \in B.$$

We say a ciphertext is *nominally* semi-functional when the values δ_x are shares of 0.

Semi-functional Keys. We define the *key for identity* GID to be the collection of $H(\text{GID})$ and all keys $\text{K}_{i,\text{GID}}$ for attributes i belonging to good authorities requested by the attacker throughout the game. (These queries may occur at different times.) Semi-functional keys for an identity GID will be of two types: Type 1 or Type 2. To create a semi-functional key for identity GID, we let $H'(\text{GID})$ be a random element of G_{p_1}, and we choose a random exponent $c \in \mathbb{Z}_N$.

To create a semi-functional key of Type 1, we define the random oracle's output on GID to be:

$$H(\text{GID}) = H'(\text{GID}) g_2^c.$$

We create $\text{K}_{i,\text{GID}}$ (for an attribute i controlled by a good authority) by first creating a normal key $\text{K}'_{i,\text{GID}}$ and setting:

$$\text{K}_{i,\text{GID}} = \text{K}'_{i,\text{GID}} g_2^{c z_i}.$$

To create a semi-functional key of Type 2, we define the random oracle's output on GID to be:

$$H(\text{GID}) = H'(\text{GID}) g_3^c.$$

We create $\text{K}_{i,\text{GID}}$ (for an attribute i controlled by a good authority) by first creating a normal key $\text{K}'_{i,\text{GID}}$ and setting:

$$\text{K}_{i,\text{GID}} = \text{K}'_{i,\text{GID}} g_3^{c t_i}.$$

We note that when a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the additional terms $e(g_2, g_2)^{c\delta_x}$ prevent decryption from succeeding, except when the values δ_x are shares of 0 (i.e. when we have a nominally semi-functional ciphertext). When a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the additional terms $e(g_3, g_3)^{c\sigma_x}$ prevent successful decryption.

We now define Game_0, which is like $\text{Game}_{Real'}$, except that the ciphertext given to the attacker is semi-functional. We let q be the number of identities GID for which the attacker makes key queries $\text{K}_{i,\text{GID}}$. We define $\text{Game}_{j,1}$ and $\text{Game}_{j,2}$ for each j from 1 to q as follows:

Game$_{j,1}$. This is like Game$_0$, except that for the first $j-1$ queried identities, the received keys are semi-functional of Type 2, and the received key for the j^{th} queried identity is semi-functional of Type 1. The remaining keys are normal.

Game$_{j,2}$. This is like Game$_0$, except that for the first j queried identities, the received keys are semi-functional of Type 2. The remaining keys are normal. We note that in Game$_{q,2}$, all keys are semi-functional of Type 2.

Game$_{Final}$. In this game, all keys are semi-functional of Type 2, and the ciphertext is a semi-functional encryption of a random message. We note that the attacker has advantage 0 in this game.

We show these games are indistinguishable in the following lemmas. We give the most interesting proof below, and the remaining proofs can be found in the full version of this paper.

Lemma 1. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{Real}Adv_{\mathcal{A}} - Game_{Real'}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Lemma 2. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{Real'}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 1.*

Lemma 3. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j-1,2}Adv_{\mathcal{A}} - Game_{j,1}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ receives g_1, g_3, X_1X_2, T. $\mathcal{B}$ will simulate either Game$_{j-1,2}$ or Game$_{j,1}$ with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters $e(g_1, g_1)^{\alpha_i}, g_1^{y_i}$.

We let GID_k denote the k^{th} identity queried by $\mathcal{A}$. When $\mathcal{A}$ first queries the random oracle for $H(\mathrm{GID}_k)$, if $k > j$, then $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_k} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_k) = g_1^{h_{\mathrm{GID}_k}}$. If $k < j$, then $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_k} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_k) = (g_1g_3)^{h_{\mathrm{GID}_k}}$ (we note that this is a random element of $G_{p_1p_3}$ since the values of h_{GID_k} modulo p_1 and modulo p_3 are uncorrelated). When $k = j$, $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_j} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_j) = T^{h_{\mathrm{GID}_j}}$. In all cases, it stores this value so that it can respond consistently if $H(\mathrm{GID}_k)$ is queried again.

When $\mathcal{A}$ makes a key query (i, GID_k), $\mathcal{B}$ responds as follows. If $H(\mathrm{GID}_k)$ has already been fixed, then $\mathcal{B}$ retrieves the stored value. Otherwise, $\mathcal{B}$ creates $H(\mathrm{GID}_k)$ according to k as above. $\mathcal{B}$ forms the key as:

$$\mathrm{K}_{i,\mathrm{GID}_k} = g_1^{\alpha_i} H(\mathrm{GID}_k)^{y_i}.$$

Notice that for $k < j$, $\mathcal{B}$ forms properly distributed semi-functional keys of Type 2, where t_i is congruent to y_i modulo p_3 (these are uncorrelated from the values of y_i modulo p_1 which appear in the public parameters). Also recall that the values t_i are fixed per attribute, and do not vary across different keys. For $k > j$, $\mathcal{B}$ forms properly distributed normal keys. For $k = j$, $\mathcal{B}$ forms a normal key if $T \in G_{p_1}$ and a semi-functional key of Type 1 if $T \in G_{p_1 p_2}$.

At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0, M_1, and an access matrix (A, ρ). $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β as follows. (We note that $\mathcal{B}$ will produce a nominally semi-functional ciphertext, but this will be hidden from $\mathcal{A}$'s view.) First, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_N$ and sets $C_0 = Me(g_1, g_1)^s$. $\mathcal{B}$ also chooses three vectors, $v = (s, v_2, \ldots, v_\ell), w = (0, w_2, \ldots, w_\ell), u = (u_1, \ldots, u_\ell)$, where $v_2, \ldots, v_\ell, w_2, \ldots, w_\ell, u_1, \ldots, u_\ell$ are chosen randomly from $\mathbb{Z}_N$. We let $\lambda_x = A_x \cdot v$, $\omega_x = A_x \cdot w$, and $\sigma_x = A_x \cdot u$.

$\mathcal{A}$ additionally supplies $\mathcal{B}$ with public parameters $g^{y_i}, e(g_1, g_1)^{\alpha_i}$ for attributes i belonging to corrupt authorities which are included in the access matrix (A, ρ). We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row A_x in B, $\mathcal{B}$ chooses a random value $r_x \in \mathbb{Z}_N$. For each row $A_x \in \overline{B}$, $\mathcal{B}$ chooses random values $\psi_x, r'_x \in \mathbb{Z}_N$, and will implicitly set $r_x = r r'_x$, where g_1^r is X_1.

For each row $A_x \in B$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} \left(e(g_1, g_1)^{\alpha_{\rho(x)}}\right)^{r_x},$$

$$C_{2,x} = g_1^{r_x},\ C_{3,x} = \left(g_1^{y_{\rho(x)}}\right)^{r_x} (X_1 X_2)^{\omega_x} g_3^{\sigma_x}.$$

For each row $A_x \in \overline{B}$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, X_1 X_2)^{\alpha_{\rho(x)} r'_x},$$

$$C_{2,x} = (X_1 X_2)^{r'_x} g_3^{\psi_x},\ C_{3,x} = (X_1 X_2)^{y_{\rho(x)} r'_x} g_3^{y_{\rho(x)} \psi_x} (X_1 X_2)^{\omega_x} g_3^{\sigma_x}.$$

We note that the $X_1^{\omega_x}$ is $g_1^{A_x \cdot rw}$, and rw is a random vector with first coordinate equal to 0. This is a semi-functional ciphertext with parameters $\delta_x = A_x \cdot cw$ modulo p_2 where g_2^c is X_2, $g_2^{\gamma_x}$ equals $X_2^{r'_x}$, $z_{\rho(x)} = y_{\rho(x)}$ modulo p_2, and $t_{\rho(x)} = y_{\rho(x)}$ modulo p_3.

To see that this is properly distributed, we note that since $r'_x, y_{\rho(x)}$ are chosen randomly in $\mathbb{Z}_N$, their values modulo p_1 and modulo p_2 are uncorrelated. This means that our $\gamma_x, \psi_x, z_{\rho(x)}, t_{\rho(x)}$ parameters are randomly distributed. It is clear that σ_x is properly distributed, since it is a share of a random vector. The entries $w_2, \ldots, w_\ell$ of w are also randomly distributed modulo p_2, however the δ_x's are shares of 0 from the simulator's perspective. We must argue that these appear to be shares of a random exponent in $\mathcal{A}$'s view.

We let the space R denote the span of the rows of A whose attributes are in B and the rows whose attributes $\rho(x)$ are queried by the attacker with identity GID_j. This space cannot include the vector $(1, 0, \ldots, 0)$, so there is some vector u' which is orthogonal to R modulo p_2 and not orthogonal to $(1, 0, \ldots, 0)$. We

can then write $cw = w' + au'$ for some a modulo p_2 and w' in the span of the other basis vectors. We note that w' is uniformly distributed in this space, and reveals no information about a. The value of the first coordinate of cw modulo p_2 depends on the value of a, but the shares δ_x for $A_x \in B$ contain no information about a. The only information $\mathcal{A}$ receives about the value of a appears in exponents of the form $\delta_x + \gamma_x z_{\rho(x)}$, where the $z_{\rho(x)}$ is a new random value each time that appears nowhere else (recall that ρ is constrained to be injective). (We note that these $z_{\rho(x)}$ values modulo p_2 do not occur in any keys for identities not equal to GID_j, since these keys are either normal or semi-functional of type 2, and hence do not have components in G_{p_2}.) As long as γ_x does not equal 0 ($\gamma_x = 0$ with only negligible probability), this means that any value of δ_x can be explained by $z_{\rho(x)}$ taking on a particular value. Since $z_{\rho(x)}$ is uniformly random, this means that no information about the value of a modulo p_2 is revealed. Hence, the value being shared is information-theoretically hidden, and the δ_x's are properly distributed in the adversary's view.

Though it is hidden from $\mathcal{A}$, the fact that we can only make δ_x shares of 0 is crucial here (i.e. the simulator can only make a nominally semi-functional ciphertext). If $\mathcal{B}$ tried to test the semi-functionality of the j^{th} key for itself by making a challenge ciphertext the key could decrypt, decryption would succeed regardless of the presence of G_{p_2} components, since the δ_x's are shares of 0. Hence the simulator would not be able to tell whether the j^{th} key was semi-functional of Type 1 or normal.

In summary, when $T \in G_{p_1}$, $\mathcal{B}$ properly simulates $\mathrm{Game}_{j-1,2}$. When $T \in G_{p_1 p_2}$, $\mathcal{B}$ properly simulates $\mathrm{Game}_{j,1}$ with probability negligibly close to 1. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to obtain advantage negligibly close to ϵ in breaking Assumption 2.

Lemma 4. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j,1}Adv_{\mathcal{A}} - Game_{j,2}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Lemma 5. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{q,2}Adv_{\mathcal{A}} - Game_{Final}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 4.*

5 Discussion

There are multiple ways in which one might extend our work.

Removing the Random Oracle. It would be desirable to remove the need for a random oracle and replace it with a concrete function H mapping identities to group elements. One approach would be to fix a degree d polynomial, $P(x)$, and map identities in $\mathbb{Z}_N$ to elements of G by setting $H(\mathrm{GID}) := g^{P(\mathrm{GID})}$, where g denotes a generator of the group G. This approach has previously been employed to obtain large universe constructions for Attributed-Based encryption [30]. The public parameters would then include $\{g^{P(x_i)}\}$ for $d+1$ points x_i so that $H(\mathrm{GID})$ could be computed for any GID by polynomial interpolation. We note that $P(x)$

is a $(d+1)$-wise independent function modulo primes, but this will leave the system vulnerable to collusion attacks when $\geq d+1$ users collude. Clearly, this is far from ideal, and we would prefer a better method with stronger security guarantees.

Prime order groups. An interesting direction is create a prime order group variant of our system. Using groups of prime order can potentially lead to more efficient systems (via faster group operations) and security under different assumptions. One approach is to simply use our exact construction except use a group order of one prime (instead of a product of three primes). Applying this setting results in an efficient system that we show to be generically secure in the full version of this paper. However, this construction does not lend itself (to the best of our knowledge) to a proof under a non-interactive assumption.

Another possible approach is to realize the subspaces needed for dual system encryption proofs using vector spaces over prime order groups instead of subgroups. We note that several systems such as BGN encryption [15], Groth-Ostrovsky-Sahai NIZK proofs [32], traitor tracing [16] , and predicate encryption [17,36] were originally developed in the composite order setting, but later variants were developed in prime order groups [31,48,33,35,25,26,37][4]. Ideally, a variant would result in security under a simple assumption such as the decision linear assumption.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. Journal of Cryptology 21, 350–391 (2008)
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. Al-Riyami, S., Malone-Lee, J., Smart, N.: Escrow-free encryption supporting cryptographic workflow. Int. J. Inf. Sec. 5, 217–229 (2006)
4. Bagga, W., Molva, R., Crosta, S.: Policy-based encryption schemes from bilinear pairings. In: ASIACCS, pp. 368 (2006)
5. Barbosa, M., Farshim, P.: Secure cryptographic workflow in the standard model. In: Barua, R., Lange, T. (eds.) INDOCRYPT 2006. LNCS, vol. 4329, pp. 379–393. Springer, Heidelberg (2006)
6. Beimel, A.: PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
7. Bellare, M., Waters, B., Yilek, S.: Identity-based encryption secure against selective opening attack. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 235–252. Springer, Heidelberg (2011)
8. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)

[4] Freeman [25] discusses a class of general transformations, although it does not encompass our construction.

9. 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)
10. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
11. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
12. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
13. 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)
14. Boneh, D., Gentry, C., Hamburg, M.: Space-efficient identity based encryption without pairings. In: Proceedings of FOCS, pp. 647–657 (2007)
15. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
16. 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)
17. 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)
18. Bradshaw, R., Holt, J., Seamons, K.: Concealing complex policies with hidden credentials. In: ACM Conference on Computer and Communications Security, pp. 146–157 (2004)
19. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 93. Springer, Heidelberg (2001)
20. 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)
21. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
22. Chase, M., Chow, S.: Improving privacy and security in multi-authority attribute-based encryption. In: ACM Conference on Computer and Communications Security, pp. 121–130 (2009)
23. Cheung, L., Newport, C.: Provably secure ciphertext policy abe. In: ACM Conference on Computer and Communications Security, pp. 456–465 (2007)
24. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 26–28. Springer, Heidelberg (2001)
25. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
26. Garg, S., Kumarasubramanian, A., Sahai, A., Waters, B.: Building efficient fully collusion-resilient traitor tracing and revocation schemes. In: ACM Conference on Computer and Communications Security, pp. 121–130 (2010)

27. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
28. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
29. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
30. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute Based Encryption for Fine-Grained Access Conrol of Encrypted Data. In: ACM Conference on Computer and Communications Security, pp. 89–98 (2006)
31. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for nizk. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
32. 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)
33. 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)
34. 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)
35. Iovino, V., Persiano, G.: Hidden-vector encryption with groups of prime order. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 75–88. Springer, Heidelberg (2008)
36. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
37. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
38. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. Cryptology ePrint Archive, Report 2010/351 (2010), http://eprint.iacr.org/
39. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
40. Lin, H., Cao, Z., Liang, X., Shao, J.: Secure threshold multi authority attribute based encryption without a central authority. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 426–436. Springer, Heidelberg (2008)
41. Miklau, G., Suciu, D.: Controlling access to published data using cryptography. In: VLDB 2003, pp. 898–909 (2003)
42. Müller, S., Katzenbeisser, S., Eckert, C.: Distributed attribute-based encryption. In: Lee, P.J., Cheon, J.H. (eds.) ICISC 2008. LNCS, vol. 5461, pp. 20–36. Springer, Heidelberg (2009)
43. Müller, S., Katzenbeisser, S., Eckert, C.: On multi-authority ciphertext-policy attribute-based encryption. Bulletin of the Korean Mathematical Society 46(4), 803–819 (2009)

44. Ostrovksy, R., Sahai, A., Waters, B.: Attribute Based Encryption with Non-Monotonic Access Structures. In: ACM Conference on Computer and Communications Security, pp. 195–203 (2007)
45. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
46. 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)
47. Shi, E., Bethencourt, J., Chan, H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security and Privacy (2007)
48. Shi, E., Bethencourt, J., Chan, H.T.-H., Xiaodong Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: IEEE Symposium on Security and Privacy, pp. 350–364 (2007)
49. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
50. 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)
51. Smart, N.: Access control using pairing based cryptography. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 111–121. Springer, Heidelberg (2003)
52. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
53. Waters, B.: Dual system encryption: realizing fully secure ibe and hibe under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
54. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)

Threshold and Revocation Cryptosystems via Extractable Hash Proofs

Hoeteck Wee*

Queens College, CUNY
hoeteck@cs.qc.cuny.edu

Abstract. We present a new unifying framework for constructing non-interactive threshold encryption and signature schemes, as well as broadcast encryption schemes, and in particular, derive several new cryptosystems based on hardness of factoring, including:

- a threshold signature scheme (in the random oracle model) that supports ad-hoc groups (i.e., exponential number of identities and the set-up is independent of the total number of parties) and implements the standard Rabin signature;
- a threshold encryption scheme that supports ad-hoc groups, where encryption is the same as that in the Blum-Goldwasser cryptosystem and therefore more efficient than RSA-based implementations;
- a CCA-secure threshold encryption scheme in the random oracle model;
- a broadcast encryption scheme (more precisely, a revocation cryptosystem) that supports ad-hoc groups, whose complexity is comparable to that of the Naor-Pinkas scheme; moreover, we provide a variant of the construction that is CCA-secure in the random oracle model.

