

本资源来自数缘社区

<http://maths.utime.cn:81>



欢迎来到数缘社区。本社区是一个**高等数学及密码学**的技术性论坛，由山东大学数学学院研究生创办。在这里您可以尽情的遨游数学的海洋。作为站长，我诚挚的邀请您加入，希望大家能一起支持发展我们的论坛，充实每个版块。把您宝贵的资料与大家一起分享！

数学电子书库

每天都有来源于各类网站的与数学相关的新内容供大家浏览和下载，您既可以点击左键弹出网页在线阅读，又可以点右键选择下载。现在书库中**藏书 1000 余本**。如果本站没有您急需的电子书，可以发帖说明，我们有专人负责为您寻找您需要的电子书。

密码学论文库

国内首创信息安全专业的密码学论文库，主要收集欧密会（Eurocrypt）、美密会（Crypto）、亚密会（Asiacrypt）等国内外知名论文。现在论文库中**收藏论文 4000 余篇**（包括论文库版块 700 余篇、论坛顶部菜单“密码学会议论文集”3000 余篇）。如果本站没有您急需的密码学论文，可以发帖说明，我们有专人负责为您寻找您需要的论文。

提示：本站已经收集到 1981—2003 年欧密会、美密会**全部**论文以及 1997 年—2003 年五大会议**全部**论文（欧密会、美密会、亚密会、PKC、FSE）。

数学综合讨论区

论坛管理团队及部分会员来源于山东大学数学学院**七大专业**（基础数学、应用数学、运筹学、控制论、计算数学、统计学、信息安全），在数学方面均为思维活跃、成绩优秀的研究生，相信会给您的数学学习带来很大的帮助。

密码学与网络安全

山东大学数学学院的信息安全专业师资雄厚，前景广阔，具有**密码理论、密码技术与网络安全技术**三个研究方向。有一大批博士、硕士及本科生活跃于本论坛。本版块适合从事密码学或网络安全方面学习研究的朋友访问。

网络公式编辑器

数缘社区公式编辑器采用 Latex 语言，**适用于任何支持图片格式的论坛或网页**。在本论坛编辑好公式后，您可以将自动生成的公式图片的链接直接复制到你要发的帖子里以图片的形式发表。

如果您觉得本站对您的学习和成长有所帮助，请把它**添加到您的收藏夹**。如果您对本论坛有任何的意见或者建议，请来论坛留下您宝贵的意见。

附录 A：本站电子书库藏书目录

<http://maths.utime.cn:81/bbs/dispbbs.asp?boardID=18&ID=2285>

附录 B：版权问题

数缘社区所有电子资源均来自网络，版权归原作者所有，本站**不承担任何版权责任**。

Lecture Notes in Computer Science 2612
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

Marc Joye (Ed.)

Topics in Cryptology – CT-RSA 2003

The Cryptographers' Track at the RSA Conference 2003
San Francisco, CA, USA, April 13-17, 2003
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editor

Marc Joye
Gemplus, Card Security Group
La Vigie, Avenue du Jujubier, ZI Athélia IV
13705 La Ciotat Cedex
France
E-mail: marc.joye@gemplus.com

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

CR Subject Classification (1998): E.3, G.2.1, D.4.6, K.6.5, K.4.4, F.2.1-2, C.2, J.1

ISSN 0302-9743

ISBN 3-540-00847-0 Springer-Verlag Berlin Heidelberg New York

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

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by DA-TeX Gerd Blumenstein
Printed on acid-free paper SPIN: 10872873 06/3142 5 4 3 2 1 0

Preface

These are the proceedings of CT-RSA 2003, the Cryptographers' Track at RSA Conference 2003. The proceedings of CT-RSA 2001 and CT-RSA 2002 were published in Springer-Verlag's Lecture Notes in Computer Science series as LNCS 2020 and LNCS 2271, respectively.

The Cryptographers' Track is one of the many parallel tracks of the RSA Conference. With many thousands of participants, the RSA Conference is the largest security and cryptography event of the year.

There were 97 submitted contributions this year, of which 26, or 27%, were selected for presentation. The program also included two invited talks by Tom Berson ("Cryptography After the Bubble: How to Make an Impact on the World") and by Adi Shamir ("RSA Shortcuts").

All submissions were reviewed by at least three members of the program committee. I am very grateful to the 21 members of the program committee for their hard and efficient work in assembling the program. My thanks also go to the 78 external referees who helped in the review process in their area of expertise: Gail-Joon Ahn, Toru Akishita, Kazumaro Aoki, Gildas Avoine, Joonsang Baek, Olivier Benoit, Alex Biryukov, Alexandra Boldyreva, Antoon Bosselaers, Emmanuel Bresson, Eric Brier, Brice Canvel, Dario Catalano, Chien Yuan Chen, Donghyeon Cheon, Jung Hee Cheon, Olivier Chevassut, Kilsoo Chun, Mathieu Ciet, Christophe Clavier, Jean-Sébastien Coron, Reza Curtmola, Christophe De Cannière, Jean-François Dhem, Xuhua Ding, Pierre-Alain Fouque, Jacques Fournier, Fabien Germain, Jovan Dj. Golić, Philippe Golle, Louis Granboulan, Jorge Guajardo, D.J. Guan, Jinsu Hyun, Eliane Jaulmes, Pascal Junod, Sungwoo Kang, Jonathan Katz, Dongryeol Kim, Tetsutaro Kobayashi, Yoshi Kohno, Takeshi Koshiba, Hyun-jo Kwon, Byoungcheon Lee, Y.C. Lee, Arjen Lenstra, Seongan Lim, Phil MacKenzie, Gwenaëlle Martinet, Jean Monnerat, Maithili Narasimha, Hanae Nozaki, Katsuyuki Okeya, Francis Olivier, Siddika Berna Ors, Elisabeth Oswald, Pascal Paillier, Benny Pinkas, Guillaume Poupart, Pankaj Rohatgi, Ludovic Rousseau, Tomas Sander, Marius Schilder, Jasper Scholten, Stefaan Seys, Hung-Min Sun, Jaechul Sung, Mike Szydlo, Mårten Trolin, Christophe Tymen, Frédéric Valette, Holger Vogt, Bogdan Warinschi, Susanne Wetzel, Karel Wouters, Shouhuai Xu, Jeong Yi, and Fangguo Zhang. I apologize for possible omissions. Finally, I would like to thank Julien Bouchier for hosting and maintaining the review website. The excellent web-based software, maintained by the COSIC group at K.U. Leuven, was used for the review process.

In addition to those mentioned above, many people contributed to the success of CT-RSA 2003, and I thank them: Ari Juels and Burt Kaliski for interfacing with the conference organizers; Marseille Burton and Kurt Stammberger for handling issues not directly related to the scientific program; and Bart Preneel, the Program Chair for CT-RSA 2002, for giving me good advice. Last, but not

least, I would like to thank all the authors who submitted papers, making the conference possible, and the authors of accepted papers for their cooperation.

It is our sincere hope that the Cryptographers' Track will remain a premium forum of intellectual exchange in the broad area of the application and theory of cryptography.

November 2002

Marc Joye

RSA Conference 2003, Cryptographers' Track

April 13–17, 2003, San Francisco, USA

RSA Conference 2003 was organized by RSA Security Inc. and its partner organizations around the world. The Cryptographers' Track at RSA Conference 2003 was organized by RSA Laboratories (<http://www.rsasecurity.com>).

Program Chair

Marc Joye Gemplus, France

Program Committee

Giuseppe Ateniese	The Johns Hopkins University, USA
John Black	University of Colorado at Boulder, USA
Daniel Bleichenbacher	Bell Laboratories, USA
Rosario Gennaro	IBM T.J. Watson Research Center, USA
Stuart Haber	Hewlett-Packard Laboratories, USA
Helena Handschuh	Gemplus, France
Markus Jakobsson	RSA Laboratories, USA
Antoine Joux	DCSSI, France
Kwangjo Kim	Information and Communications University, Korea
Seungjoo Kim	Korea Information Security Agency, Korea
Chi-Sung Laih	National Cheng Kung University, Taiwan
Tatsuaki Okamoto	NTT Labs, Japan
David Pointcheval	Ecole Normale Supérieure, France
Bart Preneel	Katholieke Universiteit Leuven, Belgium
Jean-Jacques Quisquater	Université Catholique de Louvain, Belgium
Tsuyoshi Takagi	Technische Universität Darmstadt, Germany
Gene Tsudik	University of California at Irvine, USA
Serge Vaudenay ...	Swiss Federal Institute of Technology (EPFL), Switzerland
Sung-Ming Yen	National Central University, Taiwan
Moti Yung	Columbia University, USA
Yuliang Zheng	UNC Charlotte, USA

Table of Contents

Key Self-protection

Forward-Security in Private-Key Cryptography	1
<i>Mihir Bellare and Bennet Yee</i>	
Intrusion-Resilient Public-Key Encryption	19
<i>Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung</i>	

Message Authentication

TMAC: Two-Key CBC MAC	33
<i>Kaoru Kurosawa and Tetsu Iwata</i>	
Montgomery Prime Hashing for Message Authentication	50
<i>Douglas L. Whiting and Michael J. Sabin</i>	

Digital Signatures

An Analysis of Proxy Signatures: Is a Secure Channel Necessary?	68
<i>Jung-Yeun Lee, Jung Hee Cheon, and Seungjoo Kim</i>	
Invisibility and Anonymity of Undeniable and Confirming Signatures	80
<i>Steven D. Galbraith and Wenbo Mao</i>	

Pairing Based Cryptography

A Secure Signature Scheme from Bilinear Maps	98
<i>Dan Boneh, Ilya Mironov, and Victor Shoup</i>	
Access Control Using Pairing Based Cryptography	111
<i>Nigel P. Smart</i>	

Multivariate and Lattice Problems

NTRUSIGN: Digital Signatures Using the NTRU Lattice	122
<i>Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte</i>	
About the XL Algorithm over $GF(2)$	141
<i>Nicolas T. Courtois and Jacques Patarin</i>	

Cryptographic Architectures

Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications	158
<i>Guido Bertoni, Jorge Guajardo, Sandeep Kumar, Gerardo Orlando, Christof Paar, and Thomas Wollinger</i>	
Hardware Performance Characterization of Block Cipher Structures	176
<i>Lu Xiao and Howard M. Heys</i>	

New RSA-based Cryptosystems

Simple Identity-Based Cryptography with Mediated RSA	193
<i>Xuhua Ding and Gene Tsudik</i>	
Two Birds One Stone: Signcryption Using RSA	211
<i>John Malone-Lee and Wenbo Mao</i>	

Invited Talk I

Cryptography after the Bubble: How to Make an Impact on the World	226
<i>Tom Berson</i>	

Chosen-Ciphertext Security

Rethinking Chosen-Ciphertext Security under Kerckhoffs' Assumption	227
<i>Seungjoo Kim, Masahiro Mambo, and Yuliang Zheng</i>	
Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes	244
<i>Bodo Möller</i>	

Broadcast Encryption and PRF Sharing

Fault Tolerant and Distributed Broadcast Encryption	263
<i>Paolo D'Arco and Douglas R. Stinson</i>	
Shared Generation of Pseudo-Random Functions with Cumulative Maps ..	281
<i>Huaxiong Wang and Josef Pieprzyk</i>	

Authentication Structures

Authenticated Data Structures for Graph and Geometric Searching	295
<i>Michael T. Goodrich, Roberto Tamassia, Nikos Triandopoulos, and Robert Cohen</i>	
Fractal Merkle Tree Representation and Traversal	314
<i>Markus Jakobsson, Tom Leighton, Silvio Micali, and Michael Szydlo</i>	

Invited Talk II

RSA Shortcuts	327
<i>Adi Shamir</i>	

Elliptic Curves and Pairings

The Width- w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks	328
<i>Katsuyuki Okeya and Tsuyoshi Takagi</i>	
Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation	343
<i>Kirsten Eisenträger, Kristin Lauter, and Peter L. Montgomery</i>	

Threshold Cryptography

Two Efficient and Provably Secure Schemes for Server-Assisted Threshold Signatures	355
<i>Shouhuai Xu and Ravi Sandhu</i>	
Secure Applications of Pedersen's Distributed Key Generation Protocol ...	373
<i>Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin</i>	

Implementation Issues

Seeing through MIST Given a Small Fraction of an RSA Private Key	391
<i>Colin D. Walter</i>	
Simple Backdoors for RSA Key Generation	403
<i>Claude Crépeau and Alain Slakmon</i>	
Author Index	417

Author Index

Bellare, Mihir	1
Berson, Tom	226
Bertoni, Guido	158
Boneh, Dan	98
Cheon, Jung Hee	68
Cohen, Robert	295
Courtois, Nicolas T.	141
Crépeau, Claude	403
D'Arco, Paolo	263
Ding, Xuhua	193
Dodis, Yevgeniy	19
Eisenträger, Kirsten	343
Franklin, Matt	19
Galbraith, Steven D.	80
Gennaro, Rosario	373
Goodrich, Michael T.	295
Guajardo, Jorge	158
Heys, Howard M.	176
Hoffstein, Jeffrey	122
Howgrave-Graham, Nick	122
Iwata, Tetsu	33
Jakobsson, Markus	314
Jarecki, Stanisław	373
Katz, Jonathan	19
Kim, Seungjoo	68, 227
Krawczyk, Hugo	373
Kumar, Sandeep	158
Kurosawa, Kaoru	33
Lauter, Kristin	343
Lee, Jung-Yeun	68
Leighton, Tom	314
Malone-Lee, John	211
Mambo, Masahiro	227
Mao, Wenbo	80, 211
Micali, Silvio	314
Mironov, Ilya	98
Miyaji, Atsuko	19
Möller, Bodo	244
Montgomery, Peter L.	343
Okeya, Katsuyuki	328
Orlando, Gerardo	158
Paar, Christof	158
Patarin, Jacques	141
Pieprzyk, Josef	281
Pipher, Jill	122
Rabin, Tal	373
Sabin, Michael J.	50
Sandhu, Ravi	355
Shamir, Adi	327
Shoup, Victor	98
Silverman, Joseph H.	122
Slakmon, Alain	403
Smart, Nigel P.	111
Stinson, Douglas R.	263
Szydlo, Michael	314
Takagi, Tsuyoshi	328
Tamassia, Roberto	295
Triandopoulos, Nikos	295
Tsudik, Gene	193
Walter, Colin D.	391
Wang, Huaxiong	281
Whiting, Douglas L.	50
Whyte, William	122
Wollinger, Thomas	158
Xiao, Lu	176
Xu, Shouhuai	355
Yee, Bennet	1
Yung, Moti	19
Zheng, Yuliang	227

Forward-Security in Private-Key Cryptography

Mihir Bellare and Bennet Yee

Dept. of Computer Science & Engineering, University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093, USA
{mihir,bsy}@cs.ucsd.edu
<http://www-cse.ucsd.edu/users/{mihir,bsy}>

Abstract. This paper provides a comprehensive treatment of forward-security in the context of shared-key based cryptographic primitives, as a practical means to mitigate the damage caused by key-exposure. We provide definitions of security, practical proven-secure constructions, and applications for the main primitives in this area. We identify forward-secure pseudorandom bit generators as the central primitive, providing several constructions and then showing how forward-secure message authentication schemes and symmetric encryption schemes can be built based on standard schemes for these problems coupled with forward-secure pseudorandom bit generators. We then apply forward-secure message authentication schemes to the problem of maintaining secure access logs in the presence of break-ins.

Keywords: Symmetric cryptography, forward security, pseudorandom bit generators, message authentication, proofs of security, audit logs.

1 Introduction

Today we can have more confidence in the strength of our cryptographic algorithms than in the security of the systems that implement them. Thus, in practice, the greatest threat to the security we may hope to obtain from some cryptographic scheme may simply be that an intruder breaks into our system and steals some information that will compromise the scheme, such as an underlying key.

Once the key is exposed, future uses of it are compromised: the best hope is that the intrusion is detected and we change keys. However key loss carries another danger. Namely, the security of past uses of the key are compromised. For example data might have been encrypted under this key some time back, and the ciphertexts might be in the hands of the adversary. Now that the adversary obtains the decryption key, it can decrypt the old ciphertexts.

The goal of forward security is to protect against this kind of threat. One endeavors to make sure that security of past uses of a key, in whatever context, is not compromised by loss of some underlying information at the present time.

The focus of this paper is forward-security in the symmetric setting, where parties share a key. We provide a comprehensive treatment including security

definitions and practical proven-secure constructions, for the main primitives in this area, namely pseudorandom bit generators, message authentication schemes and symmetric encryption schemes.

Of these, however, we claim the first (namely a forward-secure pseudorandom bit generator) is the most important. Not only is it useful in its own right, but once we have this, the other primitives can be easily and generically built. We thus start with this.

FORWARD SECURITY IN PSEUDORANDOM BIT GENERATION. Random values need to be generated in many cryptographic implementations. For example, session-key exchange protocols may need to generate random session keys. Signature schemes such as DSS require generation of a random value with each signature. Encryption schemes, both in the symmetric and asymmetric settings, are usually probabilistic. In any of these settings, if the random value in question is exposed, even some time after the cryptographic operation using it took place, security is compromised. In particular, data encrypted under the session key will be exposed once the key is exposed. In DSS, exposure of the random value yields the secret signing key to an attacker. In most probabilistic encryption schemes, exposure of the randomness used by the encryption algorithm leads to exposure of the plaintext or at least loss of partial information about the plaintext.

Often, random values are generated from some seed via a pseudorandom bit generator. With the threats of exposure discussed above, it is possible this seed will get exposed over time. At this point, all the past pseudorandom values fall into the hands of the adversary.

To prevent this, we want a generator that is forward secure. This generator will be stateful. At each invocation it produces some output bits as a function of the current state, updates the state, and then deletes the old state. An adversary breaking in at any point in time gets only the current state. The generator is designed so that it is infeasible to recover any previous state or previous output block from the current state. So if the output blocks were used for encryption as above, an adversary breaking in would still be unable to decrypt ciphertexts created earlier.

Our formal notion of forward security for generators, given in Section 2.2, actually requires something stronger: the sequence of output blocks generated prior to the break-in must be indistinguishable from a random, independent sequence of blocks even if the adversary holds the current state of the generator. This is a strengthening of the notion of security for standard (i.e. not forward secure) generators that was given by [13, 26].

Construction 1 is a way to transform any standard (stateless, not forward secure) pseudorandom bit generator into a forward-secure pseudorandom bit generator. The construction is quite simple and cheap. Furthermore we prove in Theorem 1 that if the original generator is secure in the sense of [13, 26] then our constructed one is forward-secure in the strong sense we define. We also suggest, via Construction 2, a design of forward-secure pseudorandom bit generators based on pseudorandom functions (PRFs). An attractive feature of this design illustrated by Theorem 2 is that the PRF need be secure against only

a very small number of queries and yet we can generate many pseudorandom blocks. The PRF can be instantiated via a block cipher or a hash function in practice, as discussed in Section 2.5.

A natural question is whether existing constructs of stateful generators in the literature are forward-secure. One such construct is the alleged-RC4 stream cipher. But we show in Section 2.3 that it is not forward secure because its state update process is easily reversible.

In Section 2.6 we look at number-theoretic constructions like those of [13, 12]. These are not presented as stateful generators, but from the underlying construction one can define an appropriate state and then use the results of the works in question to show that the generators are forward secure. In fact this extends to any generator using the iterated one-way permutation based paradigm of pseudorandom bit generation that was introduced in [13].

These number-theoretic constructions are however slower than the ones discussed above. Our suggested construction of a forward-secure pseudorandom bit generator is the one of Construction 1 instantiated with a block cipher, or the one of Construction 1 instantiated with a standard pseudorandom bit generator.

As indicated above, forward-secure pseudorandom bit generators can be useful as stand-alone tools for generating the random bits needed by other cryptographic primitives, both symmetric and asymmetric. Now we show that they are also important as building blocks in the design of forward-secure primitives for other problems in symmetric cryptography.

FORWARD-SECURE MESSAGE AUTHENTICATION. Exposure of a key being used for message authentication not only compromises future uses of the key but makes data previously authenticated under this untrustworthy, just as for digital signatures [3, 8]. To remedy this, we can use a forward-secure message authentication scheme. Such a scheme is stateful and key-evolving. The operation of the scheme is divided into stages $i = 1, 2, \dots, n$ and in each stage the parties use the current key K_i for creation and verification of authentication tags. At the end of the stage, K_i is updated to K_{i+1} and K_i is deleted. An attacker breaking in gets the current key. The desired security property is that given the current key K_i it is still not possible to forge MACs relative to any of the previous keys K_1, \dots, K_{i-1} .

Section 3.2 provides strong, formal definitions of security capturing this. Construction 4 then shows how to build a forward-secure message authentication scheme given any standard message authentication scheme (in practice this could be any popular one like a CBC-MAC, HMAC [5] or UMAC [11]) and a forward secure pseudorandom bit generator (in practice this could be any of the secure ones mentioned above). One simply applies the forward-secure generator to update the message-authentication key between stages. Theorem 3 proves the forward-security of this construction based on the assumed security of the base primitives.

SECURE AUDIT LOGS. Forward security is relevant in settings like the usage of message authentication for secure audit logs, as discussed in [25]. An attacker breaking into a machine currently can modify log entries relating to the past, erasing for example a record of the attacker's previous (unsuccessful) attempts

at break-in. To prevent this we use a forward-secure message authentication scheme to tag log entries as they are made by the system. Sequence and other information is included to prevent re-ordering and deletion. A description of such a secure audit log system is in Section 4.

The parties can agree on some convention as to how frequent the key updates should be. For example, maybe once every minute. The frequency reflects their feelings about the likelihood of break-ins: if fast, automated break-ins are considered more likely the updates should be more frequent. Our unoptimized implementation can update keys at a rate of once every 100mS without noticeably increasing system load.

FORWARD SECURE SYMMETRIC ENCRYPTION. The technique used to construct a forward-secure message authentication scheme, based on a standard message authentication and a forward-secure pseudorandom bit generator, is quite general. In particular, the same technique can be used to construct a forward-secure symmetric encryption scheme based on a standard symmetric encryption scheme (in practice this could be any common block-cipher mode of operation, such as CBC with random IV) and a forward-secure pseudorandom bit generator (in practice this could be any of the secure ones mentioned above.) Here the adversary breaking in at some point in time gets the current key but still remains incapable of decrypting data encrypted under the key of any previous stage. Definitions, the construction, and provable-security results can be found in the full version of this paper [10]. The construction preserves both security under chosen-plaintext and chosen-ciphertext attack in the sense that, if the given standard symmetric encryption scheme is secure in one of these senses then so is the constructed forward-secure scheme.

EXTENSIONS. The paradigm of using a forward-secure pseudorandom bit generator to evolve the key of some standard symmetric primitive is very general, and can also be used to convert standard primitives to forward-secure versions in the case of other primitives like PRFs or authenticated encryption schemes [9]. We do not detail these extensions in this paper since they are quite simple.

RELATED WORK. Forward security seems to have first received explicit attention in the context of session key exchange protocols [20, 18], where it is now a common requirement. Forward-security for digital signatures was introduced in [3, 8] and has since then received much attention, and forward-security for asymmetric encryption was considered in [23]. Ours seems to be the first systematic treatment of forward-security in the symmetric setting, but it reflects existing work, practice and design. Forward-secure pseudorandom bit generation has been used in [4, 21]. In practice it is common to design pseudorandom bit generators that on each iteration produce a new seed and delete the old one. Our work can be seen as analyzing, justifying and guiding such existing practice.

The threat of key compromise has been addressed by other means. One is via distribution of the key across multiple machines. Specific approaches include threshold cryptography [17] and proactive secret sharing [22]. (In particular a proactive, distributed pseudorandom bit generator has been designed

by [14, 15].) But distribution is costly. It might be a good option for (say) the secret signing key of a certification authority since the latter has the resources to invest in multiple machines. But it is hardly an option for an average user. Forward-security in contrast is possible in a single-machine environment.

Another, less cryptographic mechanism is embodied in systems such as the those incorporating FIPS 140-1 certified modules [24]. These protect against key compromise by the use of physical security and tamper detection to guarantee key erasure. Forward security is a method for providing many of the same security properties via software means.