Our framework rests on a new notion of *threshold extractable hash proofs*. The latter can be viewed as a generalization of the extractable hash proofs, which are a special kind of non-interactive zero-knowledge proof of knowledge.

1 Introduction

As the old saying goes, "Do not put all your eggs in one basket". Indeed, this is the basic principle underlying threshold cryptography, which distributes some cryptographic functionality amongst many users in such a way that: (1) any $t+1$ parties can collectively compute the functionality; and (2) no colluding subset of t parties can compromise the security of the functionality. The two canonical applications of threshold cryptography are in public-key encryption and signature schemes, where the functionalities in consideration correspond to decryption and signing respectively. The approach was initiated in [19, 20, 21], and there is now a large body of work on threshold signature schemes [18, 27, 40, 26, 28, 29, 8, 34, 30] and threshold encryption schemes [41, 11, 24, 34, 9, 10].

* Supported by NSF CAREER Award CNS-0953626, and the US Army Research laboratory and the UK Ministry of Defence under agreement number W911NF-06-3-0001.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 589–609, 2011.

If we are willing to settle solely for pairings-based schemes, then the main questions of practical threshold encryption and signature schemes are essentially solved. The threshold signature schemes of Boneh, Lynn, Shacham [8] and threshold encryption schemes of Boyen, Mei and Waters [10, 9] are non-interactive (each party locally computes a signature/decryption share without any interaction with the other parties), guarantees robustness against corrupted parties (given a verification key, each party can check that the signature/decryption shares are well-formed) and are well-suited for use in ad-hoc groups such as MANETs ("mobile ad-hoc networks", which arise in many wireless and military settings). The latter requirement, articulated in the recent work of Gennaro et al. [30], means that the cryptographic protocol supports an identity space of exponential size and does not have any dependency on the total number of parties.

Given that the underlying principle of threshold cryptography is to avoid any single point of failure, it would be quite ironic of course to base all of threshold cryptography on pairings and discrete-log assumptions. A natural class of alternative assumptions would be that related to factoring, where many problems remain open. Here, virtually all threshold signature schemes are based on the RSA assumption; the only exception is the factoring-based scheme of Katz and Yung [34], which does not support ad-hoc groups. We also do not know of any threshold encryption schemes based on hardness of factoring which supports ad-hoc groups. More notably, we do not know of any practical CCA-secure threshold encryption scheme based on hardness of factoring, even in the random oracle model; this was posed as an open problem in [41]. Similarly, very little is known about factoring-based revocation cryptosystems, a primitive seemingly unrelated to threshold cryptosystems. These are a special kind of broadcast encryption schemes [23] where a sender broadcasts encrypted messages created in such a way that all but a small subset of recipients (the "revoked" users) can decrypt the message.

This Work. We present a new unifying framework for constructing non-interactive threshold encryption and signature schemes, as well as revocation schemes, and in particular, derive several new cryptosystems based on hardness of factoring, including:

- a threshold signature scheme (in the random oracle model) that supports ad-hoc groups and implements the standard Rabin signature (namely, the end-result of running the protocol is a Rabin signature and anyone can verify that signature as if it were generated by a standard centralized signer);
- a threshold encryption scheme (in the standard model) that supports ad-hoc groups, where encryption is the same as that in the Blum-Goldwasser cryptosystem [5] and therefore more efficient than RSA-based implementations;
- a CCA-secure threshold encryption scheme (in the random oracle model), whose computation and communication complexity is roughly that of Shoup's threshold signature scheme [40] plus that of the Hofheinz-Kiltz CCA-secure encryption scheme [32].
- a revocation cryptosystem (in the standard model) that also supports ad-hoc groups, whose complexity is comparable to that of the Naor-Pinkas scheme [36]; moreover, we provide a variant of the construction that is CCA-secure in the random oracle model.

reference	assumption	security	ad-hoc
[18, 28]	RSA	CPA	×
[25, 24]	DCR	CPA, CCA (RO)	×
[34, 24]	QR	CPA, CCA (RO)	×
[34]	factoring	CPA	×
this work	factoring	CPA	✓
this work	factoring	CCA (RO)	×

Fig. 1. Summary of threshold PKEs from assumptions related to factoring

We refer to Figure 1 for a comparison with previous factoring-based threshold cryptosystems. We also note here that our framework also captures many of the aforementioned threshold and revocation cryptosystems based on pairings and discrete-log assumptions [8, 10, 36] (see Figure 2).

2 Overview of Our Constructions

We proceed to provide an overview of our framework and the constructions.

Threshold Extractable Hash Proofs. We introduce the notion of a *threshold extractable hash proof system*, which generalizes our recent work [42]. Informally, these hash proof systems are like the Cramer-Shoup universal hash proofs [15] in that they are a special kind of non-interactive zero-knowledge proofs [6], except we replace the soundness requirement (corresponding to smoothness) with a "proof of knowledge property" [38, 17]. Specifically, the proofs are specified by a family of functions $\mathcal{H}_{\text{HK}}(\cdot,\cdot)$ indexed by a public key HK and takes two inputs, a tag and an instance u. We will require that $\mathcal{H}_{\text{HK}}(\cdot,\cdot)$ be efficiently computable either given the coin tosses used to sample the instance u, or a secret key for the corresponding tag. In addition, the family of functions is parametrized by a "threshold" value 1^t which plays the following role:

- $(t+1)$-EXTRACTABILITY: given any $t+1$ proofs for the same instance u on $t+1$ different tags, we may efficiently "extract" a witness s for the instance (this is mostly meaningful when computing the witness given only u is hard-on-average);
- t-SIMULATABILITY: on the other hand, any t proofs reveal no "useful" information about the witness, that is, there exists a simulator that can efficiently generate proofs for t different tags for an instance u without knowing the witness. The formal requirement is stronger, namely that the simulator can generate a random HK along with the secret keys for any t different tags.

We point out here that the case $t = 1$ corresponds to the "all-but-one" extractable hash proofs in [42]; that is, threshold extractable hash proofs may be regarded as a "all-but-t" analogue of extractable hash proofs.

From Hash Proofs to Cryptosystems. With this informal overview of threshold extractable hash proofs, we can now outline how we derive threshold and revocation cryptosystems (see Figure 3 for the parameters). We note here that we are working in the model with a trusted dealer that generates HK and issues each party with identity ID with the secret key corresponding to the tag ID. This is well-suited for dynamic ad-hoc networks where any user can join the network at any time and register with the trusted dealer to obtain a secret key; however, in this work, we only address the setting where the threshold t is fixed once and for all.

- THRESHOLD ENCRYPTION: To encrypt a bit, we generate a random instance-witness pair (u, s), mask the bit using the hard-core bit of s, and publish u along with the masked bit. The decryption share from a party ID is simply a proof $\mathcal{H}_{\text{HK}}(\text{ID}, u)$. The functionality requirement follows from $(t+1)$-extractability – anyone can decrypt upon receiving the shares from any $t+1$ parties; moreover, t-simulatability prevents any t colluding parties from decrypting the message.

- THRESHOLD SIGNATURES: The signature for a message M is a witness s for the instance $\mathsf{H}(M)$ where $\mathsf{H}(\cdot)$ is a hash function modeled as a random oracle (a "full domain hash"). The signature share from a user ID is again a proof $\mathcal{H}_{\text{HK}}(\text{ID}, \mathsf{H}(M))$; functionality and security is exactly analogous to that for threshold encryption.

- REVOCATION CRYPTOSYSTEM: Here, we want to encrypt messages in such a way that any party outside a revoked set of t users $\text{ID}_1, \ldots, \text{ID}_t$ can decrypt. (To revoke fewer than t users, we may simply add "dummy" identities.) Again, to encrypt a bit, we generate a random instance-witness pair (u, s), mask the bit using the hard-core bit of s, and publish u along with the masked bit and t proofs $\mathcal{H}_{\text{HK}}(\text{ID}_1, u), \ldots, \mathcal{H}_{\text{HK}}(\text{ID}_t, u)$. Any party ID outside the revoked set can compute a $t+1$'th proof using the secret key for the tag ID and then derive s to decrypt.

We also show how to realize robustness for signatures following [28], and CCA security for threshold encryption and revocation schemes (with instantiations in the random oracle model). Our techniques for achieving CCA security follow the "all-but-one extractable hash proof" paradigm in [42, 12, 32], whereas most of the previous constructions [11, 22, 24] rely on the Cramer-Shoup [14, 15] or the Naor-Yung paradigm [37] which seem to be inherently limited to decisional assumptions.

Realizing Threshold Extractable Hash Proofs. We begin with an informal description of our approach for constructing threshold extractable hash proofs. The approach generalizes the direct constructions of all-but-one extractable hash proofs in [42] and provide a different perspective into those constructions. The basic idea is to exploit Shamir secret sharing in the exponent. More precisely, we sample a random degree t polynomial f subject to the constraint $f(0)$ is a special trapdoor such that $s = u^{f(0)}$. The master secret key is given by f and the public key HK is some "commitment" to the coefficients of f. The secret key corresponding to TAG is given by $f(\text{TAG})$ and $\mathcal{H}_{\text{HK}}(\text{TAG}, u) := u^{f(\text{TAG})}$. Computing $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ given the coin tosses for generating u corresponds to evaluating f in the exponent. Given the hash proofs for u corresponding to $t+1$ distinct tags, we may recover $u^{f(0)}$ via Lagrangian interpolation

primitive	CDH/DDH	BDDH	factoring
threshold PKE	folklore	folkore	new
threshold signatures	[27] (RO)	[8]	new (RO)
broadcast PKE	[36]	[36]	new
CCA threshold PKE	[41, 24] (RO)	[10, 9]	new (RO)
CCA broadcast PKE	[22]	new	new (RO)

Fig. 2. Summary of previous and present constructions from CDH/DDH, BDDH and hardness of factoring. For most primitives and assumptions, our constructions match and improve upon existing constructions. We underline the references for the two settings where existing work achieve better parameters than our work. We note that the [22] scheme only achieves a relaxed notion of CCA security.

primitive	public key	secret key	ciphertext/signature	decryption/signing share
threshold PKE	$O(1)$	$O(1)$	$O(1)$	$O(1)$
threshold signatures	$O(1)$	$O(1)$	$O(1)$	$O(1)$
broadcast PKE	$O(t)$	$O(1)$	$O(t)$	$O(1)$
CCA threshold PKE	$O(1)$	$O(1)$	$O(1)$	$O(1)$
CCA broadcast PKE	$O(t)$	$O(1)$	$O(t)$	$O(1)$

Fig. 3. Complexity of our BDDH and factoring-based schemes, as measured by number of group elements. For broadcast PKE, t denotes the revocation threshold. Here, secret key refers to the user/server's secret key; the dealer's secret key has size $O(t)$.

in the exponent. This is easy for discrete-log type settings since we know the order of the group. Generating simulated proofs or secret keys for any t distinct tags is easy since the evaluation of f at any t locations look random; the main technical complication comes in having to simulate a consistent HK (though that is again easy for discrete-log type settings).

The factoring-based construction is based on Rabin's trapdoor permutation. We begin with the simple observation that we may compute a square root of u by exponentiation to the secret value $(\phi(N)+1)/2 = 2^{-1} \pmod{\phi(N)}$. Here, we use secret sharing over the ring $\mathbb{Z}_{\phi(N)}$, whereas the previous factoring-based scheme in [34] applies secret sharing to the factorization of the modulus N. In order to perform Lagrangian interpolation over the ring $\mathbb{Z}_{\phi(N)}$ of unknown order, we build on ideas developed in the context of RSA-based schemes [40, 30] and the factoring-based cryptosystem in [32]. Informally, we set $f(0)$ to be $2^{-(t+1)\lceil \log N \rceil} \pmod{\phi(N)}$. Given the hash proofs $u^{f(\text{TAG})}$ corresponding to $t+1$ distinct tags, we may recover $u^{Df(0)}$, where D denotes an integer used to "clear the denominator" in the fractional Lagrangian coefficients and it depends on the tags used in the $t+1$ proofs. In order to support an identity space of exponential size, we bound the highest power of 2 that divides D by $2^{-t\lceil \log N \rceil}$, following [30]. Given both u and $u^{Df(0)}$, we may then recover $s = u^{1/2}$ using Shamir's algorithm for "GCD in the exponent" [39].

In our constructions, we "commit" to the coefficients of f instead of the evaluations of f in HK, evaluating $u^{f(\text{TAG})}$ given the coin tosses used to sample u does not require interpolation; this appears to be the first time this property is exploited for RSA/factoring-based schemes and is important for handling ad-hoc networks in our revocation scheme.

Towards Lattice-Based Instantiations. Looking forward, we plan to look into lattice-based instantiations of threshold extractable hash proofs, extending the ideas and results in [2, 3]. One possible starting point is the following construction: $\text{HK} := (\mathbf{A}_0, \ldots, \mathbf{A}_t) \leftarrow_{\text{R}} \mathbb{Z}_q^{m \times n}$ where $m = \text{poly}(n)$ and $f(\text{TAG}) := \mathbf{A}_0 + \mathbf{A}_1 \text{TAG} + \cdots + \mathbf{A}_t \text{TAG}^t$ for $\text{TAG} \in \mathbb{Z}_n \subset \mathbb{Z}_q$. The instance u is a perturbed lattice point $\mathbf{A} \cdot \mathbf{s} + \eta$ where $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ is of the form $f(\text{TAG})\mathbf{s} + \eta$. Due to the interaction between the Lagrangian coefficients and the noise vectors, we will need to work with field sizes and approximation factors much larger than n^t (c.f. [3]). However, under sub-exponential hardness assumptions for lattice problems, we could still potentially get meaningful results for parameters such as $t = \sqrt{n}$, say.

3 Preliminaries and Definitions

A *key encapsulation mechanism* (KEM) $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with key-space $\{0,1\}^k$ consists three polynomial-time algorithms.Via $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^k)$ the randomized key-generation algorithm produces public/secret keys for security parameter 1^k; via $(C, K) \leftarrow \mathsf{Enc}(\text{PK})$, the randomized encapsulation algorithm creates a uniformly distributed symmetric key $K \in \{0,1\}^k$, together with a ciphertext C; via $K \leftarrow \mathsf{Dec}(\text{SK}, C)$, the possessor of secret key SK decrypts ciphertext C to get back a key K which is an element in $\{0,1\}^k$ or a special reject symbol $\perp$. For consistency, we require that for all k and all $(C, K) \leftarrow \mathsf{Enc}(\text{PK})$, we have $\Pr[\mathsf{Dec}(\text{SK}, C) = K] = 1$.