Besides providing forward-security, Key-evolving constructs such as those used here can enable more cryptographic operations to be securely implemented with a single key [1]. Other notions of security for pseudorandom bit generators have been considered in [16], and it would be fruitful to augment these with forward security.

2 Forward-Secure Pseudorandom Bit Generators

We recall the standard notion of pseudorandom bit generators. We then specify how forward secure generators operate and provide a formal notion of security for them. Then we explore constructions.

2.1 Standard Pseudorandom Bit Generators

A standard pseudorandom bit generator [13, 26] is a function $G: \{0, 1\}^s \rightarrow \{0, 1\}^{b+s}$ that takes input a s -bit seed and returns a string that is longer than the seed by b bits. We adapt the notion of security of [13, 26] to a concrete security setting [7, 6]. Let D be a distinguishing algorithm that given a $b+s$ bit string $x \parallel y$ returns a bit. Consider the following experiments:

$$\begin{array}{ll|ll} \text{Experiment } \mathbf{Exp}_G^{\text{prg-1}}(D) & \text{Experiment } \mathbf{Exp}_G^{\text{prg-0}}(D) \\ y \xleftarrow{s} \{0, 1\}^s; x \parallel y \leftarrow G(y) & x \parallel y \xleftarrow{s} \{0, 1\}^{b+s} \\ g \xleftarrow{s} D(x \parallel y) & g \xleftarrow{s} D(x \parallel y) \\ \text{Return } g & \text{Return } g \end{array}$$

We let

$$\begin{aligned} \mathbf{Adv}_G^{\text{prg}}(D) &= \Pr[\mathbf{Exp}_G^{\text{prg-1}}(D) = 1] - \Pr[\mathbf{Exp}_G^{\text{prg-0}}(D) = 1] \\ \mathbf{Adv}_G^{\text{prg}}(t) &= \max_D \{\mathbf{Adv}_G^{\text{prg}}(D)\}. \end{aligned}$$

The first term is the *prg-advantage* of D in attacking G . The second term is the *prg-advantage* of G , defined as the maximum, over all adversaries D that have time-complexity at most t , of the prg-advantage of D in attacking G . This is the maximum likelihood of the security of the generator being compromised by an attack that is restricted to time t . We adopt the convention that the time-complexity is the total worst-case execution time of the first experiment

above plus the size of the code of the adversary, all measured in some fixed model of computation. (In particular the time-complexity includes the time for computation of G in the experiment as well as the running time of D .) As usual under the concrete security framework [7, 6], there is no formal notion of G being “secure,” but informally it means that G ’s prg-advantage is “small” for “practical” values of t . All formal results will be stated in concrete security terms.

2.2 Forward-Secure Pseudorandom Bit Generators

Unlike the standard pseudorandom bit generators discussed above, a forward-secure one is a stateful object, and thus we begin by discussing stateful generators.

STATEFUL GENERATORS. A *stateful generator* $\text{GEN} = (\text{GEN.key}, \text{GEN.next}, b, n)$ is specified by a pair of algorithms and a pair of positive integers. The (probabilistic) *key generation* algorithm GEN.key takes no inputs and outputs an initial state (also called seed). The (deterministic) *next step* algorithm, given the current state, returns a pair consisting of an output block, which is a b -bit string, and the next state. We can get a sequence $Out_1, Out_2, \dots, Out_n$ of b -bit output blocks by first picking a seed $St_0 \xleftarrow{\$} \text{GEN.key}$ and then iterating $(Out_i, St_i) \leftarrow \text{GEN.next}(St_{i-1})$ for $i = 1, \dots, n$. The integer n is the maximum number of output blocks the generator may be used to produce.

We can imagine the generation process as application controlled. Whenever the application needs another block of pseudorandom bits it makes a request, at which point the generator is run upon the current state to produce the needed bits, and the new state is saved.

We think of St_{i-1} as being the “key” or “seed” at time i . Forward security will require that this key is erased as soon as the next one has been generated, so that someone breaking into the machine gets only the current key.

FORWARD SECURITY. As we have seen above, a standard pseudorandom generator is said to be secure if its output (on a hidden, random seed) is computationally indistinguishable from a random string of the same length [13, 26]. For forward security of a stateful generator, more is required. The adversary may at some point break into the machine where the state is being maintained and obtain the current state. At that point, the adversary can certainly compute the future output of the generator. However, we require that the bits generated in the past still be secure, in the sense of being computationally indistinguishable from random bits. (This implies in particular that it is computationally infeasible to recover the previous state from the current state).

We allow the adversary to choose, dynamically, when it wants to break in, as a function of the output blocks seen so far. Thus, the adversary is first run in a “find” stage where it is fed output blocks, one at a time, until it says it wants to break in, and at that time is returned the current state. Now, in a “guess” stage, it must decide whether the output blocks it had been fed were really outputs of the generator, or were independent random bits. We capture this below by

considering two experiments, the “real” experiment (in which the output blocks come from the generator) and the “random” experiment (in which the output blocks are random strings). Notice that in both cases, the state advances properly with respect the operation of the generator.

We use the following notation for the adversary. $A(\text{find}, Out, h)$ denotes A in the find stage, taking an output block Out and current history h and returning a pair (d, h) where h is an updated history and $d \in \{\text{find}, \text{guess}\}$. This stage continues until $d = \text{guess}$ or all n output blocks have been generated. (In the latter case the adversary is given the final state in the guess stage.)

Experiment $\mathbf{Exp}_{\mathbf{GEN}}^{\text{fsprg-}1}(A)$	Experiment $\mathbf{Exp}_{\mathbf{GEN}}^{\text{fsprg-}0}(A)$
$St_0 \xleftarrow{\$} \mathbf{GEN}.key$	$St_0 \xleftarrow{\$} \mathbf{GEN}.key$
$i \leftarrow 0; h \leftarrow \varepsilon$	$i \leftarrow 0; h \leftarrow \varepsilon$
Repeat	Repeat
$i \leftarrow i + 1$	$i \leftarrow i + 1$
$(Out_i, St_i) \leftarrow \mathbf{GEN}.next(St_{i-1})$	$(Out_i, St_i) \leftarrow \mathbf{GEN}.next(St_{i-1})$
$(d, h) \xleftarrow{\$} A(\text{find}, Out_i, h)$	$Out_i \xleftarrow{\$} \{0, 1\}^b$
Until $(d = \text{guess})$ or $(i = n)$	$(d, h) \xleftarrow{\$} A(\text{find}, Out_i, h)$
$g \xleftarrow{\$} A(\text{guess}, St_i, h)$	Until $(d = \text{guess})$ or $(i = n)$
Return g	$g \xleftarrow{\$} A(\text{guess}, St_i, h)$
	Return g

We let

$$\begin{aligned} \mathbf{Adv}_{\mathbf{GEN}}^{\text{fsprg}}(A) &= \Pr[\mathbf{Exp}_{\mathbf{GEN}}^{\text{fsprg-}1}(A) = 1] - \Pr[\mathbf{Exp}_{\mathbf{GEN}}^{\text{fsprg-}0}(A) = 1] \\ \mathbf{Adv}_{\mathbf{GEN}}^{\text{fsprg}}(t) &= \max_D \{\mathbf{Adv}_{\mathbf{GEN}}^{\text{fsprg}}(A)\}. \end{aligned}$$

The first term is the *fsprg-advantage* of A in attacking \mathbf{GEN} . The second term is the *fsprg-advantage* of \mathbf{GEN} , defined as the maximum, over all adversaries A that have time-complexity at most t , of the fsprg-advantage of A in attacking \mathbf{GEN} . This is the maximum likelihood of the security of the generator being compromised by an attack that is restricted to time t . We adopt the convention that the time-complexity is the total worst-case execution time of the first experiment above plus the size of the code of the adversary, all measured in some fixed model of computation. (In particular the time-complexity includes the time for computations of $\mathbf{GEN}.key$, $\mathbf{GEN}.next$ in the experiment as well as the running time of D .) As usual under the concrete security framework [7, 6], there is no formal notion of \mathbf{GEN} being “secure,” but informally it means that \mathbf{GEN} ’s fsprg-advantage is “small” for “practical” values of t . All formal results will be in stated in concrete security terms.

Remark 1. Let $\mathbf{GEN} = (\mathbf{GEN}.key, \mathbf{GEN}.next, b, n)$ be a stateful generator whose key-generation algorithm produces random strings of length s bits. Then \mathbf{GEN} has a natural associated standard pseudorandom bit generator $G: \{0, 1\}^s \rightarrow \{0, 1\}^{bn}$ defined as follows. For any $St_0 \in \{0, 1\}^s$ and $i = 1, \dots, n$ we let $(Out_i, St_i) \leftarrow \mathbf{GEN}.next(St_{i-1})$, and then let $G(St_0) = Out_1 \parallel \dots \parallel Out_n$. Note

that if GEN is a forward-secure stateful pseudorandom bit generator then G is a secure standard pseudorandom bit generator. In this sense, forward-security implies standard security. \square

2.3 Alleged-RC4 Is Not Forward-Secure

Some stream ciphers like alleged-RC4 (numerous descriptions can be found on the web, for example [2]) are stateful generators. At each invocation, alleged-RC4 uses an existing table and two table indices to return some pseudorandom bits, and then updates its table and indices, so the table and the indices function as the generator state. It is natural to ask whether this stateful generator has the forward security property. It turns out that it does not. We present an attack demonstrating this.

Below we express alleged-RC4 in our stateful generator notation. Here, the state variable St is the tuple (s, x, y) , where s is a 256-element table viewed as a map of 8 bits to 8 bits, and x and y are inputs to this map, each 8 bits long.

Algorithm $\text{ARC4.next}((s, x, y))$ <pre> $x \leftarrow x + 1 \bmod 256$ $y \leftarrow y + s[x] \bmod 256$ Swap $s[x], s[y]$ Return $(s[s[x] + s[y] \bmod 256], (s, x, y))$ </pre>	Algorithm $\text{AntiRC4}((s, x, y))$ <pre> $z \leftarrow s[s[x] + s[y] \bmod 256]$ Swap $s[x], s[y]$ $y \leftarrow y - s[x] \bmod 256$ $x \leftarrow x - 1 \bmod 256$ Return $(z, (s, x, y))$ </pre>
--	--

We note that the state updates are reversible. Above we also describe Anti-RC4 , which, given the current state, will run alleged-RC4 backwards, recovering the previous state of the generator as well as the corresponding alleged-RC4 output. Here z is the output of the previous state, and (s, x, y) in the output of AntiRC4 is the previous state. This shows that alleged-RC4 is not forward secure. Actually it is a much stronger attack than required by our forward security definition, since it recovers the previous states rather than just distinguishing the output of alleged-RC4 from a sequence of random bits.

2.4 A Construction Based on Standard Generators

There are many existing pseudorandom bit generators which stretch a short seed into a longer pseudorandom sequence. These generators are not (necessarily) stateful, let alone forward secure. We show how to build out of such a generator a new, stateful generator, which has the forward security property as long as the original generator was secure in the standard sense.

Construction 1. Let $G: \{0, 1\}^s \rightarrow \{0, 1\}^{b+s}$ be a standard pseudorandom bit generator, and let $n \geq 1$ be an integer. We associate to G, n a stateful generator $\text{GEN} = (\text{GEN.key}, \text{GEN.next}, b, n)$ whose constituent algorithms are as follows:

Algorithm GEN.key <pre> $St_0 \xleftarrow{\\$} \{0, 1\}^s$ Return St_0 </pre>	Algorithm $\text{GEN.next}(St)$ <pre> $r \leftarrow G(St)$ Return $([r]_{1..b}, [r]_{b+1..b+s})$ </pre>
--	---

The state of this stateful generator is a s -bit string. We are denoting by $[r]_{i..j}$ the substring of r consisting of the bits in positions i through j . \square

The following theorem says that this stateful generator is forward secure as long as the given standard generator G met the standard notion of security of pseudorandom generators of Section 2.1. The proof is in the full version of this paper [10].