3.1 Binary Relations for Search Problems

Fix a family of (binary) relations R_{PP} indexed by a public parameter PP. We require that PP be efficiently samplable given a security parameter 1^k, and assume that all algorithms are given PP as part of its input. We omit PP henceforth whenever the context is clear. We will also require that R_{PP} be efficiently samplable, where the sampling algorithm is denoted by SampR. Intuitively, the relation R_{PP} corresponds to a hard search problem, that is, given a random u, it is hard to find s such $(u, s) \in \mathsf{R}_{\text{PP}}$. More formally, we say that a binary relation R_{PP} is *one-way* if:

- with overwhelming probability over PP, for all u, there exists at most one s such that $(u, s) \in \mathsf{R}_{\text{PP}}$; and
- there is an efficiently computable generator G such that $\mathsf{G}_{\text{PP}}(s)$ is pseudorandom even against an adversary that gets PP, u, where $(u, s) \leftarrow_{\text{R}} \mathsf{SampR}(\text{PP})$. (We will also refer to G as extracting hard-core bits from s.) That is, the following quantity is negligible for all PPT $\mathcal{A}$:

$$\mathsf{AdvPRG}^{\mathcal{A}}(k) := \Pr\left[b = b' : \begin{array}{c}(u,s) \leftarrow \mathsf{SampR}(\mathrm{PP});\\ K_0 \leftarrow \mathsf{G}(s); K_1 \leftarrow_{\mathrm{R}} \{0,1\}^k; b \leftarrow_{\mathrm{R}} \{0,1\};\\ b' \leftarrow \mathcal{A}(\mathrm{PP}, u, K_b)\end{array}\right]$$

For relations where computing s given u is hard on average, we may derive a generator G_{PP} with a one-bit output via the Goldreich-Levin hard-core bit $\mathsf{GL}(\cdot)$ [31] (with the randomness in PP). In many cases as we shall see shortly, we may derive a linear number of hard-core bits by either iterating a one-way permutation or relying on decisional assumptions.

3.2 Threshold Extractable Hash Proofs

We consider a family of hash functions $\mathcal{H}_{\mathrm{HK}}(\cdot,\cdot)$ indexed by a public key HK, that takes as input a tag and an instance. More formally, an *threshold extractable hash proof system* is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Pub}, \mathsf{Ext}, \mathsf{Priv})$ satisfying the following properties with overwhelming probability over $(\mathrm{PP}, \mathrm{SP})$:

(KEY GENERATION.) The set-up algorithm $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ generates public keys HK and a master secret key MSK. Given a tag TAG, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG}) = \mathrm{SK}_{\mathrm{TAG}}$.

(PUBLIC EVALUATION.) For all HK, TAG and $(u,s) = \mathsf{SampR}(r)$: $\mathsf{Pub}(\mathrm{HK}, \mathrm{TAG}, r) = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$.

(PRIVATE EVALUATION.) For all HK, TAG and u: $\mathsf{Priv}(\mathrm{SK}_{\mathrm{TAG}}, u) = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$

($(t+1)$-EXTRACTION.) For all HK, u and all $t+1$ distinct tags $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$:

$$(u, \mathsf{Ext}(u, \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}_1, u), \ldots, \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}_{t+1}, u))) \in \mathsf{R}_{\mathrm{PP}}$$

We note here that Ext also receives as input the tags $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$ (omitted for notational simplicity) but does not require as input HK.

(t-SIMULATION.) For all $(\mathrm{PP}, \mathrm{SP}), 1^t$ and all $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_t$, the distribution of $(\mathrm{HK}, \mathrm{SK}_{\mathrm{TAG}_1}, \ldots, \mathrm{SK}_{\mathrm{TAG}_t})$ in the following experiments are statistically indistinguishable:

- the first is that obtained via key generation: $(\mathrm{HK}, \mathrm{MSK}) \leftarrow_{\mathrm{R}} \mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ and $\mathrm{SK}_{\mathrm{TAG}_i} := \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG}_i)$ for $i = 1, \ldots, t$;
- the second is that given by $\mathsf{SetupSim}(\mathrm{PP}, \mathrm{TAG}_1, \ldots, \mathrm{TAG}_t)$

Finally, we say that a threshold extractable hash proof system is *publicly verifiable* if there is an efficient algorithm Ver that on input $(\mathrm{HK}, \mathrm{TAG}, u, \tau)$ outputs true iff $\tau = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$.

4 Threshold Encryption Schemes

We define a threshold KEM (from which we can readily build a threshold encryption scheme):

(SHARING PHASE.) The set-up algorithm $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ generates a public key PK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}) = \mathrm{SK}_{\mathrm{ID}}$.

(ENCRYPTION.) The encapsulation algorithm $\mathsf{Enc}(\mathrm{PK})$ generates (C, K), namely a random key K along with a ciphertext C.

(DECRYPTION.) The share decryption algorithm $\mathsf{ShareDec}(\mathrm{ID}, C)$ takes an identity ID and computes the decryption share for that identity using its secret key $\mathrm{SK}_{\mathrm{ID}}$. Moreover, there's a combining algorithm that takes any $t+1$ decryption shares and outputs K.

Semantic Security. We define the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[b = b' : \begin{array}{l} (\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ \mathrm{SK}_{\mathrm{ID}_i^*} \leftarrow \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}_i^*), i = 1, \ldots, t; \\ (C, K_0) \leftarrow \mathsf{Enc}(\mathrm{PK}); K_1 \leftarrow_{\mathrm{R}} \{0,1\}^k; b \leftarrow_{\mathrm{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}_2^{\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\mathrm{PK}))}(\mathrm{PK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*}, K_b, C) \end{array}\right]$$

Here, $\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\mathrm{PK}))$ denotes an oracle that given an input ID, computes a fresh ciphertext C using $\mathsf{Enc}(\mathrm{PK})$ and returns $\mathsf{ShareDec}(\mathrm{ID}, C)$ along with C. This captures the fact that the adversary may obtain decryption shares of fresh encryptions of known messages, and captures the security notion used in [16] with applications to secure voting. In the CCA setting, we provide $\mathcal{A}_2$ with oracle access to $\mathsf{ShareDec}(\cdot, \cdot)$, with the restriction that $\mathcal{A}_2$ is only allowed to query $\mathsf{ShareDec}(\cdot, \cdot)$ on ciphertexts different from the challenge ciphertext C. A threshold encryption scheme is said to be *indistinguishable against chosen plaintext attacks* (IND-CPA) if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvThEnc}^{\mathcal{A}}(k)$ is a negligible function in k.

Decryption Consistency. We consider a notion of decryption consistency that for all ciphertexts C, there exists a unique value K such that for all (possibly malformed) $t+1$ decryption shares, the combining algorithm returns a value in $\{K, \perp\}$.

Theorem 1. *If R_{PP} is a one-way relation admitting a threshold extractable hash proof, then the threshold KEM shown in Figure 4 is* IND-CPA *secure.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the threshold encryption scheme, we construct a $\mathcal{B}$ that breaks the pseudorandomness of G as follows: on input (PP, u, K):

Threshold PKE

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t) \rightarrow (\text{HK}, \text{MSK})$ and sets PK to be PP. A user with identity ID is given the share $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(ENCRYPTION.) $\mathsf{Enc}(\text{PK})$: sample $(u, s) := \mathsf{SampR}(r)$, and return $(C, K) := (u, \mathsf{G}(s))$.

(DECRYPTION.) On input a ciphertext u, a user ID computes the decryption share $\sigma_{\text{ID}} := \mathsf{Priv}(\text{SK}_{\text{ID}}, u)$. Given $t+1$ decryption shares $\sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}}$, the combining algorithm computes $s := \mathsf{Ext}(u, \sigma_{\text{ID}_1}, \ldots, \sigma_{\text{ID}_{t+1}})$ and returns $\mathsf{G}(s)$.

Fig. 4. Threshold encryption scheme from threshold hash proofs

- Run $(\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$
- Run $\mathsf{SetupSim}(\text{PP}, \text{ID}_1^*, \ldots, \text{ID}_t^*)$ to get $(\text{HK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*})$
- Output $\mathcal{A}_2((\text{PP}, \text{PK}), \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*}, K, u)$, simulating $\mathsf{ShareDec}(\cdot, \mathsf{Enc}(\text{PK}))$ using Pub.

It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvThEnc}^{\mathcal{A}}(k)$. □

5 Threshold Signature Schemes

A threshold signature scheme proceeds in two phases:

(SHARING PHASE.) The set-up algorithm $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t)$ generates a verification VK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key SK_{ID}.

(SIGNATURE COMPUTATION PHASE.) The signature computation $\mathsf{ShareSign}(\cdot, \cdot)$ algorithm takes an identity ID and a message and computes the signature share for that identity using its secret key SK_{ID}. Moreover, there's a combining algorithm that takes any $t+1$ signature shares and outputs the actual signature σ.

Unforgeability. We define the advantage $\mathsf{AdvThSign}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[\mathsf{Ver}(\text{PK}, \text{VK}, M^*, \sigma^*) = 1 : \begin{array}{l} (\text{ID}_1^*, \ldots, \text{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\text{VK}, \text{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ \text{SK}_{\text{ID}_i^*} \leftarrow \mathsf{ShareGen}(\text{MSK}, \text{ID}_i^*), i = 1, \ldots, t; \\ (M^*, \sigma^*) \leftarrow \mathcal{A}_2^{\mathsf{ShareSign}(\cdot,\cdot)}(\text{VK}, \text{SK}_{\text{ID}_1^*}, \ldots, \text{SK}_{\text{ID}_t^*}) \end{array}\right]$$

with the restriction that $\mathcal{A}_2$ never made a query to $\mathsf{ShareSign}(\cdot, \cdot)$ on the message M^*. A threshold signature scheme is said to be *existentially unforgeable* if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvThSign}^{\mathcal{A}}(k)$ is a negligible function in k.

Construction. We assume that R_{PP} is efficiently verifiable and that there is a hash function H that maps the message space into instances of R_{PP}. The signature on a message M is a witness s such that $(\mathsf{H}(M), s) \in \mathsf{R}_{\mathrm{PP}}$.

Threshold Signature Scheme

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates $(\mathrm{PP}, \mathrm{SP})$, runs $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t) \rightarrow (\mathrm{HK}, \mathrm{MSK})$ and sets VK to be PP. A user with identity ID is given the share $\mathrm{SK}_{\mathrm{ID}} := \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID})$.

(SHARING PHASE.) On input the security parameter 1^k and a threshold t, the dealer generates $(\mathrm{PP}, \mathrm{SP})$, runs $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t) \rightarrow (\mathrm{HK}, \mathrm{MSK})$ and sets VK to be PP. A user with identity ID is given the share $\mathrm{SK}_{\mathrm{ID}} := \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID})$.

(SIGNATURE COMPUTATION PHASE.) On input a message M, the user ID computes $u := \mathsf{H}(M)$ and publishes the signature fragment $\sigma_{\mathrm{ID}} := \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}}, u)$. Given $t+1$ signatures fragments $\sigma_{\mathrm{ID}_1}, \ldots, \sigma_{\mathrm{ID}_{t+1}}$, the signature is given by $\mathsf{Ext}(u, \sigma_{\mathrm{ID}_1}, \ldots, \sigma_{\mathrm{ID}_{t+1}})$.

(SIGNATURE VERIFICATION.) On input a key VK, a message M and a signature σ, the verification accepts iff $(\mathsf{H}(M), \sigma) \in \mathsf{R}_{\mathrm{VK}}$.

Fig. 5. Threshold signatures from threshold hash proofs

Theorem 2. *If R_{PP} is a one-way relation admitting a threshold extractable hash proof, then the threshold signature shown in Figure 5 is existentially unforgeable in the random oracle model. Moreover, if the hash proof is publicly verifiable, then the signature scheme is robust.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the threshold signature scheme, we construct a $\mathcal{B}$ that breaks the one-wayness of R as follows: on input (PP, u):

- Run $(\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$
- Run $\mathsf{SetupSim}(\mathrm{PP}, \mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*)$ to get $(\mathrm{HK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*})$
- Output $\mathcal{A}_2(\mathrm{VK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*})$

Suppose $\mathcal{A}_2$ requests for signatures on $M_1, \ldots, M_q$ and outputs a forgery on M^*. For each i, we sample $\mathsf{SampR}(r_i) := (u_i, s_i)$ and map $\mathsf{H}(M_i)$ to u_i for which we can compute any signature fragment using Pub. Finally, we map $\mathsf{H}(M^*)$ to u and thus a valid signature on M^* is a valid witness for u. It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvThSig}^{\mathcal{A}}(k) - q^2/2^k$ (where $q^2/2^k$ is an upper bound on the probability of a collision in the output of the random oracle). □

6 Revocation Schemes

We define a revocation KEM:

(SHARING PHASE.) The set-up algorithm $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ generates public keys HK and a master secret key MSK. Given an identity ID, the share generation algorithm computes an associated key $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}) = \mathrm{SK}_{\mathrm{ID}}$.

(ENCRYPTION.) The encapsulation algorithm Enc takes PK and a set of t revoked users $\mathrm{ID}_1, \ldots, \mathrm{ID}_t$ generates (C, K), namely a random key K along with a ciphertext C.

(DECRYPTION.) The decapsulation algorithm Dec takes $\mathrm{SK}_{\mathrm{ID}}$ for any $\mathrm{ID} \neq \mathrm{ID}_1, \ldots, \mathrm{ID}_t$ and outputs K.

Semantic Security. We define the advantage $\mathsf{AdvBrEnc}^{\mathcal{A}}(k)$ to be:

$$\Pr\left[\begin{array}{l} S = (\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k); \\ (\mathrm{PK}, \mathrm{MSK}) \leftarrow \mathsf{Gen}(1^k, 1^t); \\ b = b' : \mathrm{SK}_{\mathrm{ID}_i^*} \leftarrow \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID}_i^*), i = 1, \ldots, t; \\ (C, K_0) \leftarrow \mathsf{Enc}(\mathrm{PK}, S); K_1 \leftarrow_{\mathrm{R}} \{0,1\}^k; b \leftarrow_{\mathrm{R}} \{0,1\}; \\ b' \leftarrow \mathcal{A}_2(\mathrm{PK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*}, K_b, C) \end{array}\right]$$

In the CCA setting, we provide $\mathcal{A}_2$ with oracle access to $\mathsf{Dec}(\cdot,\cdot)$, with the restriction that $\mathcal{A}_2$ is only allowed to query $\mathsf{Dec}(\cdot,\cdot)$ on ciphertexts different from the challenge ciphertext. A revocation scheme is said to be *indistinguishable against chosen plaintext attacks* (IND-CPA) if for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{AdvBrEnc}^{\mathcal{A}}(k)$ is a negligible function in k.

Theorem 3. *If R_{PP} is a one-way relation admitting a threshold extractable hash proof, then the broadcast KEM shown in Figure 6 is* IND-CPA *secure.*

Proof. Given a PPT $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ that breaks the broadcast KEM, we construct a $\mathcal{B}$ that breaks the pseudorandomness of G as follows: on input (PP, u, K):

- Run $(\mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*) \leftarrow \mathcal{A}_1(1^k)$
- Run $\mathsf{SetupSim}(\mathrm{PP}, \mathrm{ID}_1^*, \ldots, \mathrm{ID}_t^*)$ to get $(\mathrm{HK}, \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*})$
- Set $C := (u, \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}_1^*}, u), \ldots, \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}_t^*}, u))$.
- Output $\mathcal{A}_2((\mathrm{PP}, \mathrm{PK}), \mathrm{SK}_{\mathrm{ID}_1^*}, \ldots, \mathrm{SK}_{\mathrm{ID}_t^*}, K, C)$.

It is easy to see that $\mathsf{AdvPRG}^{\mathcal{B}}(k) \approx \mathsf{AdvBrEnc}^{\mathcal{A}}(k)$. □

7 Instantiations for the Diffie-Hellman Relation

We consider a family of groups $\mathbb{G}$ of prime order q that admits a bilinear map. The secret parameter is a random $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ and the public parameter PP is given by (g, g^α) for a random $g \leftarrow_{\mathrm{R}} \mathbb{G}$ and a random $\alpha \leftarrow_{\mathrm{R}} \mathbb{Z}_q$. We consider the Diffie-Hellman relation

$$\mathsf{R}_{\mathrm{PP}}^{\mathsf{dh}} = \left\{(u, s) \in \mathbb{G} \times \mathbb{G} : s = u^\alpha\right\}$$

Note that $\mathsf{R}_{\mathsf{PP}}^{\mathsf{dh}}$ is efficiently verifiable by computing a pairing. The associated sampling algorithm SampR picks a $r \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ and outputs $(g^r, g^{\alpha r})$.

Revocation PKE

(SHARING PHASE.) On input the security parameter 1^k and a revocation threshold t, the dealer generates $(\mathrm{PP}, \mathrm{SP})$, runs $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t) \rightarrow (\mathrm{HK}, \mathrm{MSK})$ and sets the public key PK to be $(\mathrm{PP}, \mathrm{HK})$. A user with identity ID is given the key $\mathrm{SK}_{\mathrm{ID}} := \mathsf{ShareGen}(\mathrm{MSK}, \mathrm{ID})$.

(ENCRYPTION.) In order to revoke users $\mathrm{ID}_1, \ldots, \mathrm{ID}_t$, $\mathsf{Enc}(\mathrm{PK})$: sample $(u, s) := \mathsf{SampR}(r)$, compute $\tau_i := \mathsf{Pub}(\mathrm{HK}, \mathrm{ID}_i, u, r)$ for $i = 1, \ldots, t$ and return $(C, K) := ((u, \tau_1, \ldots, \tau_t), \mathsf{G}(s))$.

(DECRYPTION.) Any user ID not in the revoked set $\{\mathrm{ID}_1, \ldots, \mathrm{ID}_t\}$ may decrypt a ciphertext $C := (u, \tau_1, \ldots, \tau_t)$ as follows: compute $s := \mathsf{Ext}(u, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\mathrm{SK}_{\mathrm{ID}}, u))$ and output $\mathsf{G}(s)$.

Fig. 6. Revocation scheme from threshold hash proofs

Hard-core Bits. Next, we explain how to obtain hard-core bits for $\mathsf{R}_{\mathsf{PP}}^{\mathsf{dh}}$ under various assumptions.

- The CDH assumption [1] asserts that computing g^{ab} given (g, g^a, g^b) is hard on average; here, we may extract a single hard-core bit from s using $\mathsf{GL}(s)$.
- The Bilinear DDH (BDDH) assumption [7] asserts that $e(g,g)^{abc}$ is pseudorandom given g, g^a, g^b, g^c where g, g^a, g^b, g^c are random elements of a bilinear group. Under BDDH, we may extract a linear number of hard-core bits from s using:

$$\mathsf{G}_{\mathrm{PP}}^{\mathsf{bddh}}(s) := e(s, g^\gamma) \quad \left(\Rightarrow \mathsf{G}_{\mathrm{PP}}^{\mathsf{bddh}}(g^{\alpha r}) = e(g,g)^{\alpha\gamma r}\right)$$

  where PP is now given by (g, g^α, g^γ). In addition, we may improve efficiency by pre-computing the pairing and setting PP to be $(g, g^\alpha, e(g, g^\gamma))$ and computing $\mathsf{G}_{\mathrm{PP}}^{\mathsf{bddh}}(g^r) := e(g, g^\gamma)^r$. This construction extends naturally to the Gap Hashed DH assumption [35].

Threshold Hash Proof System. Fix the parameters $(\mathrm{PP}, \mathrm{SP}) = ((g, g^\alpha), \alpha)$; this also fixes a group $\mathbb{G}$ of prime order q. The tag space is given by $\mathbb{F}_q \setminus \{0\}$.

(KEY GENERATION.) Pick $a_1, \ldots, a_t \leftarrow_{\mathrm{R}} \mathbb{Z}_q$ and set $f(x) := \alpha + a_1 x + \cdots a_t$.

- $\mathsf{Setup}(\mathrm{PP}, \mathrm{SP}, 1^t)$ returns $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$ and $\mathrm{MSK} := f(x)$
- $\mathsf{ShareGen}(\mathrm{MSK}, \mathrm{TAG})$ returns $\mathrm{SK}_{\mathrm{TAG}} := f(\mathrm{TAG}) \in \mathbb{Z}_q$.

(PUBLIC/PRIVATE EVALUATION.) $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ is given by $u^{f(\mathrm{TAG})} = (g^{f(\mathrm{TAG})})^r$ where $u := g^r$

- $\mathsf{Pub}(\mathrm{HK}, u, r)$ returns $\left(g^{\alpha} \cdot \prod_{i=1}^{t} (g^{a_i})^{\mathrm{TAG}^i}\right)^r$.
- $\mathsf{Priv}(\mathrm{SK}_{\mathrm{TAG}}, u)$ returns $u^{\mathrm{SK}_{\mathrm{TAG}}}$.

($(t+1)$-EXTRACTION.) Given $u, \mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$, we have $(u, u^{f(0)}) \in \mathsf{R}_{\mathrm{PP}}$. In addition, we may write $f(0) = \sum_{i=1}^{t+1} L_i \cdot f(\mathrm{TAG}_i)$ where $L_i \in \mathbb{F}_q$ are the Lagrangian coefficients which may be efficiently computed given $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_{t+1}$. This means $u^{f(0)} = \prod_{i=1}^{t+1} u^{L_i \cdot f(\mathrm{TAG}_i)}$.

- $\mathsf{Ext}(u, \tau_1, \ldots, \tau_{t+1})$ returns $\prod_{i=1}^{t+1} \tau_i^{L_i}$.

(t-SIMULATION.) Pick $\gamma_1, \ldots, \gamma_t \leftarrow_{\mathrm{R}} \mathbb{Z}_q$. This uniquely determines a degree t polynomial $f(x) = \alpha + a_1 x + \cdots + a_t x^t$ such that $f(\mathrm{TAG}_i) = \gamma_i$ for $i = 1, \ldots, t$. Moreover, $a_1, \ldots, a_t$ are given by the solution to the following linear system:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \mathrm{TAG}_1 & \cdots & \mathrm{TAG}_1^t \\ \vdots & \vdots & & \vdots \\ 1 & \mathrm{TAG}_t & \cdots & \mathrm{TAG}_t^t \end{bmatrix} \begin{bmatrix} \alpha \\ a_1 \\ \vdots \\ a_t \end{bmatrix} = \begin{bmatrix} \alpha \\ \gamma_1 \\ \vdots \\ \gamma_t \end{bmatrix}$$

In particular, each of $a_1, \ldots, a_t$ may be written as a linear combination of $\alpha, \gamma_1, \ldots, \gamma_t$ (where the coefficients are efficiently computable given $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_t$) and therefore each of $g^{a_1}, \ldots, g^{a_t}$ may be written as a product of $g^{\alpha}, g^{\gamma_1}, \ldots, g^{\gamma_t}$ raised to the appropriate powers.

- $\mathsf{SetupSim}(\mathrm{PP}, 1^t)$ returns $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$ as computed above and $(\mathrm{SK}_{\mathrm{TAG}_1}, \ldots, \mathrm{SK}_{\mathrm{TAG}_t}) := (\gamma_1, \ldots, \gamma_t)$

Remark 1. Instead of setting $\mathrm{HK} := (g^{a_1}, \ldots, g^{a_t})$, we may also set $\mathrm{HK} := (g^{f(1)}, \ldots, g^{f(t)})$. Public evaluation and t-simulation then proceed via Lagrange interpolation in the exponent.

Public Verifiability. Given $(\mathrm{HK}, u, \mathrm{TAG}, \tau)$, checking that $\tau = \mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ corresponds exactly to verifying that $(g, g^{f(\mathrm{TAG})}, u, \tau)$ is a valid DDH tuple. Given HK and TAG, we may compute $g^{f(\mathrm{TAG})}$ using $\left(g^{\alpha} \cdot \prod_{i=1}^{t} (g^{a_i})^{\mathrm{TAG}^i}\right)$. This implies public verifiability in bilinear groups.

For general groups that do not admit a bilinear pairing, we may realize public verifiability in the random oracle model following [41, Section 4.3] (also [28]). That is, we append to $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ non-interactive zero-knowledge proof that $(g, g^{f(\mathrm{TAG})}, u, u^{f(\mathrm{TAG})})$ is a valid DDH tuple. Such a proof is derived by applying the Fiat-Shamir heuristic to Chaum-Pedersen Σ-protocol for the langugage comprising valid DDH tuples [13]; soundness of the verification algorithm follows immediately from soundness of the Σ-protocol. Handling public and private evaluation requires more care, and we proceed differently depending on the application:

- In the application to threshold cryptosystems, we instantiate the Chaum-Pedersen protocol so that there is an efficient prover given $f(\mathrm{TAG})$ as a witness. This

guarantees efficient private evaluation. For public evaluation, we rely on the zero-knowledge simulator, which in turn requires programming the random oracle. This is not an issue since we only rely on public evaluation in the proof of security.

– In the application to revocation cryptosystems, we instantiate the Chaum-Pedersen protocol so that there is an efficient prover given r such that $u = g^r$ as a witness. This guarantees efficient public evaluation. For private evaluation, we rely on the zero-knowledge simulator, which again requires programming the random oracle. This is not an issue since in the decryption algorithm, Ext ignores non-interactive zero-knowledge proof.

8 Instantiations from Hardness of Factoring

Fix a Blum integer $N = PQ$ for safe primes $P, Q \equiv 3 \pmod 4$ (such that $P = 2p+1$ and $Q = 2q + 1$ for primes p, q). Following [33], we work over the cyclic group of signed quadratic residues, given by the quotient group $\mathbb{QR}_N^+ := \mathbb{QR}_N / \pm 1$. $\mathbb{QR}_N^+$ is a cyclic group of order pq and is efficiently recognizable (by verifying that the Jacobi symbol is $+1$). In addition, the map $x \mapsto x^2$ is a permutation over $\mathbb{QR}_N^+$. Furthermore, assuming that factoring is hard on average and that safe primes are dense, the family of permutations $x \mapsto x^2$ (indexed by N) acting on the groups $\mathbb{QR}_N^+$ is one-way.

In our constructions, the public parameter PP comprises (N, g), where N is a random $2k$-bit Blum integer and g is chosen uniformly from $\mathbb{QR}_N^+$. We will henceforth assume that g is a generator for $\mathbb{QR}_N^+$, which happens with probability $1 - O(1/\sqrt{N})$. For signatures, we consider the relation

$$\mathsf{R}_{\mathsf{PP}}^{\mathsf{sqr}} = \left\{(u, s) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ : u = s^2\right\}$$

For encryption, we consider the relation:

$$\mathsf{R}_{\mathsf{PP}}^{\mathsf{isqr}} = \left\{(u, s) \in \mathbb{QR}_N^+ \times \mathbb{QR}_N^+ : u = s^{2^k}\right\}$$

Here, we focus on the latter relation. The associated sampling algorithm SampR picks a random $r \in [(N-1)/4]$ and outputs $(g^{2^k r}, g^r)$. Note that the output distribution is statistically close to the uniform distribution over $\mathbb{QR}_N^+$ whenever g is a generator. Using the Blum-Blum-Shub (BBS) pseudorandom generator [4], we may extract k hard-core bits from s that are pseudorandom even given u, that is:

$$\mathsf{G}_{\mathsf{PP}}^{\mathsf{bbs}}(s) := (\mathsf{lsb}_N(s), \mathsf{lsb}_N(s^2), \ldots, \mathsf{lsb}_N(s^{2^{k-1}}))$$

Basic Idea. The next claim shows that we can do Shamir secret sharing over a composite modulus:

Claim (implicit in [40]). Fix two primes $p < q$. For any $t < p$ and any $t + 1$ distinct values $v_1, \ldots, v_{t+1}$ in $\{1, 2, \ldots, p\}$, the map $\psi : (a_0, a_1, \ldots, a_t) \mapsto (f(v_1), f(v_2), \ldots, f(v_{t+1}))$ where $f(x) := a_0 + a_1 x + \cdots + a_t x^t$ defines a bijection from $\mathbb{Z}_{pq}^{t+1}$ to $\mathbb{Z}_{pq}^{t+1}$.

proof (*sketch*). That ψ is injective follows from the fact that any polynomial of degree t over $\mathbb{Z}_{pq}$ that vanishes at the $t+1$ distinct values $v_1, \ldots, v_{t+1}$ must be identically 0 modulo p and modulo q and thus identically 0 modulo pq (by the Chinese remainder theorem). That ψ is surjective follows via Lagrange polynomial interpolation (since the pairwise differences $v_i - v_j$ are all coprime with pq). □

Threshold Hash Proof System. Fix the parameters $(\text{PP}, \text{SP}) = (N, \phi(N))$. The tag space is given by $Z_{\sqrt{N}/4}$. Note that $\sqrt{N}/4 \leq \min\{p, q\}$ so every valid tag is coprime with $\phi(N)/4$. SampR takes an additional $g \in \mathbb{QR}_N^+$ which is provided as part of HK, and $\mathsf{SampR}(r) := (u, s)$ where $s = g^{2^{tk}r}, u = s^{2^k} = g^{2^{(t+1)k}r}$.

(KEY GENERATION.) Pick $a_1, \ldots, a_t \leftarrow_{\text{R}} \mathbb{Z}_{\phi(N)/4}$ and set $f(x) := 2^{-(t+1)k} + a_1 x + \cdots a_t x^t$. In addition, pick $g \leftarrow_{\text{R}} \mathbb{QR}_N^+$.

- $\mathsf{Setup}(\text{PP}, \text{SP}, 1^t)$ returns $\text{HK} := (g, g^{2^{(t+1)k}a_1}, \ldots, g^{2^{(t+1)k}a_t})$ and $\text{MSK} := (f(x), \phi(N))$.
- $\mathsf{ShareGen}(\text{MSK}, \text{TAG})$ returns $\text{SK}_{\text{TAG}} := f(\text{TAG}) \pmod{\phi(N)/4}$.

(PUBLIC/PRIVATE EVALUATION.) $\mathcal{H}_{\text{HK}}(\text{TAG}, u)$ is given by $u^{f(\text{TAG})} = (g^{2^{(t+1)k}f(\text{TAG})})^r$ where $u := g^{2^{(t+1)k}r}$

- $\mathsf{Pub}(\text{HK}, u, r)$ returns $\left(g \cdot \prod_{i=1}^{t} (g^{2^{(t+1)k}a_i})^{\text{TAG}^i}\right)^r$.
- $\mathsf{Priv}(\text{SK}_{\text{TAG}}, u)$ returns $u^{\text{SK}_{\text{TAG}}}$.

($(t+1)$-EXTRACTION.) Given $\text{TAG}_1, \ldots, \text{TAG}_{t+1}$, we may efficiently compute the fractional Lagrangian coefficients $L_1, \ldots, L_{t+1}$ such that $f(0) = \sum_{i=1}^{t+1} L_i \cdot f(\text{TAG}_i) \pmod{\phi(N)/4}$. In addition, we may compute

$$D := \text{lcm}\left\{\prod\nolimits_{j \neq i} (\text{TAG}_i - \text{TAG}_j) : i \in [t+1]\right\}.$$

We make the following observations: (1) $DL_1, \ldots, DL_{t+1}$ are all integers, so we may compute $u^{D \cdot f(0)} = \prod_{i=1}^{t+1} \tau_i^{DL_i}$; (2) let 2^c be the highest power of 2 that divides D, and we have $c \leq kt$ (since $|\text{TAG}_i - \text{TAG}_j| \leq 2^k$); and (3) given $u = s^{2^k}$ and $u^{2^{kt-c}D \cdot f(0)} = s^{2^{-c}D}$, we may efficiently recover s using Shamir's "GCD in the exponent" algorithm [39], since $\gcd(2^k, 2^{-c}D) = 1$.

- $\mathsf{Ext}(u, \tau_1, \ldots, \tau_{t+1})$ returns s as computed above.

(t-SIMULATION.) Pick $\gamma_1, \ldots, \gamma_t \leftarrow_{\text{R}} \mathbb{Z}_{N/4}$[1]. This uniquely determines a degree t polynomial $f(x) = 2^{-(t+1)k} + a_1 x + \cdots + a_t x^t$ such that $f(\text{TAG}_i) = \gamma_i$

[1] This yields a distribution that is statistically close to picking $\gamma_1, \ldots, \gamma_t \leftarrow_{\text{R}} \mathbb{Z}_{\phi(N)/4}$.

$(\mathrm{mod}\ \phi(N)/4)$ for $i = 1, \ldots, t$. Moreover, we $a_1, \ldots, a_t$ are given by the solution to the following linear system:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 1 & \mathrm{TAG}_1 & \cdots & \mathrm{TAG}_1^t \\ \vdots & \vdots & & \vdots \\ 1 & \mathrm{TAG}_t & \cdots & \mathrm{TAG}_t^t \end{bmatrix} \begin{bmatrix} 1 \\ 2^{(t+1)k} a_1 \\ \vdots \\ 2^{(t+1)k} a_t \end{bmatrix} = \begin{bmatrix} 1 \\ 2^{(t+1)k} \gamma_1 \\ \vdots \\ 2^{(t+1)k} \gamma_t \end{bmatrix}$$

Let $D := \prod_{1 \le i < j \le t} (\mathrm{TAG}_i - \mathrm{TAG}_j)$ (the determinant of the Vandermonde matrix on the left). Then, we may efficiently compute the integer values $D \cdot 2^{(t+1)k} a_1, \ldots, D \cdot 2^{(t+1)k} a_t$ (given $\gamma_1, \ldots, \gamma_t$ and $\mathrm{TAG}_1, \ldots, \mathrm{TAG}_t$) without computing any modular inverse.