Theorem 1. *Let $G: \{0,1\}^s \rightarrow \{0,1\}^{b+s}$ be a standard pseudorandom bit generator, let $n \geq 1$ be an integer, and let GEN be the stateful generator associated to G, n by Construction 1. Then*

$$\mathbf{Adv}_{\text{GEN}}^{\text{fsprg}}(t) \leq 2n \cdot \mathbf{Adv}_G^{\text{prg}}(t'),$$

where $t' = t + O(n \cdot (b + s))$. \square

2.5 A Construction Based on PRFs

We specify a construction of a forward-secure pseudorandom bit generator based on a pseudorandom function (PRF). A special case of practical interest is when we instantiate the PRF via a block cipher. We will see that this turns out to be good for cost and security, and is the preferred construction we suggest. Let us first recall some background about PRFs and their security.

PRFs. A PRF is a map $F: \{0,1\}^s \times \{0,1\}^l \rightarrow \{0,1\}^L$, where s is the key-length, l is the input length and L is the output length. For each key $S \in \{0,1\}^s$ we let F_S denote the function $F(S, \cdot)$. Let $\text{Func}[l \rightarrow L]$ denote the set of all functions with domain $\{0,1\}^l$ and range $\{0,1\}^L$. We now recall the measure of the security of F from [7], which in turn is a quantified version of the original notion of [19]. Let D be a distinguishing algorithm, having access to an oracle for a function $f: \{0,1\}^l \rightarrow \{0,1\}^L$ and returning a bit. Let

$$\begin{aligned} \mathbf{Adv}_F^{\text{prf}}(D) &= \Pr[D^f = 1 : S \xleftarrow{\$} \{0,1\}^s ; f \leftarrow F_S] \\ &\quad - \Pr[D^f = 1 : f \xleftarrow{\$} \text{Func}[l \rightarrow L]] \\ \mathbf{Adv}_F^{\text{prf}}(q, t) &= \max_D \{\mathbf{Adv}_F^{\text{prf}}(D)\}. \end{aligned}$$

The maximum is over all adversaries D that have time-complexity (as per the usual convention, this is the total worst-case execution time of the experiment underlying the first term in the difference above, plus the size of the code of D) at most t and make at most q oracle queries.

CONSTRUCTION. A PRF can be easily transformed into a standard pseudorandom bit generator, and we can then apply our previous construction. Here are the details.

Construction 2. Let $F: \{0,1\}^s \times \{0,1\}^\ell \rightarrow \{0,1\}^L$ be a PRF, and let $b, n \geq 1$ be integers such that $\lceil (b + s)/L \rceil \leq 2^\ell$. For any s -bit S we let $G(S)$ denote the first $b + s$ bits of the sequence $F(S, 0) \parallel F(S, 1) \parallel F(S, 2) \parallel \dots$. (Here we are

using S as the key to the PRF, and the integer inputs to the PRF are interpreted as l -bit strings in some natural way.) This defines a standard pseudorandom bit generator G : $\{0,1\}^s \rightarrow \{0,1\}^{b+s}$. The stateful generator associated to F, b, n is defined as the stateful generator associated by Construction 1 to G, n . \square

The above construction of **GEN** is quite efficient, using only $\lceil(b+s)/L\rceil$ applications of F to implement one step of the forward secure generator (which yields one b -bit output block and an updated state).

SECURITY. An attractive feature of the PRF based construction is that the security requirements on the PRF are low in the sense that it need only resist attacks involving a small number of queries, much fewer than the number of output blocks the generator can generate. This is illustrated by the following theorem.

Theorem 2. *Let $F: \{0,1\}^s \times \{0,1\}^l \rightarrow \{0,1\}^L$ be a PRF, let $b, n \geq 1$ be integers with $\lceil(b+s)/L\rceil \leq 2^l$, and let **GEN** be the stateful generator associated to F, b, n as per Construction 2. Then*

$$\mathbf{Adv}_{\text{GEN}}^{\text{fsprg}}(t) \leq 2n \cdot \mathbf{Adv}_F^{\text{prf}}(q, t'),$$

where $q = \lceil(b+s)/L\rceil$ and $t' = t + O(n \cdot (b+s))$. \square

Proof (Theorem 2). Let G be the standard pseudorandom bit generator associated to F, b as per Construction 2. Then

$$\mathbf{Adv}_{\text{GEN}}^{\text{fsprg}}(t) \leq n \cdot \mathbf{Adv}_G^{\text{prg}}(t') \leq n \cdot \mathbf{Adv}_F^{\text{prf}}(q, t')$$

where the first inequality is by Theorem 1 and the second is standard. \square

AES BASED INSTANTIATION. Let $F = \text{AES}$. In that case, $s = l = L = 128$. Let us choose $b = 128$ and let $n \geq 1$ be an integer. Let **GEN** be the stateful generator associated to F, b, n as per Construction 2. Each iteration of **GEN** returns $b = 128$ output bits and requires $\lceil(b+s)/L\rceil = 2$ AES computations under the same key, which is quite cheap.

Regarding security, note that $\mathbf{Adv}_F^{\text{prf}}(2, t')$ can be expected to be very small, on the order of $t'/2^s$. This is because exhaustive key-search is likely to be the best attack when the attacker has access to only two input-output pairs of the cipher under the key being attacked. (Cryptanalytic attacks typically need many more examples, and so do birthday attacks.) Thus, Theorem 2 tells us that the probability of an attacker, having time-complexity t and attacking up to n output blocks, being able to compromise the forward-security of our AES-based stateful generator, is at most around $2n \cdot t/2^s$, meaning we can safely produce up to 2^{64} output blocks.

HASH-FUNCTION BASED INSTANTIATIONS. Let H be the cryptographic hash function SHA-1, and let $F: \{0,1\}^{80} \times \{0,1\}^{160} \rightarrow \{0,1\}^{160}$ be defined by $F_S(x) = H(S \parallel x)$ for all 80-bit S and 160 bit x . We could regard F as a PRF with $s = 80$ and $l = L = 160$. Setting $b = 80$, we could apply Construction 2 to get a forward-secure pseudorandom bit generator **GEN** that outputs 80 pseudorandom bits per stage while using only a single application of the hash function per stage.

Alternatively, and perhaps better for security although more costly, let H be HMAC-SHA-1 [5] and let $F: \{0, 1\}^{160} \times \{0, 1\}^{160} \rightarrow \{0, 1\}^{b+160}$ be defined by letting $F_S(x)$ be the first $b + 160$ bits of the sequence $H_S(1) \parallel H_S(2) \parallel \dots$. We could apply Construction 2 to get a forward-secure pseudorandom bit generator GEN that outputs b pseudorandom bits per stage while using $\lceil 1 + b/160 \rceil$ applications of H per stage.

2.6 Number-Theoretic Constructions

It turns out that existing number-theory based pseudorandom bit generators such as the Blum-Micali [13] and Blum-Blum-Shub [12] generators can be modified so that they are forward secure. More generally, this is true of any generator using the paradigm introduced by [13] in which the seed is an input to an injective one-way function, and the output bits are obtained by iteration of the function, dropping one hard-core bit into the output at each iteration.

To exemplify this let us look more closely at the Blum-Blum-Shub generator. It is based on repeated squaring of an initial value x modulo a composite N . We specify the stateful version below.

Construction 3. Let $n \geq 1$ be an integer. The constituent algorithms of the stateful BBS generator $\text{GEN} = (\text{GEN.key}, \text{GEN.next}, 1, n)$ are as follows:

Algorithm GEN.key	Algorithm $\text{GEN.next}((N, x))$
Pick primes $p, q \equiv 3 \pmod{4}$	Return $(\text{Parity}(x), (N, x^2 \bmod N))$
$N \leftarrow pq; x \xleftarrow{\$} \mathbb{Z}_N^*$	
Return $(N, x^2 \bmod N)$	

The key-generation algorithm pick at random primes p, q both congruent to three modulo 4 and having about the same bit-length. The state of this stateful generator is a pair (N, x) where x is a quadratic residue in \mathbb{Z}_N^* . $\text{Parity}(x)$ is 0 if x is even and 1 if x is odd. The length of an output block is one bit. \square

Note that in the original generator of [12] one might elect to keep the factorization of N as part of the seed, and use it in computing the outputs of the generator. This is useful for performance reasons: doing the squaring modulo the primes and then using Chinese Remainders is significantly faster than doing the squaring modulo N directly. However this is not an option with the forward secure version. Had the factorization been part of the state, forward security would have been compromised: computing square roots modulo primes is easy. As it is we can only have the modulus in the state.

Forward security of the modified BBS generator can be easily proven, assuming the hardness of factoring numbers N of the form above, based on the results of [12].

3 Forward-Secure Message Authentication

We recall the standard notion of message authentication schemes and their security. We then introduce key-evolving message authentication schemes and a formal notion of forward-security for them. Next we show how a standard message authentication scheme can be transformed into a forward-secure one by using a forward-secure pseudorandom bit generator.

3.1 Message Authentication Schemes

The definitions here follow [7]. A message authentication scheme $\text{mas} = (\text{mas.key}, \text{mas.tag}, \text{mas.vf})$ is specified by its key-generation, tagging and verifying algorithms. We say it has key-length b if the strings (keys) output by mas.key are always of length b bits. Let f be a forging algorithm that has access to an oracle. Consider the following experiment:

```
Experiment  $\text{Exp}_{\text{mas}}^{\text{ma}}(f)$ 
 $k \xleftarrow{\$} \text{mas.key}; (M, \tau) \xleftarrow{\$} f^{\text{mas.tag}(k, \cdot)}(\text{find})$ 
If  $\text{mas.vf}(k, M, \tau) = 1$  and  $A$  did not query  $M$  to its oracle
    then return 1 else return 0
```

We let

$$\begin{aligned}\text{Adv}_{\text{mas}}^{\text{ma}}(f) &= \Pr[\text{Exp}_{\text{mas}}^{\text{ma}}(f) = 1] \\ \text{Adv}_{\text{mas}}^{\text{ma}}(q, t) &= \max_f \{\text{Adv}_{\text{mas}}^{\text{ma}}(f)\}.\end{aligned}$$

The first term is the *ma-advantage* of f in attacking mas . The second term is the *ma-advantage* of mas . The maximum is over all adversaries f that have time-complexity at most t and make at most q oracle queries. As above we adopt the convention that the time-complexity is the total worst-case execution time of the experiment above plus the size of the code of the adversary.

3.2 Forward-Secure Message Authentication Schemes

The definitions here are modeled on those for digital signatures [8].

KEY-EVOLVING MESSAGE AUTHENTICATION SCHEMES. A *key-evolving message authentication scheme* $\text{MAS} = (\text{MAS.key}, \text{MAS.tag}, \text{MAS.vf}, \text{MAS.update}, n)$ consists of four algorithms and an integer $n \geq 1$. Randomized algorithm MAS.key is run to obtain the initial key (state) K_0 . The operation of the scheme is then divided into stages $i = 1, 2, \dots, n$, and in stage i the parties use a key denoted K_i . The key at any stage is obtained from the key at the previous stage via the deterministic update algorithm: $K_i \leftarrow \text{MAS.update}(K_{i-1})$. (After the update, K_{i-1} should be deleted so that it is no longer available to an attacker who might break in.) Within stage i , the parties can generate a tag (MAC) for message M via $\langle \tau, i \rangle \leftarrow \text{MAS.tag}(K_i, M)$. (Notice that the stage number i is always a part of the tag. This is in order to tell the verifier which key to use for verification. Also notice that in order to put i in the tag, its value must be obtainable from K_i , and indeed we will always make sure K_i contains i .) In stage i , a verifier possessing K_i can generate a decision $d \in \{0, 1\}$ to reject or accept a candidate message-tag $(M, \langle \tau, i \rangle)$ via the deterministic verification algorithm: $d \leftarrow \text{MAS.vf}(K_i, M, \tau)$.

The parties can agree on some convention as to how frequent the key updates should be. For example, maybe once a day. The frequency reflects their feelings about the likelihood of break-ins: if break-ins are considered more likely the updates should be more frequent. But it also gives a means of extending the lifetime of the message authentication scheme via re-keying. After a certain number of messages have been tagged under K_{i-1} , it might be advisable to change keys.