- $\mathsf{SetupSim}(\mathrm{PP}, 1^t)$ picks $\tilde{g} \leftarrow_{\mathrm{R}} \mathbb{QR}_N^+$ and returns $(\mathrm{SK}_{\mathrm{TAG}_1}, \ldots, \mathrm{SK}_{\mathrm{TAG}_t}) := (\gamma_1, \ldots, \gamma_t)$ and $\mathrm{HK} := (\tilde{g}^D, \tilde{g}^{D \cdot 2^{(t+1)k} a_1}, \ldots, \tilde{g}^{D \cdot 2^{(t+1)k} a_t})$.

Public Verifiability. Following [28, Section 4], it suffices to construct a Σ-protocol for DDH-tuples over $\mathbb{QR}_N^+$, that is, (g, g^B, u, u^B). The honest sender is given the witness r such that $u = g^r$, picks $s \in [N^3]$ and sends $(g_0, g_1) := (g^s, g^{Bs})$. Upon receiving a challenge $e \in [N/4]$, it responds with $z := s + re$. The analysis is entirely analogous to that in [28].

9 Chosen-Ciphertext Security

9.1 Broadcast CCA

In this section, we construct revocation schemes secure against CCA attacks.

Public Verifiability, Signatures and Random Oracles. As stated, the construction also requires public verifiability and a one-time signature scheme. For factoring-based instantiations and Diffie-Hellman in general groups, we already rely on a random oracle to implement public verifiability, so we may as well also rely on the random oracle to instantiate an efficient signature scheme[2]. For Diffie-Hellman in bilinear groups (which satisfy public verifiability without random oracles), we can avoid the use of signatures and instead use a TCR – the ciphertext is given by (u, τ) where $\tau := \mathsf{Pub}(\mathrm{PK}, \mathrm{TAG}, u, r)$ and $\mathrm{TAG} := \mathsf{TCR}(u)$, and as such, we obtain efficient constructions without random oracles.

Theorem 4. *If* R_{PP} *is a one-way relation admitting a threshold extractable hash proof with public verifiability, then the above revocation scheme is* IND-CCA *secure.*

[2] The reason we need a signature scheme is that the addition of the non-interactive proof of membership in the random oracle to provide public verifiability means that (TAG, u) no longer uniquely determines an accepting proof.

Revocation PKE

(SHARING PHASE.) On input the security parameter 1^k and a revocation threshold t, the dealer generates (PP, SP), runs $\mathsf{Setup}(\text{PP}, \text{SP}, 1^{t+1}) \rightarrow (\text{HK}, \text{MSK})$ and sets the public key PK to be (PP, HK). A user with identity ID is given the key $\text{SK}_{\text{ID}} := \mathsf{ShareGen}(\text{MSK}, \text{ID})$.

(ENCRYPTION.) In order to revoke users $\text{ID}_1, \ldots, \text{ID}_t$, $\mathsf{Enc}(\text{PK})$:

1. generate a verification VKSIG for a one-time signature scheme;
2. sample $(u, s) := \mathsf{SampR}(r)$;
3. compute $\tau := \mathsf{Pub}(\text{PK}, \text{VKSIG}, u, r)$ and $\tau_i := \mathsf{Pub}(\text{HK}, \text{ID}_i, u, r)$ for $i = 1, \ldots, t$;
4. compute the signature σ on $(u, \tau, \tau_1, \ldots, \tau_t)$;
5. return $(C, K) := ((\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma), \mathsf{G}(s))$.

(DECRYPTION.) Any user ID not in the revoked set $\{\text{ID}_1, \ldots, \text{ID}_t\}$ may decrypt a ciphertext $C := (\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma)$ as follows:

1. verify proofs $\tau, \tau_1, \ldots, \tau_t$ and signature σ; output $\perp$ if any of these tests fails;
2. compute $s := \mathsf{Ext}(u, \tau, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\text{SK}_{\text{ID}}, u))$ and output $\mathsf{G}(s)$.

Fig. 7. CCA-secure revocation scheme from threshold hash proofs

proof (*sketch*). In the following, we write $(u^*, s^*) = \mathsf{SampR}(r), C^* = (u^*, \tau^*)$, K_0^*, K_1^* to denote the challenge ciphertext and keys chosen by the IND-CCA experiment, and we set VKSIG^* to denote the verification key used in computing C^*. We proceed via a sequence of games. We start with Game 0, where the challenger proceeds like in the standard IND-CCA game (i.e, K_0^* is a real key and K_1^* is a random key) and end up with a game where both K_0^* and K_1^* are chosen uniformly at random. Then, we show that all games are indistinguishable under the assumption that $\mathsf{G}(s)$ is pseudorandom even given u.

GAME 1: ELIMINATING COLLISIONS. We replace the decapsulation mechanism Dec with Dec' that outputs $\perp$ on inputs $(\text{VKSIG}, u, \tau, \sigma)$ such that $\text{VKSIG} = \text{VKSIG}^*$ but otherwise proceeds like Dec. We show that Games 0 and 1 are computationally indistinguishable, by arguing that Dec and Dec' essentially agree on all inputs. This follows readily from the security of the one-time signature.

GAME 2: DECAPSULATION WITH SetupSim. We modify the IND-CCA experiment from Game 1, we generate the keys using $\mathsf{SetupSim}(\text{PP}, \text{ID}_1^*, \ldots, \text{ID}_t^*, \text{VKSIG}^*)$, and we replace $\mathsf{Dec}'(\text{ID}, \cdot)$ with $\mathsf{Dec}^*(\text{ID}, \cdot)$:

On input $(\text{VKSIG}, u, \tau, \tau_1, \ldots, \tau_t, \sigma)$:

- if $\text{VKSIG} = \text{VKSIG}^*$, return $\perp$.
- if σ or any of $\tau, \tau_1, \ldots, \tau_t$ fails to verify, return $\perp$.
- compute $s := \mathsf{Ext}(u, \tau, \tau_1, \ldots, \tau_t, \mathsf{Priv}(\text{SK}_{\text{VK}^*}, u))$ and output $\mathsf{G}(s)$.

Here, we use the fact that in both Dec' and Dec^*, we run Ext with $t+2$ valid proofs and therefore it outputs the correct witness s.

GAME 3: ENCAPSULATION WITH Priv. We compute all $t+1$ proofs $\tau, \tau_1, \ldots, \tau_t$ in C^* using Priv instead of Pub and sign using the secret key corresponding to VK^*.

GAME 4: REPLACING $\mathsf{G}(s^*)$ WITH RANDOM. We generate K_0^* at random from $\{0,1\}^k$ instead of using $\mathsf{G}(s^*)$ (recall here that $(u^*, s^*) = \mathsf{SampR}(r)$). Observe that in Game 3, we never use knowledge of the witness s^* or randomness r associated with u^*. It follows from the pseudorandomness of G that Games 3 and 4 are computationally indistinguishable. Specifically, we may transform any distinguisher for Games 3 and 4 into a distinguisher K_0^* and $\mathsf{G}(s^*)$.

We conclude by observing that in Game 4, both K_0^* and K_1^* are identically distributed, so the probability that $b' = b$ is exactly $1/2$. □

9.2 Threshold CCA

In this section, we construct threshold encryption schemes secure against CCA attacks. The main technical difficulty is as follows: the simulator knows $f(\mathrm{TAG}_1^*), \ldots, f(\mathrm{TAG}_t^*)$, and needs to be able to compute $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u) = u^{f(\mathrm{TAG})}$ given any TAG, u. This is in order to simulate the $\mathsf{ShareDec}(\mathrm{TAG}, \cdot)$, the decryption share for some user TAG. To handle this, we modify our basic threshold encryption scheme in three ways:

- The first modification is to add to the ciphertext which contains the instance u, a publicly-verifiable 1-threshold extractable hash proof —or, an all-but-one extractable hash proof following the terminology in [42]— for the relation $(u, u^{f(0)})$. For this, we need to turn to either pairings or the random oracle model. (Similar issues arise even in previous discrete-log based schemes not based on pairings.) This way, simulator will be able to recover the $t+1$ values $u^{f(0)}, u^{f(\mathrm{TAG}_1^*)}, \ldots, u^{f(\mathrm{TAG}_t^*)}$ for any well-formed ciphertext.
- Next, using interpolation, we can recover $u^{Df(\mathrm{TAG})}$, where D is some factor we use to clear out the fractional Lagrangian coefficients. In the discrete-log based instantiations, we may compute D^{-1} and thus recover $u^{f(\mathrm{TAG})}$. In the factoring-based instantation, we will modify the hash function $\mathcal{H}_{\mathrm{HK}}(\mathrm{TAG}, u)$ to be $u^{Df(\mathrm{TAG})}$; as such, we can only support a fixed identity space of polynomial size, since we need to compute D in advance. Similarly, we will need to modify Ext so that it computes $u^{D^2 f(0)}$ via Lagrange interpolation and s from both $u^{D^2 f(0)}$ and u via Shamir's "GCD in the exponent" algorithm.
- Finally, we modify $\mathsf{ShareDec}(\cdot)$ so that it will only output the decryption share if the 1-threshold extractable hash proof verifies properly.

The details are deferred to the full version of this paper.

Acknowledgments

I would like to thank Dan Boneh, Mario Szegedy and Moti Yung for insightful discussions. I am also very grateful to the anonymous referees for detailed and

constructive feedback. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government.

References

[1] Abdalla, M., Bellare, M., Rogaway, P.: The oracle diffie-hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (2001)

[2] Agrawal, S., Boneh, D., Boyen, X.: Efficient lattice (H)IBE in the standard model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)

[3] Bendlin, R., Damgård, I.: Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 201–218. Springer, Heidelberg (2010)

[4] Blum, L., Blum, M., Shub, M.: Comparison of two pseudo-random number generators. In: CRYPTO 1982, pp. 61–78 (1982)

[5] 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. 289–299. Springer, Heidelberg (1985)

[6] Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: STOC, pp. 103–112 (1988)

[7] Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)

[8] Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology 17(4), 297–319 (2004)

[9] Boneh, D., Boyen, X., Halevi, S.: Chosen ciphertext secure public key threshold encryption without random oracles. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 226–243. Springer, Heidelberg (2006)

[10] Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: ACM CCS, pp. 320–329 (2005)

[11] Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90–106. Springer, Heidelberg (1999)

[12] 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)

[13] Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)

[14] 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)

[15] 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)

[16] Damgård, I., Jurik, M.: A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)

[17] De Santis, A., Persiano, G.: Zero-knowledge proofs of knowledge without interaction. In: FOCS, pp. 427–436 (1992)
[18] De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: STOC, pp. 522–533 (1994)
[19] Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (1988)
[20] Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
[21] Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures (extended abstract). In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
[22] 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)
[23] Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994)
[24] Fouque, P.-A., Pointcheval, D.: Threshold cryptosystems secure against chosen-ciphertext attacks. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 351–368. Springer, Heidelberg (2001)
[25] Fouque, P.-A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
[26] Frankel, Y., Gemmell, P., Yung, M.: Witness-based cryptographic program checking and robust function sharing. In: STOC, pp. 499–508 (1996)
[27] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 354–371. Springer, Heidelberg (1996)
[28] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and efficient sharing of RSA functions. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 157–172. Springer, Heidelberg (1996)
[29] Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999)
[30] Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T.: Threshold RSA for dynamic and ad-hoc groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 88–107. Springer, Heidelberg (2008)
[31] Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: STOC, pp. 25–32 (1989)
[32] 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)
[33] Hofheinz, D., Kiltz, E.: The group of signed quadratic residues and applications. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 637–653. Springer, Heidelberg (2009)
[34] Katz, J., Yung, M.: Threshold cryptosystems based on factoring. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer, Heidelberg (2002)
[35] 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)
[36] Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 1–20. Springer, Heidelberg (2001)

[37] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437 (1990)
[38] 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)
[39] Shamir, A.: On the generation of cryptographically strong pseudorandom sequences. ACM Trans. Comput. Syst. 1(1), 38–44 (1983)
[40] Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
[41] Shoup, V., Gennaro, R.: Securing threshold cryptosystems against chosen ciphertext attack. J. Cryptology 15(2), 75–96 (2002)
[42] Wee, H.: Efficient chosen-ciphertext security via extractable hash proofs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 314–332. Springer, Heidelberg (2010)

Deniable Encryption with Negligible Detection Probability: An Interactive Construction

Markus Dürmuth[1,⋆] and David Mandell Freeman[2,⋆⋆]

[1] Ruhr-University Bochum, Germany
markus.duermuth@rub.de
[2] Stanford University, USA
dfreeman@cs.stanford.edu

Abstract. *Deniable encryption*, introduced in 1997 by Canetti, Dwork, Naor, and Ostrovsky, guarantees that the sender or the receiver of a secret message is able to "fake" the message encrypted in a specific ciphertext in the presence of a coercing adversary, without the adversary detecting that he was not given the real message. To date, constructions are only known either for weakened variants with separate "honest" and "dishonest" encryption algorithms, or for single-algorithm schemes with non-negligible detection probability.

We propose the first sender-deniable public key encryption system with a single encryption algorithm and negligible detection probability. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

Keywords: Deniable encryption, electronic voting, multi-party computation.

1 Introduction

One of the central goals of cryptography is protecting the secrecy of a transmitted message. The secrecy property of an encryption scheme is usually formalized as semantic security [10], which guarantees that an adversary cannot gain even partial information about an encrypted message.

The notion of semantic security has proven to be very useful in a large number of applications. However, there are some scenarios where semantic security is not sufficient. For example, semantic security does not ensure message secrecy if the adversary can coerce the sender or the receiver of a message to reveal the secret keys and/or the randomness that was used to form an encryption. Specifically, semantic security does not prevent an encryption scheme from being committing, in the sense that if an adversary sees a ciphertext and then tries to coerce the sender to reveal all of the input to the encryption (message and randomness), any inputs that the sender can reveal that are consistent with the ciphertext must reveal the true message encrypted. In fact, many encryption schemes have only one set of possible inputs per ciphertext.

⋆ Research conducted at Stanford University.
⋆⋆ Supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship.

K.G. Paterson (Ed.): Eurocrypt 2011, LNCS 6632, pp. 610–626, 2011.

This committing property of encryption can be problematic in applications such as electronic voting [15] or keeping information secret when facing a coercer using physical force, or in the case of secure multi-party computation in the presence of an adaptive adversary [4].

Deniable encryption, introduced by Canetti, Dwork, Naor, and Ostrovsky in 1997 [3], possesses a stronger security property than semantic security that avoids these shortcomings. Informally, a (possibly interactive) public key encryption scheme is *sender-deniable* if, given an encryption $c = \mathsf{Enc}_{\mathsf{pk}}(m_1; r_1)$ of a message m_1 with randomness r_1, for every message m_2 the sender can compute alternative randomness r_2 such that $(\mathsf{Enc}_{\mathsf{pk}}(m_1; r_1), r_1)$ and $(\mathsf{Enc}_{\mathsf{pk}}(m_2; r_2), r_2)$ are computationally indistinguishable. Thus when coerced, the sender can reveal either the "real" randomness r_1 or the "fake" randomness r_2, and the adversary cannot tell whether m_1 or m_2 was really encrypted in the ciphertext c. A similar property defines *receiver-deniability*, and a system with both properties is *sender-and-receiver-deniable* or *bi-deniable*.

Canetti et al. also considered a weaker form of deniability, called *flexible deniability*. In a flexibly deniable system, the sender has two different encryption algorithms, *honest* and *dishonest* encryption. At the time of encryption the sender chooses either honest or dishonest encryption. When choosing honest encryption he is unable to fake later when coerced, but when choosing dishonest encryption he can create randomness such that the distribution matches randomness for *honest* encryption. What makes this notion weak for practical purposes is that if the adversary believes that the sender used the dishonest encryption, then the adversary can coerce the sender to reveal randomness for the dishonest encryption, and the sender looses his ability to fake. The system's security relies on the fact that the adversary really believes that the sender used the honest encryption.

Canetti et al. constructed a non-interactive sender-deniable encryption scheme that transmits bits, based on a primitive called *translucent sets* that can be realized under trapdoor permutations and other standard assumptions. The system is semantically secure, but it has only $1/O(n)$-deniability. That is, an adversary can detect whether the user is "faking" messages and randomness with probability $1/O(n)$, where n is the system's security parameter. Canetti et al. left open the problem of constructing an encryption scheme with negligible deniability. This problem has remained open for more than 13 years.

Our Contribution. We give the first public key encryption scheme that satisfies the definition of sender-deniability in [3] with a single encryption algorithm and negligible probability of detection. We describe a generic interactive construction based on a public key bit encryption scheme that has certain properties, and we give two examples of encryption schemes with these properties, one based on the quadratic residuosity assumption and the other on trapdoor permutations.

1.1 Overview of Our Construction

The basic idea of our construction is the following. We take a public key bit encryption scheme with dense ciphertexts, in which a uniformly random ciphertext decrypts to a uniformly random message. To encrypt a bit $b \in \{0, 1\}$, we first obtain $4n + 1$ public

keys for the underlying encryption scheme. We construct $n+1$ encryptions of b, construct n encryptions of $1-b$, and sample $2n$ random ciphertexts, each under a different public key, and then permute the output randomly. Decrypting all ciphertexts individually and taking the majority recovers the original message with noticeable probability. Repeating the protocol multiple times in parallel reduces the decryption error.

To fake the sender's input, we claim that a constructed ciphertext encrypting b was sampled randomly. This trick has been used before, but it usually gives an adversary a non-negligible probability of detecting the faking (e.g., inverse linear in the ciphertext length as in [3]). What is unique in our construction is that we additionally construct "real-looking" randomness for a *sampled* ciphertext encrypting $1-b$, so we obtain a distribution which is computationally indistinguishable with *negligible* advantage for any efficient adversary.

In order to compute fake randomness for a sampled ciphertext, we need two ingredients. First, the encryption scheme needs to have the property that given a ciphertext, the secret key holder can compute randomness that is indistinguishable from the randomness used to compute the ciphertext. Second, the sender must know which sampled ciphertext she should use in her faking and she must obtain a secret key for that ciphertext. (After all, she doesn't know which sampled ciphertexts encrypt b and which encrypt $1-b$.) On the other hand, the receiver doesn't know which ciphertexts were sampled and which were constructed. Our basic idea for giving the sender the necessary information is to have the receiver send back pairs of indices for ciphertexts that decrypt to opposite plaintexts, so that with high probability one of the pairs corresponds to a constructed encryption of b and a sampled encryption of $1-b$. The sender then indicates one such pair and obtains from the receiver the secret keys for both elements of that pair. Since the two ciphertexts encrypt opposite messages, these revealed values do not compromise the secrecy of the system.

Once in possession of the correct secret key, the sender can fake randomness. Deniability ultimately rests on the semantic security of the underlying encryption scheme, as our manipulation changes the distribution on the set of sampled ciphertexts whose indices were *not* sent back to the sender in the above process.

To instantiate our system, we observe that two well known encryption schemes have the properties necessary for our construction: the Goldwasser-Micali bit encryption scheme based on the quadratic residuosity assumption [10], and a simple bit encryption scheme constructed from a trapdoor permutation using a hard-core predicate.

1.2 Related Work

In addition to their sender-deniable scheme with $1/O(n)$-deniability, Canetti et al. [3] also constructed a flexible (i.e., two-algorithm) sender-deniable encryption scheme with negligible deniability. More recently, O'Neill, Peikert, and Waters [12] announced a flexible *bi*-deniable encryption scheme with negligible deniability based on lattice assumptions. We view this latter work as orthogonal to our own: it is non-interactive and achieves deniability for both sender and receiver simultaneously, but the construction uses in an essential way the fact that there are different honest and dishonest encryption algorithms.

A related concept originating from adaptive security of multi-party computation is *non-committing encryption* [4,6,8,2]. Informally, a public key bit encryption scheme is non-committing if a simulator can efficiently sample a distribution of ciphertexts c and two sets of randomness r_0, r_1 such that the ciphertext c along with r_b is indistinguishable from a legitimate encryption of b along with the true randomness. The main difference between non-committing encryption and deniable encryption is while the *simulator* can generate ciphertexts and randomness corresponding to either message, a *user* cannot, in general, compute randomness that reveals a different message than was actually encrypted. While deniable encryption implies non-committing encryption, the converse does not hold; in particular, the non-committing encryption scheme in [4] is not deniable.

A different concept is *plausible deniability*. This term usually describes engineering techniques that allow one to deny the existence of encrypted data or knowledge of the secret key. A well known example is the TrueCrypt file system encryption [17], where one can add secret containers inside an encrypted volume such that one can deny their existence. Another example [5] uses steganographic measures to hide encrypted data in cover text. These techniques come without formal proof or even definition, and attacks exist that can reveal the presence and content of encrypted data [7]. In addition, this form of deniability usually is not suitable for online applications such as electronic voting.

1.3 Outline

In Section 2 we give the formal definition of a deniable encryption scheme. In Section 3 we describe the building block for our deniable protocol, which we call a *samplable* public key bit encryption scheme. In Section 4 we construct an interactive encryption scheme from a samplable public key bit encryption scheme. We prove our scheme is secure and deniable under the assumption that the underlying scheme is samplable and semantically secure. In Section 5 we describe two samplable public key bit encryption schemes, one based on the quadratic residuosity assumption and one based on trapdoor permutations. Finally, in Section 6 we discuss some open questions related to our work.

2 Deniable Encryption

We begin by fixing some notation. If X is a set and Δ is a distribution on X, we use $x \leftarrow \Delta$ to denote an element of X sampled according to the distribution Δ, and we use $x \leftarrow y$ to denote assignment of the value y to x. If X is finite we use $x \xleftarrow{\mathrm{R}} X$ to denote an element sampled uniformly at random from X. For a two-party protocol π between S and R we write

$$(o_S, o_R, tr) \leftarrow \pi((i_S; r_S), (i_R; r_R))$$

for the execution of π with input i_S, i_R and randomness r_S, r_R from S and R, respectively, producing output o_S, o_R for the respective parties, and a public transcript tr.

A function $\mu\colon \mathbb{N} \to \mathbb{R}$ is said to be *negligible* iff for all $c \in \mathbb{N}$ we have that $|\mu(n)| \leq \frac{1}{n^c}$ for sufficiently large n. If this is the case, we write $\mu(n) = \mathrm{negl}(n)$. A probability p is said to be *overwhelming* if $p = 1 - \mathrm{negl}(n)$.

Two sequences of random variables $(X_n)_{n\in\mathbb{N}}, (Y_n)_{n\in\mathbb{N}}$ are $\mu(n)$-*computationally indistinguishable*, and denoted by $(X_n) \approx^{\mu(n)} (Y_n)$, if for all polynomial-time adversaries $\mathcal{A}$ we have $|\Pr[\mathcal{A}(x) = 1; x \leftarrow X_n] - \Pr[\mathcal{A}(x) = 1; x \leftarrow Y_n]| \leq \mu(n)$ for sufficiently large n. We say (X_n) and (Y_n) are *statistically indistinguishable*, denoted $(X_n) \approx (Y_n)$, if the same holds for *all* adversaries $\mathcal{A}$, regardless of running time.

Our definition of deniable encryption is a slight rephrasing of the definition of Canetti et al. [3].

Definition 2.1. Let $n \in \mathbb{N}$ be a security parameter. An efficiently computable protocol[1] π between two parties S and R (*sender* and *receiver*, respectively) is called a $\mu(n)$-*sender-deniable public key bit encryption scheme* if the following three conditions are satisfied:

Correctness: We say π is *correct* if for all messages $b \in \{0,1\}$ we have

$$\Pr\left[b' \neq b;\ r_S, r_R \xleftarrow{R} \{0,1\}^*, (\cdot, b', \cdot) \leftarrow \pi((b; r_S), (n; r_R))\right] \ \leq\ \nu(n)$$

for some negligible function $\nu\colon \mathbb{N} \to \mathbb{R}$.

Passive secrecy: The two random variables tr_0, tr_1 defined by

$$\begin{aligned} r_S, r_R \xleftarrow{R} \{0,1\}^*,\ & (\cdot,\cdot,tr_0) \leftarrow \pi((0; r_S), (n; r_R)), \\ & (\cdot,\cdot,tr_1) \leftarrow \pi((1; r_S), (n; r_R)), \end{aligned} \tag{2.1}$$

are $\nu(n)$-computationally indistinguishable for some negligible $\nu(n)$.

Deniability: There is an efficient *faking algorithm* Fake that takes input a message $b \in \{0,1\}$, sender randomness $r_S \in \{0,1\}^*$, and a transcript of the protocol π, such that for any $b \in \{0,1\}$ and

$$\begin{aligned} r_S, r_R \xleftarrow{R} \{0,1\}^*,\ & (\cdot,\cdot,\widetilde{tr}) \leftarrow \pi((1-b; r_S), (n; r_R)), \\ \tilde{r}_S \leftarrow \mathsf{Fake}(b, r_S, \widetilde{tr}),\ & (\cdot,\cdot,tr) \leftarrow \pi((b; r_S), (n; r_R)), \end{aligned} \tag{2.2}$$

the following two distributions are $\mu(n)$-computationally indistinguishable:

$$(b, r_S, tr) \approx^{\mu(n)} (b, \tilde{r}_S, \widetilde{tr}). \tag{2.3}$$

The distribution on the left is produced by a real encryption of b with randomness r_S. The distribution on the right is produced when we actually encrypt $1-b$ with randomness r_S, but tell the adversary that we encrypted b with randomness $\tilde{r}_S$.

We call π a *sender-deniable public key bit-encryption scheme* if it is $\mu(n)$-sender-deniable for a negligible $\mu(n)$. We can define a *receiver-deniable* encryption scheme analogously by having Fake produce fake receiver randomness $\tilde{r}_R$. A *sender-and-receiver-deniable* or *bi-deniable* scheme has both properties, with a faking algorithm for each party.

A straightforward reduction shows that deniability (with negligible $\mu(n)$) implies passive secrecy.

[1] I.e., all computations run in (expected) time polynomial in n and the number of rounds is polynomial in n.

Lemma 2.2. *A two-party protocol π that has the deniability property of Definition 2.1 for a negligible $\mu(n)$ also has the passive secrecy property.*

Proof. Assume there exists an efficient adversary $\mathcal{A}$ that can distinguish transcripts of π corresponding to messages 0 and 1 with non-negligible probability. Now if we are given a challenge (m, r_S, tr) for deniability, we can use $\mathcal{A}$ to decide if tr is an encryption of b or $1-b$, and thus can win the deniability game with the same probability and the same running time. This contradicts the hypothesis that π has the deniability property for negligible $\mu(n)$. □

3 Samplable Public Key Encryption

As a building block for our deniable encryption scheme, we use a public key bit encryption scheme for which a secret key holder can recover randomness used in the encryption. We do not require the recovered randomness to be exactly that used to encrypt, as this may be impossible to compute, but rather that it be indistinguishable from the real randomness. In this section we formalize this idea.

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption system encrypting messages in $\{0,1\}$. We write the encryption algorithm as

$$\mathsf{Enc}\colon \mathcal{PK} \times \{0,1\} \times \mathcal{R} \to \mathcal{C}$$

and the decryption algorithm as

$$\mathsf{Dec}\colon \mathcal{SK} \times \mathcal{C} \to \{0,1\},$$

where $\mathcal{PK}$, $\mathcal{SK}$, $\mathcal{R}$, and $\mathcal{C}$ are the spaces of public keys, secret keys, sender randomness, and ciphertexts, respectively. The key spaces $\mathcal{PK}$ and $\mathcal{SK}$ depend on the security parameter n, and the spaces $\mathcal{R}$ and $\mathcal{C}$ also depend on the public key pk used in the encryption; for the sake of readability we omit these dependencies from the notation.

We denote the encryption of a bit b under public key pk with randomness r by $\mathsf{Enc}_{\mathsf{pk}}(b; r)$, and the system's security parameter (input to KeyGen) by n. We let $\Delta_{\mathcal{R}}$ denote the distribution on $\mathcal{R}$ that the encryption algorithm samples. We require the usual correctness condition: for all $\mathsf{pk} \in \mathcal{PK}$, all $\mathsf{sk} \in \mathcal{SK}$, and $b \in \{0,1\}$, we have $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{Enc}_{\mathsf{pk}}(b; r)) = b$ with overwhelming probability over $r \leftarrow \Delta_{\mathcal{R}}$.

Definition 3.1. We say that $\mathcal{E}$ is *samplable* if the following conditions hold:

1. The set $\mathcal{C}$ is finite, and the distribution on $\mathcal{C}$ given by
$$(\mathsf{Enc}_{\mathsf{pk}}(b; r) : b \overset{\mathrm{R}}{\leftarrow} \{0,1\}, r \leftarrow \Delta_{\mathcal{R}}) \tag{3.1}$$
is statistically indistinguishable from the uniform distribution on $\mathcal{C}$.
2. There is an efficient algorithm $\mathsf{SampleRand}$ that takes as input a secret key and a ciphertext and outputs a value in $\mathcal{R}$, such that if we choose
$$b \overset{\mathrm{R}}{\leftarrow} \{0,1\},\ r \leftarrow \Delta_{\mathcal{R}},$$
$$c \leftarrow \mathsf{Enc}_{\mathsf{pk}}(b; r),\ \tilde{r} \leftarrow \mathsf{SampleRand}(\mathsf{sk}, c),$$

then the following two distributions are statistically indistinguishable:

$$(\mathsf{sk}, c, r) \approx (\mathsf{sk}, c, \tilde{r}). \tag{3.2}$$

Note that the second condition implies that for all but a negligible fraction of $c \in \mathcal{C}$, we have

$$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}}(c); \mathsf{SampleRand}_{\mathsf{sk}}(c)) = c.$$

We present two examples of samplable encryption schemes in Section 5.

4 A Deniable Encryption Protocol

We are now ready to construct a sender-deniable encryption scheme from any samplable public key bit-encryption scheme. Let $n \in \mathbb{N}$ be a security parameter. At a high level, our protocol consists of the following exchange, iterated n times to ensure correctness of decryption. Since the "sender" and "receiver" are both sending and receiving messages, for clarity we describe our protocol in terms of two players Sarah and Ronald. Sarah has a bit b that she wishes to transmit to Ronald in a deniable manner.

To encrypt the bit b, first Ronald sends $4n + 1$ freshly generated public keys pk_i to Sarah. Sarah partitions the indices $\{0, \ldots, 4n\}$ into three random disjoint subsets and computes $4n + 1$ ciphertexts as follows:

- Choose a set A containing $n + 1$ indices and encrypt b under the key pk_i for $i \in A$.
- Choose a set B disjoint from A containing n indices and encrypt $1 - b$ under the key pk_i for $i \in B$.
- Let C denote the remaining $2n$ indices, and sample a uniformly random ciphertext c_i from $\mathcal{C}$ for $i \in C$. Definition 3.1 guarantees that approximately half of these ciphertexts decrypt to b.

Ronald computes a bit b' by decrypting all of the received ciphertexts and taking the majority of the plaintexts. We will show that the probability that $b' = b$ is at least $1/2 + 1/5\sqrt{n}$; thus repeating the protocol independently n times ensures that Ronald can compute b correctly with overwhelming probability[2].

When coerced to reveal her randomness, Sarah will reveal sets $\tilde{A}, \tilde{B}, \tilde{C}$ in which a real encryption of b from the set A is exchanged with a sampled encryption of $1-b$ from the set C. She must also reveal appropriately distributed randomness for the sampled encryption. However, to do this she needs Ronald's help; in particular, she needs a secret key for an index $i \in C$ such that c_i encrypts $1 - b$. Since Ronald cannot tell which encryptions were sampled randomly, he sends back $n/2$ random pairs of indices (u_i, v_i) such that c_{u_i} and c_{v_i} decrypt to different messages. Almost certainly, he will choose at least one pair (u_j, v_j) such that $u_j \in A$ corresponds to an encryption of b and $v_j \in C$ corresponds to an encryption of $1 - b$, or vice versa. Sarah indicates such a pair, and Ronald sends the corresponding secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$. Knowing the secret key for a randomly generated ciphertext enables Sarah to use the $\mathsf{SampleRand}$ algorithm to produce randomness which is indistinguishable from "real" randomness that would produce the given ciphertext.