FORWARD-SECURITY. The scheme must withstand forgery relative to past keys even if the adversary has broken in and obtained the current key. We allow an adaptive chosen-message attack under which the adversary can first obtain valid MACs of messages of its choice under whatever key the users happen to be using in the current stage, and then based on this decide when to break in. At the point it breaks in, it is returned the current key and then it wins if it can forge a new message relative to any previous key. Let us now describe and explain the experiment associated to an adversary algorithm F :

```

Experiment  $\text{Exp}_{\text{MAS}}^{\text{fsma}}(F)$ 
 $K_0 \xleftarrow{\$} \text{MAS.key} ; i \leftarrow 0 ; h \leftarrow \varepsilon$ 
Repeat
     $i \leftarrow i + 1 ; K_i \leftarrow \text{MAS.update}(K_{i-1})$ 
     $(d, h) \xleftarrow{\$} F^{\text{MAS.tag}(K_i, \cdot)}(\text{find}, h)$ 
Until  $(d = \text{forge})$  or  $(i = n)$ 
     $(M, \langle \tau, j \rangle) \xleftarrow{\$} F(\text{forge}, K_i, h)$ 
If all the following are true then return 1 else return 0
    -  $\text{MAS.vf}(K_j, M, \tau) = 1$ 
    -  $1 \leq j < i$ 
    -  $M$  was not queried of  $\text{MAS.tag}(K_j, \cdot)$ 
```

Above, the forger F runs first in a *find* stage where it gets an oracle for the tagging algorithm under the current key. At the conclusion of a stage it may decide to output $d = \text{forge}$ thereby saying it is ready to break-in. At that point it is given K_i . Run in its *forging* stage it now returns a pair $(M, \langle \tau, j \rangle)$, and wins if τ is a valid tag for message M under K_j . Of course it only wins if $j < i$ and also if it had never asked previously for the tag of M under K_j . The input h is a history used by F to maintain information across its own invocations; it might, for example, choose to record the outcome of its oracle queries here for use in the next stage. We let

$$\begin{aligned}\mathbf{Adv}_{\text{MAS}}^{\text{fsma}}(F) &= \Pr[\text{Exp}_{\text{MAS}}^{\text{fsma}}(F) = 1] \\ \mathbf{Adv}_{\text{MAS}}^{\text{fsma}}(q, t) &= \max_F \{\mathbf{Adv}_{\text{MAS}}^{\text{fsma}}(F)\}.\end{aligned}$$

The first term is the *fsma-advantage* of F in attacking **MAS**. The second term is the *fsma-advantage* of **MAS**. The maximum is over all adversaries F that have time-complexity at most t and make at most q queries *in each stage*. (So the total number of queries made can reach qn .) This is the maximum likelihood of the forward security of the message authentication scheme **MAS** being compromised by an adversary using the indicated resources. As above we adopt the convention that the time-complexity is the total worst-case execution time of the experiment above plus the size of the code of the adversary.

3.3 A General Construction

We show how a forward secure message authentication scheme can be designed given any secure standard message authentication scheme and forward-secure pseudorandom bit generator. The base key K_0 for the key-evolving message authentication scheme is a seed (initial state) of the generator. The key for stage i will be a triple $K_i = (i, k_i, St_i)$ consisting of the value i indicating the stage for which this is the key, an actual key k_i to be used with the standard message authentication scheme, and state information St_i for the generator, based on which the next key will be generated by iteration of the generator. This is detailed below.

Construction 4. Let $\text{mas} = (\text{mas.key}, \text{mas.tag}, \text{mas.vf})$ be a standard message authentication scheme with key-length b . Let $\text{GEN} = (\text{GEN.key}, \text{GEN.next}, b, n)$ be a forward secure pseudorandom bit generator with block-length b . We associate to mas, GEN the key-evolving message authentication scheme $\text{MAS} = (\text{MAS.key}, \text{MAS.tag}, \text{MAS.vf}, \text{MAS.update}, n)$ whose constituent algorithms are as follows:

Algorithm MAS.key $St_0 \xleftarrow{\$} \text{GEN.key}$ Return $(0, \varepsilon, St_0)$	Algorithm $\text{MAS.tag}((i, k, St), M)$ $\tau \leftarrow \text{mas.tag}(k, M)$ Return $\langle \tau, i \rangle$
Algorithm $\text{MAS.vf}((i, k, St), M, \langle \tau, j \rangle)$ If $j \neq i$ then return 0 $d \leftarrow \text{mas.vf}(k, M, \tau)$ Return d	Algorithm $\text{MAS.update}((i, k, St))$ $(k, St) \leftarrow \text{GEN.next}(St)$ Return $(i + 1, k, St)$

Above ε denotes the empty string. Recall that GEN.next returns a pair consisting of a pseudorandom output block (here b bits long) and an updated state; the above is saying the pseudorandom block becomes the effective new key for the underlying standard message authentication scheme. \square

SECURITY. We claim that the key-evolving message authentication scheme we constructed above is forward secure as long as the underlying message authentication scheme is secure in the standard sense and the stateful generator is forward secure. The proof of the following is in the full version of this paper [10].

Theorem 3. Let $\text{mas} = (\text{mas.key}, \text{mas.tag}, \text{mas.vf})$ be a standard message authentication scheme with key-length b , and $\text{GEN} = (\text{GEN.key}, \text{GEN.next}, b, n)$ a forward secure pseudorandom bit generator with block-length b . Let MAS be the key-evolving message authentication scheme associated to mas, GEN as per Construction 4. Then

$$\mathbf{Adv}_{\text{MAS}}^{\text{fsma}}(q, t) \leq \mathbf{Adv}_{\text{GEN}}^{\text{fsprg}}(t_1) + n \cdot \mathbf{Adv}_{\text{mas}}^{\text{ma}}(q, t_2),$$

where $t_1 = t_2 = 2t + O(n + b)$. \square

Notice that the forward secure scheme can authenticate up to qn messages (q per stage) even though the base scheme could only handle q . This is an added advantage of key-evolving constructions already highlighted in [1].

4 Forward-Secure Audit Logs

Computer audit logs contain descriptions of noteworthy events — crashes of system programs, system resource exhaustion, failed login attempts, etc. The ability to know about the occurrences of these events is critical for intrusion post-mortem analysis, since it enables the system administrator to determine the extent of the damage and possibly the method(s) of attack. The first target of an experienced attacker will be the audit log system: the attacker wishes to erase traces of the compromise, to elude detection as well as to keep the method of attack secret.

Standard audit log protection techniques typically involve writing the audit log data to some form of append-only media such as continuous-feed printers, CD-R, or DVD-R drives, or sending the log data to remote machines. In the former case, there is some form of dedicated hardware providing append-only storage semantics, preventing attackers from modifying the audit log entries. In the latter case, the hope is that with distributed logging the probability that the attackers can break into all the machines unnoticed is far lower than that of breaking into a single machine. In this case, the log data must be distributed to dedicated logging machines (i.e., one which provides no other network services, etc) rather than another “normal” machine. Otherwise common-mode failures, e.g., a bug in the system software common to all the machines, would drastically increase the chances of an attacker evading detection.

In our application of forward-secure message authentication schemes to audit logs, we use cryptographic means to try to provide the same append-only semantics as dedicated logging hardware. Unlike real append-only media, however, it is impossible to prevent data destruction, and the best that can be achieved is tamper-detection: the audit log is not tamper proof since entries can be modified or destroyed, but modifications cannot occur undetected.

In our scheme, we modify logging services such as Unix’s system log daemon (`syslogd`) to write a tag along with the log message to verify its integrity. As with standard `syslogd`, any program may request a log entry be made; `syslogd` serializes the log entries into the appropriate log file(s). The log file(s) are later sent to a separate, secure machine for analysis and verification, perhaps periodically or because intrusion was detected and the system administrator wishes to review the logs to determine the severity of the breach. This log verifier / analysis host need not be network connected, and needs to communicate with the logging service only during a set-up phase and when the log file(s) are actually transferred.

Providing integrity for audit log data is very similar to providing integrity for messages, and our threat model is very similar to that presented above for message authentication schemes. First, the attackers first adaptively generate log entries, so that the log entries and corresponding tags are available to them. This may occur if, for example, user accounts can read the system log files, or if network logging is performed. Next, the attackers break into the system and obtains the system’s current state information. Lastly, the attackers attempt to create an alternate history by forging or altering previously generated log messages.

In addition to simply trying to forge log messages, the attackers may also try to undetectably delete or reorder previously generated log messages, so in addition to simple message integrity we require *stream integrity*, where in addition to the unforgeability of messages we require that the log messages cannot be reordered or deleted undetectably. When logging with dedicated hardware, reordering and deletion of log messages is naturally prevented. In our setup, the intruder is allowed full read/write access to the complete state of the compromised machine, and thus may overwrite previously generated, locally stored log entries.

We provide forward secure stream integrity by building our audit log scheme on top of forward secure message authentication schemes. We initialize the logging service by using a forward secure key agreement protocol, so that the logging service and the log verifier share an initial secret. This initial secret is used to initialize the message authentication scheme.

When a client program requests that an audit log entry M be made, the logging service uses an internally maintained counter j to include a sequence number with the message when generating the tag. The recorded log entry is then the tuple $\langle M, \text{MAS.tag}(0, j, M) \rangle$. The sequence counter is then incremented.

To update the logging system to a new stage, we first record a log entry $\langle \varepsilon, \text{MAS.tag}(1, j) \rangle$ prior to running **MAS.update**. This log entry serves to mark the end of the stage, so that the actual number of entries made within the stage is known. After running **MAS.update**, the sequence number is reset to zero.

Verifying the log entries requires knowledge of the initial secret. It does not, however, require that the verifier know what stage the log system should be in: the attacker can only use **MAS.update** to run forward, so even if the attacker deletes the end-of-stage markers, the attacker cannot obtain the previous keys to make it appear that no roll-back had occurred. To detect such an attack it suffices to have the logging service update to the next stage and log a new (random) message prior to sending the log to the verifier. Only a logging service that has not been rolled back can do this correctly.

The sequence numbers prevent log message reordering, and the end-of-stage log entry prevents audit log truncation within previous stages. Since an attacker must break the forward-secure message authentication scheme in order to generate bogus log messages or to reorder or delete existing log messages generated in a previous stage, the security of the forward secure audit log is the same as that of the forward-secure message authentication scheme, with the proviso of the slightly shorter messages and one fewer message per stage. (We are also restricted in the number of log messages per stage by the sequence number's encoding, since it is a fixed width field.) Secure audit logs may also be built using forward secure signatures instead of forward secure message authentication codes, but currently the cost of public key operations would make that prohibitive.

A similar solution is provided in [25]. There, instead of permitting several log entries to be made per stage, rekeying is performed after each log entry is made,

so no per-stage sequence numbers are needed; the log entries are also encrypted. Our approach is a more modular design: whether a forward secure encryption scheme should be used is an orthogonal log design decision. Furthermore, our design makes use of an arbitrary forward secure message authentication scheme, thereby separating the underlying cryptographic problem from the application. By using the forward secure message authentication scheme constructed above, our audit log design inherits its provable security, while the construction of [25] has only heuristic security arguments.

Acknowledgments

The first author is supported in part by NSF grants CCR-0098123 and ANR-0129617, and by an IBM Faculty Partnership Development Award. The second author is supported in part by a 1996 National Semiconductor Faculty Development Award, the U. S. Postal Service, NSF CAREER Award CCR-9734243, and a gift from Microsoft Corp.