We now formally describe our scheme.

[2] More precisely, we can obtain correctness with overwhelming probability by repeating the protocol $f(n)$ times for any $\omega(\sqrt{n \log n})$ function f; we use $f(n) = n$ for simplicity.

4.1 The Protocol

Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{SampleRand})$ be a samplable public key bit encryption scheme. Define a protocol $\pi_{\mathcal{E}}$ between a sender Sarah and a receiver Ronald as follows:

1. Ronald's input is a security parameter $n \in \mathbb{N}$; we assume for simplicity that n is even.
   (a) Choose key pairs $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{KeyGen}(1^n)$ for $i = 0, \ldots, 4n$.
   (b) Send $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{4n})$ to Sarah.
2. Sarah's input is a bit $b \in \{0, 1\}$. Sarah computes $n + 1$ encryptions of b and n encryptions of $1 - b$ under different keys, and chooses $2n$ additional random ciphertexts:
   (a) Choose a random partition of $\{0, \ldots, 4n\}$ into disjoint subsets A, B, C of cardinality $n + 1$, n, and $2n$, respectively.
   (b) For $i \in A$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i = b$, and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.
   (c) For $i \in B$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i = 1 - b$, and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.
   (d) For $i \in C$, choose random ciphertexts $c_i \xleftarrow{\mathrm{R}} \mathcal{C}$.
   (e) Send $(c_0, \ldots, c_{4n})$ to Ronald.
3. Ronald decrypts all ciphertexts: he sets $\beta'_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i)$ for $i = 0, \ldots, 4n$, and outputs the majority b'.
4. Ronald sends pairs of indices corresponding to ciphertexts with opposite messages back to Sarah.
   (a) Set $I = \varnothing$. Do the following for $i = 1, \ldots, n/2$.
      i. Choose a random pair of indices $(u_i, v_i) \in \{0, \ldots, 4n\}$ such that $u_i, v_i \notin I$ and $\beta'_{u_i} \neq \beta'_{v_i}$.
      ii. Set $I \leftarrow I \cup \{u_i, v_i\}$.
   (b) Ronald sends $(u_1, v_1), \ldots, (u_{n/2}, v_{n/2})$ to Sarah.
5. Sarah chooses a pair of indices such that one index is in the set A and one is in the set C:
   (a) Choose a random $j \in \{1, \ldots, n/2\}$ such that either $u_j \in A$ and $v_j \in C$ or $u_j \in C$ and $v_j \in A$. If no such index exists, then abort the protocol.
   (b) Send j to Ronald.
6. Ronald sends the secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ to Sarah.

This completes the description of the protocol.

The Transcript. The protocol's transcript consists of:
1. The public keys $(\mathsf{pk}_0, \ldots, \mathsf{pk}_{4n})$ sent by Ronald in Step 1,
2. The ciphertexts $(c_0, \ldots, c_{4n})$ sent by Sarah in Step 2e,
3. The tuples (u_i, v_i) sent by Ronald in Step 4,
4. The index j Sarah has chosen in Step 5 (or the decision to abort), and
5. The secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ sent by Ronald in step 6.

The Faking Algorithm. The algorithm Fake is defined as follows: suppose we are given a message b, randomness

$$r_S = \left(A, B, C, (\alpha_i)_{i\in A\cup B}, (c_i)_{i\in C}\right) \tag{4.1}$$

and a transcript tr. If tr indicates that the protocol has aborted in Step 5a, then Fake outputs $\tilde{r}_S = \perp$. Otherwise, do the following:

1. Let $y = \{u_j, v_j\} \cap A$ and $z = \{u_j, v_j\} \cap C$.
   (In particular, we have $\beta'_y = b$ and $\beta'_z = 1 - b$.)
2. Compute
   - $\tilde{A} \leftarrow B \cup \{z\}, \quad \tilde{B} \leftarrow A \setminus \{y\}, \quad \tilde{C} \leftarrow (C \setminus \{z\}) \cup \{y\}$.
   - $\tilde{\alpha}_i \leftarrow \alpha_i$ for $i \in (A \setminus \{y\}) \cup B$.
   - $\tilde{\alpha}_z \leftarrow \mathsf{SampleRand}_{\mathsf{sk}_z}(c_z)$.
   - $\tilde{c}_i \leftarrow c_i$ for $i \in \tilde{C}$.
3. Output
$$\tilde{r}_S = \left(\tilde{A}, \tilde{B}, \tilde{C}, (\tilde{\alpha}_i)_{i\in\tilde{A}\cup\tilde{B}}, (\tilde{c}_i)_{i\in\tilde{C}}\right). \tag{4.2}$$

We denote the n-fold independent execution of the encryption protocol by $\pi^n_{\mathcal{E}}$, where the final output is determined as the majority of the output bits of the individual instances.

It is clear that the protocol runs in time polynomial in n. We now show that the protocol almost never aborts in Step 5a.

Proposition 4.1. *The probability that Sarah aborts the protocol $\pi_{\mathcal{E}}$ in Step 5a is negligible in n.*

Proof. For $\beta \in \{0,1\}$, let $C_\beta := C \cap \{i : \mathsf{Dec}_{\mathsf{sk}_i}(c_i) = \beta\}$. By Property 1 of Definition 3.1 and Chernoff bounds [14, Ch. 8, Prop. 5.3] we have, with overwhelming probability, $\frac{7n}{8} \le |C_\beta| \le \frac{9n}{8}$ for $\beta \in \{0,1\}$. Now in each iteration of Step 4(a)i, the probability of choosing a pair (u_i, v_i) with either $u_i \in A$ and $v_i \in C$ or $v_i \in A$ and $u_i \in C$ is at least

$$\frac{|A| - n/2}{|A| + |C_b|} \cdot \frac{|C_{1-b}| - n/2}{|C_{1-b}| + |B|} \ge \frac{n/2}{18n/8} \cdot \frac{3n/8}{17n/8} = \frac{2}{51}$$

whenever $n \ge 8$. Since $n/2$ pairs are chosen in total, by another application of Chernoff bounds we obtain that the probability that no suitable pair is chosen is negligible in n. □

4.2 Correctness

Lemma 4.2. *Let $\mathcal{E}$ be a samplable public key bit encryption scheme, and let b' be Ronald's output computed by $\pi_{\mathcal{E}}((b, r_S), (n, r_R))$. Then the probability (over r_S and r_R) that $b' = b$ is at least $1/2 + 1/(5\sqrt{n})$.*

Proof. A single instance of $\pi_{\mathcal{E}}$ outputs the majority of messages obtained from decrypting all $4n + 1$ ciphertexts c_i. Out of these, c_i encrypts b for $i \in A$ and c_i encrypts $1 - b$ for $i \in B$. This means that $b' = b$ if and only if least half of the remaining $2n$ ciphertexts $\{c_i : i \in C\}$ decrypt to b.

Property 1 of Definition 3.1 implies that a single c_i decrypts to b with probability $p(n)$ satisfying $|1/2 - p(n)| = \nu(n)$ for some negligible $\nu(n)$. The probability that at least half of the ciphertexts $\{c_i : i \in C\}$ decrypt to b is thus

$$\begin{aligned}
\sum_{i=n}^{2n} \binom{2n}{i} p(n)^i \cdot (1-p(n))^{2n-i} &\geq \sum_{i=n}^{2n} \binom{2n}{i} \left(\frac{1}{2} - \nu(n)\right)^{2n} \\
&= \left(\frac{1}{2} - \nu(n)\right)^{2n} \cdot \frac{1}{2}\left(\binom{2n}{n} + \sum_{i=0}^{2n} \binom{2n}{i}\right) \\
&\overset{(*)}{\geq} (1 - 2\nu(n))^{2n} \cdot \left(\frac{1}{4\sqrt{n}} + \frac{1}{2}\right) \\
&\geq 1/2 + \frac{1}{5\sqrt{n}},
\end{aligned}$$

where $(*)$ follows from the inequality $\binom{2n}{n} \geq 2^{2n-1}/\sqrt{n}$, which in turn follows from Stirling's approximation $n! \sim \sqrt{2\pi n} \cdot e^{-n} \cdot n^n$ [16, p. 13]. □

4.3 Deniability

Our main result is the following:

Theorem 4.3. *Let $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public key encryption scheme. If $\mathcal{E}$ is semantically secure and samplable, then $\pi_{\mathcal{E}}^n$ is a sender-deniable encryption scheme.*

Proof. We must show that the n-fold repetition of $\pi_{\mathcal{E}}$ satisfies the three conditions of Definition 2.1 for negligible $\mu(n)$. Lemma 4.2 shows that the probability that Ronald's message b' is equal to Sarah's message b is at least $1/2 + 1/(5\sqrt{n})$. Thus repeating the protocol n times and taking the majority of the b' gives the correct answer b with overwhelming probability by using Chernoff bounds [14, Ch. 8, Prop. 5.3].

Next, Lemma 2.2 shows that passive secrecy follows from deniability. It thus only remains to prove deniability. We show that if $\mathcal{E}$ is semantically secure and samplable, then for a *single* execution of $\pi_{\mathcal{E}}$, the two distributions of (2.3) are $\mu(n)$-computationally indistinguishable for some negligible $\mu(n)$. Assuming this is the case, a standard hybrid argument shows that the corresponding distributions for the n-fold parallel repetition of $\pi_{\mathcal{E}}$ are $\mu'(n)$-computationally indistinguishable for some negligible $\mu'(n)$.

We now consider a single execution of $\pi_{\mathcal{E}}$ and define a series of games. Game_0 will output the distribution of the left hand side of (2.3), while Game_8 will output the distribution of the right hand side of (2.3). We will then show that for all i, the outputs of Game_i and Game_{i+1} are $\mu_i(n)$-computationally indistinguishable for some negligible $\mu_i(n)$. By the triangle inequality, this implies that the two distributions in (2.3) are $\mu(n)$-computationally indistinguishable for some negligible $\mu(n)$.

We now define our series of games. In each game Ronald's randomness r_R and the security parameter n are taken to be the same. Unless otherwise stated, the output of Game_i is the same as that of Game_{i-1}.

Game_0: The two parties run the protocol $\pi_{\mathcal{E}}$ as defined in Section 4.1, with Sarah's input message b and randomness r_S given by (4.1). The game outputs the message b, the randomness r_S, and the transcript tr.

Game_1: The two parties run the protocol as in Game_0, but change Step 2d as follows:

(2d)' For $i \in C$, choose random $\beta_i \xleftarrow{R} \{0,1\}$ and $\alpha_i \leftarrow \Delta_{\mathcal{R}}$ and compute $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$.

The ciphertexts in tr are still the ciphertexts $c_0, \ldots, c_{4n}$.

Game_2: The two parties run the protocol as in Game_1, but change Steps 2e and 3 as follows:

(2e)' Send $(\beta_0, \ldots, \beta_{4n})$ to Ronald.

(3)' Ronald sets $\beta'_i \leftarrow \beta_i$ for all i and outputs the majority b' of the β'_i.

The ciphertexts $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\beta_i; \alpha_i)$ are now computed at the time the transcript is output. The output is the same as in Game_1.

Game_3: The two parties run the protocol as in Game_2, with an additional step that flips the bit of one ciphertext:

7. (a) Choose a random $x \in C$, not equal to any of the indices sent in Step 4, with $\beta_x = b$.
   (b) Compute $\tilde{c}_x \leftarrow \mathsf{Enc}_{\mathsf{pk}_x}(1 - \beta_x; \alpha_x)$.

The output is the same as in Game_2, except the ciphertext $\tilde{c}_x$ is output instead of c_x.

Game_4: The two parties run the protocol as in Game_3, but now Sarah flips all of the bits that she sends to Ronald (including the bit in ciphertext c_x), while still computing the ciphertexts in the transcript from the original bits (with only β_x flipped). For easier readability we restate most of the protocol.

1. Ronald sends $4n + 1$ key pairs $(\mathsf{pk}_i, \mathsf{sk}_i)$ to Sarah.
2. Sarah does the following:
   (a) Choose a random partition of $\{0, \ldots, 4n\}$ into disjoint subsets A, B, C of cardinality $n + 1$, n, and $2n$, respectively.

   (2b)' For $i \in A$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i \leftarrow b$ and $\tilde{\beta}_i \leftarrow 1 - b$.

   (2c)' For $i \in B$, choose encryption randomness $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\beta_i \leftarrow 1 - b$ and $\tilde{\beta}_i = b$.

   (2d)" For $i \in C$ choose random $\beta_i \xleftarrow{R} \{0,1\}$ and $\alpha_i \leftarrow \Delta_{\mathcal{R}}$, set $\tilde{\beta}_i \leftarrow 1 - \beta_i$.

   (2e)" Send $(\tilde{\beta}_0, \ldots, \tilde{\beta}_{4n})$ to Ronald.

3." Ronald sets $\beta'_i = \tilde{\beta}_i$ and outputs the majority b'.
4. Ronald sends pairs of indices corresponding to ciphertexts with opposite messages back to Sarah. (This step is identical to the corresponding step in the real protocol.)
5. Sarah chooses a pair of indices such that one index is in the set A and one is in the set C. (This step is identical to the corresponding step in the real protocol.)
6. Ronald sends the secret keys $\mathsf{sk}_{u_j}, \mathsf{sk}_{v_j}$ to Sarah.
7. (a) Choose a random $x \in C$, not equal to any of the indices sent in Step 4, with $\beta_x = b$.
   (b) Let $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\beta_i; \alpha_i)$ for all $i \neq x$, and let $\tilde{c}_x \leftarrow \mathsf{Enc}_{\mathsf{pk}_x}(1 - \beta_x; \alpha_x)$.

The output is the same as in Game_3.

Game_5: The two parties run the protocol as in Game_4, but now Sarah uses the flipped bits $\tilde{\beta}_i$ to compute all ciphertexts (including the ciphertext c_x).

7. Set $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}_i}(\tilde{\beta}_i; \alpha_i)$ for all i.
8. The game outputs the protocol's transcript and computes fake randomness:
   - $\tilde{A} \leftarrow B \cup \{z\}, \quad \tilde{B} \leftarrow A \setminus \{y\}, \quad \tilde{C} \leftarrow (C \setminus \{z\}) \cup \{y\}.$
   - $\tilde{\alpha}_i \leftarrow \alpha_i$ for $i \in \tilde{A} \cup \tilde{B}$.
   - $\tilde{c}_i = c_i$ for $i \in \tilde{C}$.

   We define $\tilde{r}_S$ as in (4.2), and the game outputs $\tilde{r}_S$ instead of r_S.

Game_6: The two parties run the protocol as in Game_5, but Sarah fakes the randomness for the ciphertext with index z by choosing $\tilde{\alpha}_z \leftarrow \mathsf{SampleRand}_{\mathsf{sk}_z}(c_z)$. (Recall $z = \{u_j, v_j\} \cap C$.) This $\tilde{\alpha}_z$ is output in the appropriate position of $\tilde{r}_S$. Other than this change, the output is the same as in Game_5.

Game_7: We reverse the change made between Game_1 and Game_2, by setting

(2b–2d) Set $c_i \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\tilde{\beta}_i; \alpha_i)$ for all i.
(2e) Send $(c_0, \ldots, c_{4n})$ to Ronald.
(3) Set $\beta'_i \leftarrow \mathsf{Dec}_{\mathsf{sk}_i}(c_i)$ for $i = 0, \ldots, 4n$, and output the majority b'.

Game_8: We reverse the change made between Game_0 and Game_1, by setting

(2d) For $i \in C$, choose random ciphertexts $c_i \xleftarrow{\mathrm{R}} \mathcal{C}$.

Therefore the output of Game_8 is the message b, the transcript $\widetilde{tr}$ of a real encryption of $1 - b$ using Sarah's randomness r_S, and the fake randomness $\tilde{r}_S = \mathsf{Fake}(b, r_S, \widetilde{tr})$.

Since the output of Game_0 is distributed as the left hand side of (2.3) and the output of Game_8 is distributed as the right hand side of (2.3), it suffices to show that for each i in $1, \ldots, 8$, if there is an efficient adversary that can distinguish the output of Game_i from that of Game_{i-1}, then this adversary can be used to solve some problem that we have assumed to be (computationally or statistically) infeasible. We now consider each pair of games in turn.

$\mathsf{Game}_0 \rightarrow \mathsf{Game}_1$: The fact that $\mathcal{E}$ is samplable implies that the outputs of these two games are statistically indistinguishable. To show this, we put an ordering on C and define hybrid games for $j = 0, \ldots, 2n$ in which the first j ciphertexts c_i for $i \in C$ are chosen at random from C, and the last $2n - j$ are chosen as real encryptions of random bits. The 0th hybrid is Game_0 and the $2n$th hybrid is Game_1.

Suppose we are given a ciphertext X chosen as either a real encryption of a random bit or as a uniformly random ciphertext. Let x be the index of the ciphertext that changes between the jth and $(j+1)$th hybrid. We can simulate the protocol by choosing secret keys sk_i for all $i \neq x$ and using X as c_x. When X is a random ciphertext for pk_x we are in the jth hybrid, and when X is a real encryption of a random bit under pk_x we are in the $(j+1)$th. Thus any adversary that can distinguish these two hybrids can distinguish a random ciphertext from a real encryption of a random bit, which contradicts the assumption that $\mathcal{E}$ is samplable.

$\mathsf{Game}_1 \rightarrow \mathsf{Game}_2$: Note that the output of Ronald in Step 4, as well as the remainder of the protocol, depends only on the *plaintexts* of the messages he receives in Step 2e. Since Sarah now knows all of the plaintexts, she can send these plaintexts instead in Step 2e and compute the c_i for the transcript later. Thus the output distributions of Game_1 and Game_2 are identical.

$\mathsf{Game}_2 \rightarrow \mathsf{Game}_3$: The fact that $\mathcal{E}$ is semantically secure implies that the outputs of these two games are $\nu(n)$-computationally indistinguishable for some negligible $\nu(n)$. To show this, first note that the indices Ronald sends in Step 4 are now all determined before Sarah computes any ciphertexts. Thus we can choose a random $x \in C$ not equal to *any* of these indices (assuming such an x exists) without using any public or secret keys.

Now let X be a semantic security challenge for pk_x. We can simulate the protocol by choosing secret keys sk_i for all $i \neq x$ and setting $c_x = X$. When X is an encryption of b we are in Game_2, and when X is an encryption of $1-b$ we are in Game_3. Thus any adversary that can distinguish the outputs of Game_2 and Game_3 can be used to break the semantic security of $\mathcal{E}$.

Finally, we show that an index x as above exists with overwhelming probability. Let C_β be defined as in the proof of Proposition 4.1, and recall that, with overwhelming probability, we have $\frac{7n}{8} \leq |C_\beta| \leq \frac{9n}{8}$. Since Ronald chooses $n/2$ indices in Step 4 that correspond to encryptions of b, with overwhelming probability there remain at least $3n/8$ indices in C_b from which to choose x.

$\mathsf{Game}_3 \rightarrow \mathsf{Game}_4$: First, the distribution of the β_i's does not change, and consequently the distribution of the c_i's does not change. Second, note that choosing the process of choosing the pairs (u_i, v_i) in Step 4 is identical in Game_3 and Game_4: since Ronald chooses pairs of indices with opposite plaintexts, flipping all of Ronald's bits does not affect these choices. Furthermore, since Sarah's computation of j in Step 5 depends only on the location of the indices u_i, v_i in the sets A, B, C, Sarah's computation of j using flipped bits is exactly the same as her computation of j in Game_3. Thus the output distributions of Game_3 and Game_4 are identical.

$\mathsf{Game}_4 \rightarrow \mathsf{Game}_5$: Let σ be a permutation of order 2 acting on the set $\{0, \ldots, 4n\}$, such that

$$\sigma(y) = z, \quad \sigma(x) = x, \quad \sigma(A \setminus \{y\}) = B.$$

Note that in particular, this means $\sigma(A) = \tilde{A}$, $\sigma(B) = \tilde{B}$, and $\sigma(C) = \tilde{C}$.

For all $i \in A$, we have $\beta_i = b$ and $\beta_{\sigma(i)} = 1 - b$, and for all $i \in B \cup \{z\}$ we have $\beta_i = 1 - b$ and $\beta_{\sigma(i)} = b$. The index x only appears in the output as the ciphertext $\tilde{c}_x$, so the output of Game_4 is distributed as if $\beta_x = 1 - b$; furthermore, we have $\tilde{\beta}_x = 1 - b$ in Game_5.

Next, observe that the random bits β_i for $i \in C$ are distributed such that inverting all of them does not change their distribution. Because $\beta_x = b$ and $\beta_z = 1 - b$, the distribution on the β_i for $i \in C \setminus \{x, z\}$ is also symmetric, so permuting just these indices does not change the distribution of the corresponding β_i; the same is true for flipping just these bits. We thus conclude that the following two distributions are identical:

- Run Game_4 to encrypt bit b with randomness r_S, producing transcript tr, and output $\sigma(tr)$ and $\sigma(r_s)$. (That is, we apply the permutation σ to all of the indices of the computed elements.)
- Run Game_5 to encrypt bit $1-b$ with randomness r_S, producing transcript $\widetilde{tr}$, and output $\widetilde{tr}$ and $\tilde{r}_S$.

Since the sets A, B, C are uniformly random disjoint subsets of $\{0, \ldots, 4n\}$ of cardinality $n+1$, n, and $2n$, respectively, applying a permutation to the indices cannot change the output distribution of Game_4. It follows that the output distributions of Game_4 and Game_5 are identical.

$\mathsf{Game}_5 \rightarrow \mathsf{Game}_6$: The fact that $\mathcal{E}$ is samplable implies that these two distributions are statistically indistinguishable.

Suppose we are given a secret key sk^*, a ciphertext X, and randomness R from one of the two distributions of (3.2). We can simulate the protocol by using sk^* as sk_z, X as c_z, and R as $\tilde{\alpha}_z$. When R is chosen from $\Delta_{\mathcal{R}}$ we are in Game_5, and when R is computed using $\mathsf{SampleRand}$ we are in Game_6. Thus any adversary that can distinguish these two games can distinguish the two distributions of (3.2), which contradicts the assumption that $\mathcal{E}$ is samplable.

$\mathsf{Game}_6 \rightarrow \mathsf{Game}_7$: By the same argument as above for $\mathsf{Game}_1 \rightarrow \mathsf{Game}_2$, the outputs of these two games are identical.

$\mathsf{Game}_7 \rightarrow \mathsf{Game}_8$: By the same argument as above for $\mathsf{Game}_0 \rightarrow \mathsf{Game}_1$, the fact that $\mathcal{E}$ is samplable implies that the outputs of these two games are statistically indistinguishable. □

5 Instantiations

5.1 Quadratic Residuosity

Our first example of a samplable encryption system is the Goldwasser-Micali bit encryption system [10], which is secure under the quadratic residuosity assumption.

For a positive integer N that is a product of two distinct odd primes, we define the set $\mathcal{J}(N) = \{x \in \mathbb{Z}_N^* : \left(\frac{x}{N}\right) = 1\}$ and let $\mathcal{Q}(N)$ be the subgroup of squares in $\mathbb{Z}_N^*$, which has index 2. The *quadratic residuosity assumption* states that when N is the product of two randomly chosen n-bit primes, the two distributions obtained by sampling uniformly at random from $\mathcal{Q}(N)$ and from $\mathcal{J}(N) \setminus \mathcal{Q}(N)$ are $\mu(n)$-computationally indistinguishable for negligible $\mu(n)$.

Construction 5.1.

- $\mathsf{KeyGen}(n)$: Compute $N = pq$, where p, q are n-bit primes. Choose a quadratic non-residue $g \in \mathcal{J}(N) \setminus \mathcal{Q}(N)$. The public key is $\mathsf{pk} = (N, g)$ and the secret key is $\mathsf{sk} = p$.
- $\mathsf{Enc}_{\mathsf{pk}}(b; r)$: Choose $r \xleftarrow{\mathrm{R}} \mathbb{Z}_N^*$ and output $c = g^b r^2 \pmod N$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Let $\mathsf{sk} = p$. If $\left(\frac{c}{p}\right) = 1$, output 0; otherwise output 1.
- $\mathsf{SampleRand}_{\mathsf{sk}}(c)$: If $\mathsf{Dec}_{\mathsf{sk}}(c) = 0$, output a random solution to $X^2 = c \pmod N$; otherwise output a random solution to $X^2 = c/g \pmod N$.

It is a standard result [10] that the encryption scheme is semantically secure under the quadratic residuosity assumption. We now show the samplable properties.

Proposition 5.2. *The public key encryption scheme of Construction 5.1 is samplable.*

Proof. In the notation of Definition 3.1, the set $\mathcal{R}$ is $\mathbb{Z}_N^*$, and the set $\mathcal{C}$ is $\mathcal{J}(N)$. The distribution $\Delta_{\mathcal{R}}$ is the uniform distribution on $\mathbb{Z}_N^*$. Since $\mathcal{J}(N) = (\mathbb{Z}_N^*)^2 \cup g \cdot (\mathbb{Z}_N^*)^2$, it follows that the distribution (3.1) is the uniform distribution on $\mathcal{J}(N)$.

For the second condition, we observe that for a given $c \in \mathcal{J}(N)$ the value $x = \mathsf{SampleRand}_{\mathsf{sk}}(c)$ is distributed uniformly amongst the four possible values of r such that $c = \mathsf{Enc}_{\mathsf{pk}}(\mathsf{Dec}_{\mathsf{sk}}(c), r)$. Since real randomness comes from the uniform distribution on $\mathbb{Z}_N^*$, it follows that the two distributions (3.2) are identical. □

5.2 Trapdoor Permutations

Our second samplable encryption system is the bit encryption scheme built from a generic trapdoor permutation (cf. [11, Construction 10.27]):

Construction 5.3.

- $\mathsf{KeyGen}(n)$: Let $f : \mathcal{R} \to \mathcal{R}$ be sampled from a family $\mathcal{F}$ of trapdoor permutations (using n as the security parameter) and let $g = f^{-1}$. Let $H : \mathcal{R} \to \{0,1\}$ be a hard-core predicate for f. The public key is $\mathsf{pk} = (f, H)$ and the secret key is $\mathsf{sk} = g$.
- $\mathsf{Enc}_{\mathsf{pk}}(b; r)$: Choose $r \xleftarrow{R} \mathcal{R}$ and output $c = (f(r), H(r) \oplus b) \in \mathcal{R} \times \{0,1\}$.
- $\mathsf{Dec}_{\mathsf{sk}}(c)$: Write $c = (y, z) \in \mathcal{R} \times \{0,1\}$. Output $H(g(y)) \oplus z$.
- $\mathsf{SampleRand}_{\mathsf{sk}}(c)$: Write $c = (y, z)$ and output $g(y)$.

It is a standard result (see e.g. [11, Theorem 10.28]) that if $\mathcal{F}$ is a family of trapdoor permutations, then the encryption scheme is semantically secure. In particular, under the RSA assumption we can let f be the function $x \mapsto x^e \pmod N$ with the hard-core predicate $H(x) = \mathsf{lsb}(x)$ [1].

Proposition 5.4. *The public key encryption scheme of Construction 5.3 is samplable.*

Proof. Since r is sampled uniformly from $\mathcal{R}$ and f is a permutation, the first component of the ciphertext is uniformly distributed in $\mathcal{R}$. Furthermore, if b is a uniformly random bit, the second component of the ciphertext is random and independent of the first component. Thus the distribution (3.1) is the uniform distribution on $\mathcal{R} \times \{0,1\}$.

For the second condition, since f is a permutation the map from $\mathcal{R} \times \{0,1\}$ to itself given by $(r, b) \mapsto \mathsf{Enc}_{\mathsf{pk}}(b; r)$ is a bijection. Thus there is a unique value of the randomness r for any ciphertext (which is exactly what is recovered by $\mathsf{SampleRand}$), and the two distributions (3.2) are identical. □

6 Conclusion and Open Problems

We have presented a sender-deniable public key encryption scheme that has a single (interactive) protocol for encryption and has negligible detection probability. It is the

first construction that satisfies these strict requirements. The security of our construction is based on well established assumptions; we give one construction based on the hardness of deciding quadratic residuosity and one based on the existence of trapdoor permutations.

Receiver-deniable encryption can be obtained from sender-deniable encryption by a straightforward construction [3]; basically, the roles of sender and receiver are reversed, and the receiver encrypts a bit that is used by the sender as a one-time pad. It is an open problem to construct a bi-deniable encryption scheme with a single encryption algorithm and no trusted third party. (The recent work of O'Neill, Peikert, and Waters [12] achieves this goal for a weaker notion of deniability that allows two encryption algorithms.) Another open problem arising from our work is to remove the interaction from our encryption protocol.

Both instantiations of our scheme rely on the hardness of factoring. It is an open problem to construct deniable encryption schemes from other assumptions. A promising direction for this problem is cryptosystems based on the hardness of Learning With Errors (LWE), a lattice-related problem. In particular, the variant of Regev's LWE cryptosystem [13] presented by Gentry, Peikert, and Vaikuntanathan [9, §8] has the property that a trapdoor can be embedded in the secret key that allows the key holder to efficiently sample from the distribution of randomness used in the encryption. However, the scheme does not have a dense ciphertext space, so the scheme is not samplable according to our definition. So far, all methods we have looked at to modify the cryptosystem to overcome this difficulty either induce a non-uniform distribution on real ciphertexts, introduce non-negligible decryption error, or destroy the system's security.

Finally, the overhead of our construction is significant: for a security parameter n, one execution of our protocol requires transmission of $O(n)$ secret keys and ciphertexts, each of length n (not to mention the computational cost of generating $O(n)$ keys and ciphertexts); furthermore, to ensure correctness we must repeat the protocol n times. Several straightforward improvements are possible, since we optimized our presentation for clarity rather than efficiency. However, a significantly more efficient construction seems to require substantial new ideas.

Acknowledgments

The authors thank Dan Boneh, Chris Peikert, Brent Waters, and the anonymous referees for helpful discussions and/or feedback on earlier versions of this work.

References

1. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.P.: RSA and Rabin functions: Certain parts are as hard as the whole. SIAM J. Comput. 17(2), 194–209 (1988)
2. Beaver, D.: Plug and play encryption. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 75–89. Springer, Heidelberg (1997)
3. 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)
4. Canetti, R., Feige, U., Goldreich, O., Naor, M.: Adaptively secure multi-party computation. In: STOC. pp. 639–648 (1996)

5. Chapman, M., Davida, G.: Plausible deniability using automated linguistic stegonagraphy. In: Davida, G.I., Frankel, Y., Rees, O. (eds.) InfraSec 2002. LNCS, vol. 2437, pp. 276–287. Springer, Heidelberg (2002)
6. Choi, S., Dachman-Soled, D., Malkin, T., Wee, H.: Improved non-committing encryption with applications to adaptively secure protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 287–302. Springer, Heidelberg (2009)
7. Czeskis, A., Hilaire, D.J.S., Koscher, K., Gribble, S.D., Kohno, T., Schneier, B.: Defeating encrypted and deniable file systems: TrueCrypt v5.1a and the case of the tattling OS and applications. In: 3rd Usenix Workshop on Hot Topics in Security (2008)
8. Damgård, I., Nielsen, J.: Improved non-committing encryption schemes based on a general complexity assumption. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 432–450. Springer, Heidelberg (2000)
9. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) STOC, pp. 197–206. ACM, New York (2008)
10. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences 28(2), 270–299 (1984); preliminary version in 14th Annual ACM Symposium on Theory of Computing (STOC)
11. Katz, J., Lindell, Y.: Introduction to modern cryptography. Chapman & Hall/CRC, Boca Raton (2008)
12. O'Neill, A., Peikert, C., Waters, B.: Bi-deniable public-key encryption (2010) (manuscript)
13. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: STOC 2005: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM, New York (2005)
14. Ross, S.: A First Course in Probability, 5th edn. Prentice-Hall, Englewood Cliffs (1998)
15. Sako, K., Kilian, J.: Receipt-free mix-type voting scheme: A practical solution to the implementation of a voting booth. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
16. Spitzer, F.: Principles of Random Walks, 2nd edn. Graduate Texts in Mathematics, vol. 34. Springer, Heidelberg (1976)
17. TrueCrypt: Free open-source disk encryption software, http://www.truecrypt.org/

Author Index