References

- [1] M. ABDALLA AND M. BELLARE, “Increasing the lifetime of a key: A comparative analysis of the security of rekeying techniques.” *Advances in Cryptology – ASIACRYPT ’00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000. [5](#), [14](#)
- [2] Alleged RC4. <http://home.earthlink.net/~neilbawd/arcfour.html>. [8](#)
- [3] R. ANDERSON, “Two Remarks on Public-Key Cryptology,” Manuscript, 2000, and Invited Lecture at the Fourth Annual Conference on Computer and Communications Security, Zurich, Switzerland, April 1997. [3](#), [4](#)
- [4] D. BEAVER AND S. HABER, “Cryptographic protocols provably secure against dynamic adversaries,” *Advances in Cryptology – EUROCRYPT ’92*, Lecture Notes in Computer Science Vol. 658, R. Rueppel ed., Springer-Verlag, 1992. [4](#)
- [5] M. BELLARE, R. CANETTI AND H. KRAWCZYK, “Keying hash functions for message authentication,” *Advances in Cryptology – CRYPTO ’96*, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed., Springer-Verlag, 1996. [3](#), [11](#)
- [6] M. BELLARE, A. DESAI, E. JOKIPII AND P. ROGAWAY, “A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation,” *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997. [5](#), [6](#), [7](#)
- [7] M. BELLARE, J. KILIAN AND P. ROGAWAY, “The security of cipher block chaining,” *Journal of Computer and System Sciences*, Vol. 61, No. 3, Dec 2000, pp. 362–399. [5](#), [6](#), [7](#), [9](#), [12](#)
- [8] M. BELLARE AND S. MINER, “A forward-secure digital signature scheme,” *Advances in Cryptology – CRYPTO ’99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999. [3](#), [4](#), [12](#)
- [9] M. BELLARE AND C. NAMPREMPRE, “Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm,” *Advances in Cryptology – ASIACRYPT ’00*, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed., Springer-Verlag, 2000. [4](#)

- [10] M. BELLARE AND B. YEE, “Forward-security in private-key cryptography,” Full version of this paper, available via <http://www-cse.ucse.edu/users/mihir>. 4, 9, 14
- [11] J. BLACK, S. HALEVI, H. KRAWCZYK, T. KROVETZ AND P. ROGAWAY, “UMAC: Fast and Secure Message Authentication,” *Advances in Cryptology – CRYPTO ’99*, Lecture Notes in Computer Science Vol. 1666, M. Wiener ed., Springer-Verlag, 1999. 3
- [12] L. BLUM, M. BLUM AND M. SHUB, “A simple unpredictable pseudo-random number generator,” *SIAM Journal on Computing* Vol. 15, No. 2, 364-383, May 1986. 3, 11
- [13] M. BLUM AND S. MICALI, “How to generate cryptographically strong sequences of pseudo-random bits,” *SIAM Journal on Computing*, Vol. 13, No. 4, 850-864, November 1984. 2, 3, 5, 6, 11
- [14] R. CANETTI AND A. HERZBERG, “Maintaining security in the presence of transient faults,” *Advances in Cryptology – CRYPTO ’94*, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994. 5
- [15] C.-S. CHOW AND A. HERZBERG, “Network randomization protocol: A proactive pseudo-random generator,” *Proceedings of the 5th Usenix Unix Security Symposium*, June 1995. 5
- [16] A. DESAI, A. HEVIA AND L. YIN, “A Practice-Oriented Treatment of Pseudorandom Number Generators,” *Advances in Cryptology – EUROCRYPT ’02*, Lecture Notes in Computer Science Vol. 2332 , L. Knudsen ed., Springer-Verlag, 2002. 5
- [17] Y. DESMEDT, “Threshold cryptography,” *European Trans. on Telecommunications*, Vol. 5, No. 4, pp. 449-457, July-August 1994. 4
- [18] W. DIFFIE, P. VAN OORSCHOT AND M. WIENER, “Authentication and authenticated key exchanges”, *Designs, Codes and Cryptography*, 2, 1992, pp. 107–125. 4
- [19] O. GOLDREICH, S. GOLDWASSER AND S. MICALI, “How to construct random functions,” *Journal of the ACM*, Vol. 33, No. 4, 1986, pp. 210–217. 9
- [20] C. GÜNTHER, “An identity-based key-exchange protocol,” *Advances in Cryptology – EUROCRYPT ’89*, Lecture Notes in Computer Science Vol. 434, J-J. Quisquater, J. Vandewille ed., Springer-Verlag, 1989. 4
- [21] H. KRAWCZYK, “Simple forward-secure signatures from any signature scheme,” *Proceedings of the 7th Annual Conference on Computer and Communications Security*, ACM, 2000. 4
- [22] A. HERZBERG, S. JARECKI, H. KRAWCZYK AND M. YUNG, “Proactive secret sharing, or: How to cope with perpetual leakage,” *Advances in Cryptology – CRYPTO ’95*, Lecture Notes in Computer Science Vol. 963, D. Coppersmith ed., Springer-Verlag, 1995. 4
- [23] J. KATZ, “A forward-secure public-key encryption scheme,” Cryptology ePrint Archive: Report 2002/060, May 2002, <http://eprint.iacr.org/2002/060/>. 4
- [24] U. S. National Institute of Standards and Technology, “Federal information processing standards publication 140-1: Security requirements for cryptographic modules”, January 1994. 5
- [25] B. SCHNEIER AND J. KELSEY, “Cryptographic support for secure logs on untrusted machines,” ACM TISSEC, Vol. 2, 1999. Preliminary version in *Proceedings of the 7th USENIX Security Symposium*, USENIX Press, 1998. 3, 16, 17
- [26] A. YAO, “Theory and applications of trapdoor functions,” *Proceedings of the 23rd Symposium on Foundations of Computer Science*, IEEE, 1982. 2, 5, 6

Intrusion-Resilient Public-Key Encryption

Yevgeniy Dodis¹, Matt Franklin², Jonathan Katz³,
Atsuko Miyaji^{2,4}, and Moti Yung⁵

¹ Department of Computer Science, New York University
`dodis@cs.nyu.edu`

² Department of Computer Science, University of California, Davis
`{miyaji,franklin}@cs.ucdavis.edu`

³ Department of Computer Science, University of Maryland, College Park
`jkatz@cs.umd.edu`

⁴ Japan Advanced Institute of Science and Technology
`miyaji@jaist.ac.jp`

⁵ Department of Computer Science, Columbia University
`moti@cs.columbia.edu`

Abstract. Exposure of secret keys seems to be inevitable, and may in practice represent the most likely point of failure in a cryptographic system. Recently, the notion of *intrusion-resilience* [17] (which extends both the notions of *forward security* [3, 5] and *key insulation* [11]) was proposed as a means of mitigating the harmful effects that key exposure can have. In this model, time is divided into distinct periods; the public key remains fixed throughout the lifetime of the protocol but the secret key is periodically updated. Secret information is stored by both a *user* and a *base*; the user performs all cryptographic operations during a given time period, while the base helps the user periodically update his key. Intrusion-resilient schemes remain secure in the face of multiple compromises of both the user and the base, as long as they are not both compromised simultaneously. Furthermore, in case the user and base *are* compromised simultaneously, prior time periods remain secure (as in forward-secure schemes).

Intrusion-resilient signature schemes have been previously constructed [17, 15]. Here, we give the first construction of an intrusion-resilient public-key encryption scheme, based on the recently-constructed forward-secure encryption scheme of [8]. We also consider generic transformations for securing intrusion-resilient encryption schemes against chosen-ciphertext attacks.

1 Introduction

Exposure of secret keys is perhaps the most debilitating attack on a cryptosystem since it typically implies that all security guarantees are lost. This problem is emerging as an ever-greater threat as cryptographic primitives are deployed on inexpensive, lightweight, and mobile devices; in these cases, it is typically much easier for an adversary to break into the device and obtain the secret keys than

to crack the computational assumptions on which the system is based. Clearly, concerns about key exposure must be addressed in a satisfactory manner by the research community.

Recognizing the need to address these concerns, a long line of research has focused on dealing with the threat of key exposure. Methods to prevent key exposure entirely (e.g., by using tamper-resistant devices) seem cost-prohibitive and impractical for most common applications. Thus, research has focused on making key exposures more difficult, or, alternately, minimizing the damage when (partial) key exposure occurs. As an example, threshold cryptography [10, 9] distributes secrets among n devices so that exposure of secrets from, say, t of these devices will not allow an adversary to “break” the scheme. On the other hand, this requires that at least $t + 1$ devices participate every time a cryptographic operation must be performed. While this may be acceptable in some scenarios, this does not seem appropriate for mobile users and in other settings where the risk of key exposure is high but users need the ability to perform cryptographic computations on their own.

Alternative approaches to this problem have been proposed whereby the key required for cryptographic computations always resides on a single device. In such proposals, time is divided into distinct periods $1, \dots, N$ and secret keys *evolve* over time (public keys, however, are fixed for the lifetime of the scheme). The goal here is to contain — as much as possible — damage from key exposures that occur. *Forward-secure cryptosystems* [3, 5] were the first solutions in this vein. In forward-secure schemes, the secret key is stored by a single user and this key is updated by the user at the beginning of every time period. An adversary who exposes a key for period t can perform cryptographic operations (signing, decrypting, etc.) for periods $t' \geq t$ but cannot break the scheme (in the appropriate sense) for any prior time periods $t' < t$. The effect of key exposure is thereby contained as much as possible given that a single user stores all secret keying information.

To address the issue of obtaining security for time periods *following* key exposure, the notion of *key-insulation* [11, 12] was proposed. This model had the distinguishing feature of assuming (a limited amount of) secure storage on a server with which the user periodically interacts. (Here, one can imagine the “user” as a mobile device and the “server” as a desktop PC in the user’s home.) The user can perform all cryptographic operations during any particular time period on his own and can also update his keys — with the help of the server — at discrete time intervals as above. Because a limited amount of secure storage is assumed, the security obtainable here is better than in the forward-secure case; in particular, schemes can be designed such that an adversary who exposes keys stored by the user multiple times (i.e., at time periods $T = \{t_1, \dots, t_\ell\}$) cannot “break” the scheme for *any* other time period either in the past or in the future (i.e., for any time periods $t \notin T$). In *strong* key-insulated schemes, exposing only the secret keys stored on the server does not permit an adversary to “break” the scheme at all.

Recently, these models were synthesized into the most powerful notion set forth to date: *intrusion-resilience* [17]. As in the key-insulated model, this model assumes a user who performs all cryptographic operations and a server with which the user interacts to update his keys at discrete time intervals. Now, however, it is no longer assumed that the server is secure. Since key exposures at the server are now assumed to occur frequently, the user and server have the option of “refreshing” their secrets (this is reminiscent of proactivation [23]). Here (informally speaking; see the formal definition in Section 2), schemes can be designed such that an adversary who exposes keys stored at *both* the user and the server on multiple occasions — but never at the same time — cannot “break” the scheme for any time periods other than those for which keys were exposed at the user. Furthermore, in case the keys of the user and server *are* both exposed at some time t (and no refresh was performed in between these exposures), the scheme remains forward-secure so that the adversary cannot “break” the scheme at any prior time periods $t' < t$.

We note that each of these models may be appropriate in different environments. Forward-secure schemes are advantageous in that the user is self-sufficient and need not interact with any other device. On the other hand, the security provided by key-insulated and intrusion-resilient schemes is better and these schemes might therefore be used when interacting with a server is feasible and does not represent a serious drawback. Finally, although the intrusion-resilient model offers stronger security guarantees than the key-insulated model, we note that solutions for the latter are (thus far) much more efficient. The choice of which type of scheme to use therefore depends heavily on an assumption about the (physical) security of the server.

1.1 Our Contributions

Much work has focused on the design and analysis of forward-secure signature schemes [5, 20, 2, 16, 21] and, more recently, a forward-secure public-key encryption scheme has been constructed [8]. Key insulated public-key encryption schemes [11, 7, 6] and signature schemes [12] are also known. Thus far, however, only constructions of intrusion-resilient signature schemes [17, 15] have been proposed.¹ Here, we give the first definitions of intrusion-resilient public-key encryption and the first construction of an intrusion-resilient public-key encryption scheme. Our scheme is based on the recent forward-secure encryption scheme of [8], and the security of our scheme is therefore based on the BDH assumption in the random oracle model. As in [8], we may modify our scheme so as to achieve semantic security in the *standard model* under the *decisional* BDH assumption.

We also consider generic transformations for securing intrusion-resilient public-key encryption schemes against adaptive chosen-ciphertext attacks (in the random oracle model). We show that *any* such transformation that works for

¹ As mentioned by [17], forward-secure public-key encryption is necessary in order to build intrusion-resilient public-key encryption. At that time, however, no non-trivial forward-secure public-key encryption schemes were known.

“standard” public-key encryption schemes also works for intrusion-resilient public-key encryption schemes.² In particular, then, we may apply known conversions (e.g., those of [13, 22]) to our scheme so as to obtain the first intrusion-resilient public-key encryption scheme secure against chosen-ciphertext attacks.

2 Definitions and Preliminaries

The definitions given here are the first to appear for the case of intrusion-resilient encryption; they exactly parallel those appearing in [17] for the case of intrusion-resilient signatures.

In our model time is divided into distinct periods labeled $1, \dots, N$. We have a *base* and a *user* who (jointly) establish a public key which remains fixed for the duration of the protocol. Encryption of a message depends on the current time period; thus, ciphertexts have the form $\langle t, C \rangle$ where t indicates the time period during which encryption was performed. The base and the user each store secret keying information: at time period t , the user stores sufficient information to decrypt messages sent during time t (so, to decrypt, the user does not need to contact the base). At the beginning of time period $t+1$, the base updates its keys and sends an *update message* to the user; this update message, in particular, will contain information enabling the user to compute the key needed to decrypt during period $t+1$. Finally, the base can also send *refresh messages* to the user at any point in time; if a refresh is executed between compromises of the user and the base, the system remains secure (refreshes thus serve a similar purpose to proactivation [23]).

As in [17], the adversary in our model can obtain secrets from the base and the user for multiple, adaptively-chosen time periods. The adversary can also intercept update and refresh messages sent by the base. Informally speaking (a precise definition is given below), if the adversary only compromises the base, the encryption scheme remains secure. If the adversary compromises the user repeatedly, the encryption scheme remains secure except for periods whose secrets were obtained (either directly or by combination of compromise and interception of update/refresh messages). Finally, even if the base and the user are compromised during the same time period (and no refresh was sent in the interim), the encryption scheme remains secure for prior time periods.

2.1 Functional Specification

We first define correct operation of the scheme, and defer a definition of security to the next section. The notation $SK_{t,r}$ (respectively, $SKB_{t,r}$) denotes the user’s (respectively, base’s) secret key for time period t following r refreshes. We say $t.r = t'.r'$ if $t = t'$ and $r = r'$. We say $t.r < t'.r'$ if $t < t'$ or if $t = t'$ and $r < r'$. As in prior work, we assume for convenience that a key update occurs immediately after key generation to obtain keys for $t = 1$, and that a key refresh occurs immediately after every key update to obtain keys for $r = 1$.

² As pointed out in [8], this does *not* seem to be the case for forward-secure encryption schemes.

Definition 1. A key-updating public-key encryption scheme Π consists of the following PPT algorithms:

- **Gen**, the key generation algorithm, takes as input security parameter 1^k and the total number of time periods N . It outputs the initial user key $SK_{0.0}$, the initial base key $SKB_{0.0}$, and the public key PK .
- **Enc**, the encryption algorithm, takes as input the public key PK , a time period t , and a message m . It outputs ciphertext $\langle t, C \rangle$. (Note that refreshes are transparent to the sender, but updates are not since they occur at well-defined intervals.)
- **Dec**, the decryption algorithm, takes as input the current user key $SK_{t.r}$ and a ciphertext $\langle t, C \rangle$. It outputs a message m . (As usual, we require that, for any r , we have $\text{Dec}(SK_{t.r}, \text{Enc}(PK, t, m)) = m$.)
- **UpdB**_{Base}, the base key update algorithm, takes as input the current base key $SKB_{t.r}$. It outputs a new base key $SKB_{t+1.0}$ for the next time period as well as key update message SKU_t .
- **UpdUser**, the user key update algorithm, takes as input the current user key $SK_{t.r}$ and a key update message SKU_t . It outputs the new user key $SK_{t+1.0}$ for the next time period.
- **RefBase**, the base key refresh algorithm, takes as input the current base key $SKB_{t.r}$. It outputs a new base key $SKB_{t.r+1}$ as well as key refresh message $SKR_{t.r}$.
- **RefUser**, the user key refresh algorithm, takes as input the current user key $SK_{t.r}$ and a key refresh message $SKR_{t.r}$. It outputs a new user key $SK_{t.r+1}$.

2.2 Definition of Security

To aid our definition of security, we let $RN(t)$ denote the number of refreshes that occur in time period t . Recall that each update is assumed to be followed by an immediate refresh, so keys with $r = 0$ are never actually used. RN is for notational convenience only; it need not be known in advance.

Consider the following “thought experiment” which generates all keying information for the entire lifetime of the scheme:

```

Experiment Generate-Keys( $k, N, RN$ )
 $t := 0, r := 0$ 
 $(SK_{0.0}, SKB_{0.0}, PK) \leftarrow \text{Gen}(1^k, N)$ 
for  $t = 1$  to  $N$ 
   $(SKB_{t.0}, SKU_{t-1}) \leftarrow \text{UpdB}(SKB_{(t-1).r})$ 
   $SK_{t.0} \leftarrow \text{UpdUser}(SK_{(t-1).r}, SKU_{t-1})$ 
  for  $r = 1$  to  $RN(t)$ 
     $(SKB_{t.r}, SKR_{t.(r-1)}) \leftarrow \text{RefBase}(SKB_{t.(r-1)})$ 
     $SK_{t.r} \leftarrow \text{RefUser}(SK_{t.(r-1)}, SKR_{t.(r-1)})$ 

```

Let SK^* , SKB^* , SKU^* , and SKR^* denote the sets of user keys, base keys, update messages, and refresh messages generated in the course of the above

experiment. We now define the following oracles available to the adversary:

- LR, the *left-or-right* encryption oracle. On input (t, m_0, m_1) the oracle chooses a random bit b and returns $\text{Enc}(PK, t, m_b)$. This oracle may be queried only once³ by the adversary, and allows us to define a notion of security.
- Osec, the *key exposure* oracle (based on the sets SK^* , SKB^* , SKU^* , and SKR^*) which:
 1. On input (“s”, $t.r$) outputs $SK_{t.r}$;
 2. On input (“b”, $t.r$) outputs $SKB_{t.r}$;
 3. On input (“u”, t) outputs SKU_t and $SKR_{(t+1).0}$;
 4. On input (“r”, $t.r$) outputs $SKR_{t.r}$.

Queries to this oracle correspond to compromise of the user or base, or to intercepting update or refresh messages. (We assume that queries to this oracle always have t, r within the appropriate bounds.)

For any set Q of key exposure queries, we say that $SK_{t.r}$ is Q -exposed if at least one of the following is true:

- $(“s”, t.r) \in Q$
- $r > 1$, $(“r”, t.(r-1)) \in Q$, and $SK_{t.(r-1)}$ is Q -exposed
- $r = 1$, $(“u”, t-1) \in Q$, and $SK_{(t-1).RN(t-1)}$ is Q -exposed

A completely analogous definition may be given for Q -exposure of a base key $SKB_{t.r}$.

Clearly, if $SK_{t.r}$ is Q -exposed then the adversary can decrypt messages encrypted during time period t . Similarly, if $SK_{t.r}$ and $SKB_{t.r}$ are both Q -exposed (for some t, r) then the adversary can run the update algorithms itself and thereby decrypt messages encrypted during any time period $t' \geq t$. Thus, we say the scheme is (t, Q) -compromised if either $SK_{t.r}$ is Q -exposed (for some r) or if both $SK_{t'.r}$ and $SKB_{t'.r}$ are Q -exposed (for some r and $t' < t$).

We say the adversary *succeeds* if it can determine the bit b used by the LR oracle, where this oracle was queried on a time period t for which the scheme was not (t, Q) -compromised. More formally, consider the following experiment executed with some algorithm A :

Experiment Run-Adversary(A, k, N, RN)

Generate-Keys(k, N, RN)

$b' \leftarrow A^{\text{LR}, \text{Osec}}(1^k, N, PK, RN)$

Let Q be the set of queries made by A to Osec

Let (t, m_0, m_1) be the query made to LR

Let b be the bit used by LR in responding to the query

if $b \neq b'$ or the scheme is (t, Q) -compromised

return 0

else return 1

We define A 's probability of success as the probability that 1 is output in the above experiment. The *advantage* of adversary A in attacking scheme Π

³ A hybrid argument (as in [4]) shows that security under a single access to LR is equivalent to security under polynomially-many accesses to LR. We thus use the simpler definition for convenience.

(where N and $RN(\cdot)$ are assumed to be clear from context, and are always at most polynomial in k) is defined as twice A 's probability of success, minus 1. We denote this advantage by $\text{Adv}_{A,\Pi}(k)$. We may now define security of an intrusion-resilient scheme.

Definition 2. A key-updating public-key encryption scheme Π is said to be an intrusion-resilient scheme achieving semantic security if, for all PPT adversaries A and all $N, RN(\cdot)$ polynomial in k , we have $\text{Adv}_{A,\Pi}(k) < \varepsilon(k)$ for some negligible function $\varepsilon(\cdot)$.

A definition appropriate for describing the concrete security of Π may be easily derived from the above. A definition of security against adaptive chosen-ciphertext attacks is also evident, and we defer such a definition to the full version of this paper.

2.3 Cryptographic Assumptions

The security of our scheme is based on the difficulty of the bilinear Diffie-Hellman (BDH) problem as recently formalized by Boneh and Franklin [7] (see also [19, 18]). We review the relevant definitions as they appear in [7]. Let \mathbb{G}_1 and \mathbb{G}_2 be two cyclic groups of prime order q , where \mathbb{G}_1 is represented additively and \mathbb{G}_2 is represented multiplicatively. We use a map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ for which the following hold:

1. The map \hat{e} is *bilinear*; that is, for all $P_0, P_1 \in \mathbb{G}_1$ and all $x, y \in \mathbb{Z}_q$ we have $\hat{e}(xP_0, yP_1) = \hat{e}(yP_0, xP_1) = \hat{e}(P_0, P_1)^{xy}$.
2. There is an efficient algorithm to compute $\hat{e}(P_0, P_1)$ for any $P_0, P_1 \in \mathbb{G}_1$.

A *BDH parameter generator* \mathcal{IG} is a randomized algorithm that takes a security parameter 1^k , runs in polynomial time, and outputs the description of two groups $\mathbb{G}_1, \mathbb{G}_2$ and a map \hat{e} satisfying the above conditions. We define the *BDH problem with respect to \mathcal{IG}* as the following: given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ output by \mathcal{IG} along with random $P, aP, bP, cP \in \mathbb{G}_1$, compute $\hat{e}(P, P)^{abc}$. We say that \mathcal{IG} satisfies the *BDH assumption* if the following is negligible (in k) for all PPT algorithms A :

$$\Pr[(\mathbb{G}_1, \mathbb{G}_2, \hat{e}) \leftarrow \mathcal{IG}(1^k); P \leftarrow \mathbb{G}_1; a, b, c \leftarrow \mathbb{Z}_q : A(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, aP, bP, cP) = \hat{e}(P, P)^{abc}].$$

We note that BDH parameter generators for which the BDH assumption is believed to hold can be constructed from Weil and Tate pairings associated with supersingular elliptic curves or abelian varieties. As our results do not depend on any specific instantiation, we refer the interested reader to [7] for details.

3 Construction

Our construction builds on the forward-secure encryption scheme of [8], which is based on previous work of [14] (both of which were enabled by the identity-based encryption scheme of [7]). We assume here that the reader is familiar with the scheme of [8]; in fact, many elements of their scheme are used directly here.

3.1 Scheme Intuition

Assume for simplicity that the total number of time periods N is a power of 2; that is, $N = 2^\ell$. We imagine a full binary tree of height ℓ in which the root is labeled with ε (representing the empty string) and furthermore if a node at depth less than ℓ is labeled with w then its left child is labeled with $w0$ and its right child is labeled with $w1$. Let $\langle t \rangle$ denote the ℓ -bit representation of integer t (where $0 \leq t \leq 2^\ell - 1$). The leaves of the tree (which are labeled with strings of length ℓ) correspond to successive time periods in the obvious way; i.e., time period t is associated with the leaf labeled by $\langle t \rangle$. For simplicity, we refer to the node labeled by w as simply “node w ”. Every node $w = w_1 \dots w_j$ in the tree will have an associated “secret point” $S_w \in \mathbb{G}_1$ and all interior nodes also have an associated “translation point” $Q_w \in \mathbb{G}_1$. For all nodes we then have the “local secret key” $sk_w = (S_w, Q_w)$, where $Q_w = (Q_{w_1}, \dots, Q_{w_1 \dots w_{j-1}})$. We remark that while the translation points are needed for efficient decryption, they do not need to be kept secret.

The properties of these keys will be as follows:

1. To decrypt a message encrypted using PK during period t , only key $sk_{\langle t \rangle}$ is needed.
2. Given key sk_w , it is possible to efficiently derive keys sk_{w0} and sk_{w1} .
3. Given PK and t , and *without* sk_w for all prefixes w of $\langle t \rangle$, it is infeasible to derive $sk_{\langle t \rangle}$ and furthermore infeasible to decrypt messages encrypted during period t .

Once we have a scheme satisfying the above requirements, we utilize the following method. For a given period t , let $t_0 t_1 \dots t_\ell = \langle t \rangle$, where $t_0 = \epsilon$. The “global secret key” gsk_t for this period will consist of (1) $sk_{\langle t \rangle}$ and also (2) $\{sk_{t_0 t_1 \dots t_{j-1} 0}\}$ for all $1 \leq j \leq \ell$ such that $t_j = 0$. We denote the latter keys (i.e., the “local secret keys” for all right siblings of nodes on the path from $\langle t \rangle$ to the root) by $\rho(t)$. We refer to the right sibling corresponding to swapping the “last” 0 of $\langle t \rangle$ to 1 as the *deepest sibling*. Also, we let $\text{Sec}_{\langle t \rangle} = (\{S_w \mid w \in \rho(t)\})$ be the set of secret points corresponding to nodes in $\rho(t)$. Since there is some redundant information stored as part of gsk_t (in particular, the translation points do not need to be stored multiple times as part of each local secret key), we may note that gsk_t actually consists of $S_{\langle t \rangle}$, $\text{Sec}_{\langle t \rangle}$, and $Q_{\langle t \rangle}$.

We may notice that gst_t enables derivation of all the local secret keys for periods t through N (indeed, one can easily derive gsk_{t+1} from gsk_t), but none of the local secret keys for periods $t' < t$. This will allow us to achieve forward security, as in [8]. However, in our model we also need to proactively split gsk_t between the user and the base, so that we can derive the sharing for period $t+1$ from that of period t . To achieve this, we let the user store $sk_{\langle t \rangle}$ — to enable decryption within the current time period — but additively share each secret point in $\text{Sec}_{\langle t \rangle}$ between the user and the base. Intuitively, $sk_{\langle t \rangle}$ by itself only allows the user to decrypt at period t , and the fact that the rest of the global key $\text{Sec}_{\langle t \rangle}$ is split ensures that exposure of the user cannot compromise any of the future periods. Security against compromises of the base is similar (and

even simpler since the base does not even store $\mathcal{Q}_{\langle t \rangle}$, except that here, even the current period is secure. Proactivation is simple as well: the base simply randomly refreshes the 2-sharing of $\text{Sec}_{\langle t \rangle}$. This gives us full intrusion-resilience.

The only issue to resolve is how to update (the sharing of) gsk_t to (the sharing of) gsk_{t+1} without compromising security. We notice that this procedure requires choosing several new secret points and translation points. We cannot let the user or the base generate these on their own, since this will compromise the security of the scheme if the corresponding player is corrupted during this phase. Instead, we will have the user and base generate these points *jointly*. The challenge is to do it via a single message from the base to the user. We give a high level description of the main step for doing so. For any node w , if we let $Q_w = s_w P$ for some s_w (where P is some fixed public point) we will have $S_{w0} = S_w + s_w H_1(w0)$ and $S_{w1} = S_w + s_w H_1(w1)$ (here, H_1 is a hash function defined as part of the scheme). Assume the user and base already have a random sharing $S_w = S'_w + S''_w$, and wish to generate Q_w and a random sharing of S_{w0} and S_{w1} . The base chooses a random s'_w , sets $Q'_w = s'_w P$, $S'_{w0} = S'_w + s'_w H_1(w0)$, and $S'_{w1} = S'_w + s'_w H_1(w1)$, and sends Q'_w to the user. The user chooses a random s''_w , sets $Q_w = Q'_w + s''_w P$ (implicitly, $Q_w = (s'_w + s''_w)P$, so $s_w = s'_w + s''_w$), $S''_{w0} = S''_w + s''_w H_1(w0)$, and $S''_{w1} = S''_w + s''_w H_1(w1)$. This step is immediately followed by a random refresh, which ensures that no single party has enough control to cause any security concerns.

3.2 Formal Description

We assume that hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \rightarrow \{0, 1\}^n$ are defined, either by Gen or else as part of the specification of the scheme. These hash functions will be treated as random oracles in the analysis.

Gen($1^k, N = 2^\ell$) does the following:

1. $\mathcal{IG}(1^k)$ is run to generate group $\mathbb{G}_1, \mathbb{G}_2$ (of order q) and \hat{e} .
2. $P \leftarrow \mathbb{G}_1$; $s_\varepsilon \leftarrow \mathbb{Z}_q$. Set $Q = s_\varepsilon P$.
3. The public key is $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$.
4. Set $S_0 = s_\varepsilon H_1(0)$ and $S_1 = s_\varepsilon H_1(1)$.
5. For $j = 1, \dots, \ell - 1$:
 - (a) $s_{0j} \leftarrow \mathbb{Z}_q$. Set $Q_{0j} = s_{0j} P$.
 - (b) Set $S_{0j0} = S_{0j} + s_{0j} H_1(0^j0)$ and $S_{0j1} = S_{0j} + s_{0j} H_1(0^j1)$.
6. (Note that we have $\mathcal{Q}_{\langle 0 \rangle} = \{Q_0, \dots, Q_{0^{\ell-1}}\}$, $sk_{\langle 0 \rangle} = (S_{\langle 0 \rangle}, \mathcal{Q}_{\langle 0 \rangle})$, and $\text{Sec}_{\langle 0 \rangle} = (S_1, \dots, S_{0^{\ell-1}1})$.)
7. Pick random $\text{Sec}'_{\langle 0 \rangle}$ and $\text{Sec}''_{\langle 0 \rangle}$ such that $\text{Sec}_{\langle 0 \rangle} = \text{Sec}'_{\langle 0 \rangle} + \text{Sec}''_{\langle 0 \rangle}$ (i.e., for each $w \in \rho(\langle 0 \rangle)$ we have $S_w = S'_w + S''_w$). Set $SKB_{0,0} = \text{Sec}'_{\langle 0 \rangle}$, $SK_{0,0} = (sk_{\langle 0 \rangle}, \text{Sec}''_{\langle 0 \rangle})$.
8. Output $PK, SK_{0,0}$, and $SKB_{0,0}$ and erase all other information.

UpdB_{Base}(SKB_{t,r}) does the following:

1. Parse $\langle t \rangle$ as $t_0 t_1 \dots t_\ell$ where $t_0 = \varepsilon$ for convenience. Parse SKB_{t,r} as $\text{Sec}'_{\langle t \rangle} = \{S'_{t_0 \dots t_{j-1}} \mid t_j = 0\}$.
2. If $t_\ell = 0$, erase $S'_{\langle t \rangle}$ and set $\text{Sec}'_{\langle t+1 \rangle}$ equal to the remaining keys. Note that $S'_{\langle t+1 \rangle}$ is available, since it was stored as part of $\text{Sec}_{\langle t \rangle}$.
3. Otherwise, let i be the largest value such that $t_i = 0$. Denote by $w = t_0 \dots t_{i-1}$ the “deepest sibling” of t . For $j = 0, \dots, \ell - i - 1$:
 - (a) $s'_{w0^j} \leftarrow \mathbb{Z}_q$. Set $Q'_{w0^j} = s'_{w0^j} P$, $S'_{w0^{j+1}} = S'_{w0^j} + s'_{w0^j} H_1(w0^j 0)$, and $S'_{w0^{j+1}} = S'_{w0^j} + s'_{w0^j} H_1(w0^j 1)$.
4. Erase share S'_w and replace it by the resulting $(\ell - i)$ shares $\{S'_{w0^j}\}$ above, thus obtaining the new vector $\text{SKB}_{t+1,0} = \text{Sec}'_{\langle t+1 \rangle}$.
5. Set $\text{SKU}_t = (S'_{\langle t+1 \rangle}, Q'_w, Q'_{w0}, \dots, Q'_{w0^{\ell-i-1}})$. (Note that when $t_\ell = 0$ only $S'_{\langle t+1 \rangle}$ is sent.)
6. Output $\text{SKB}_{t+1,0}$, SKU_t .

UpdUser(SK_{t,r}, SKU_t) does the following:

1. Parse $\langle t \rangle$ as $t_0 t_1 \dots t_\ell$ where $t_0 = \varepsilon$ for convenience. Parse SK_{t,r} as $(S_{\langle t \rangle}, \mathcal{Q}_{\langle t \rangle}, \text{Sec}''_{\langle t \rangle})$, where $\text{Sec}''_{\langle t \rangle} = \{S''_{t_0 \dots t_{j-1}} \mid t_j = 0\}$. Erase $S_{\langle t \rangle}$.
2. If $t_\ell = 0$, erase $S''_{\langle t \rangle}$ and set $\text{Sec}''_{\langle t+1 \rangle}$ equal to the remaining keys. Note that $S_{\langle t+1 \rangle}$ is available, since it was stored as part of $\text{Sec}_{\langle t \rangle}$.
3. Otherwise, let i be the largest value such that $t_i = 0$. Denote by $w = t_0 \dots t_{i-1}$ the “deepest sibling” of t . Parse SKU_t as $(S'_{\langle t+1 \rangle}, Q'_w, Q'_{w0}, \dots, Q'_{w0^{\ell-i-1}})$. For $j = 0, \dots, \ell - i - 1$:
 - (a) $s''_{w0^j} \leftarrow \mathbb{Z}_q$. Set $Q_{w0^j} = Q'_{w0^j} + s''_{w0^j} P$, $S''_{w0^{j+1}} = S''_{w0^j} + s''_{w0^j} H_1(w0^j 0)$, and $S''_{w0^{j+1}} = S''_{w0^j} + s''_{w0^j} H_1(w0^j 1)$.
4. Erase share S''_w and replace it by the resulting $(\ell - i)$ shares $\{S''_{w0^j}\}$, thus obtaining the new vector $\text{Sec}''_{\langle t+1 \rangle}$. For $j = 0, \dots, \ell - i - 1$, erase $Q_{t_0 \dots t_{i-1} 01^j}$ and replace it by $Q_{t_0 \dots t_{i-1} 10^j}$. Thus, we obtain the new vector $\mathcal{Q}_{\langle t+1 \rangle}$.
5. Set $S_{\langle t+1 \rangle} = S'_{\langle t+1 \rangle} + S''_{\langle t+1 \rangle}$.
6. Output $\text{SK}_{t+1,0} = (S_{\langle t+1 \rangle}, \mathcal{Q}_{\langle t+1 \rangle}, \text{Sec}''_{\langle t+1 \rangle})$.

RefBase(SKB_{t,r}) does the following:

1. Parse SKB_{t,r} as $\text{Sec}'_{\langle t \rangle} = (\{S'_w \mid w \in \rho(t)\})$. For each $w \in \rho(t)$, pick random $R_w \in \mathbb{G}_1$ and reset each $S'_w := S'_w + R_w$.
2. Output resulting SKB_{t,r+1} and SKR_{t,r} = ($\{R_w \mid w \in \rho(t)\}$).

RefUser(SK_{t,r}, SKR_{t,r}) does the following:

1. Parse SK_{t,r} as $(sk_{\langle t \rangle}, \text{Sec}''_{\langle t \rangle})$, where $\text{Sec}''_{\langle t \rangle} = (\{S''_w \mid w \in \rho(t)\})$. Parse SKR_{t,r} as ($\{R_w \mid w \in \rho(t)\}$). For each $w \in \rho(t)$, reset $S''_w := S''_w - R_w$.
2. Output resulting SK_{t,r+1}.

$\text{Enc}_{PK}(t, M)$ (where $M \in \{0, 1\}^n$) does the following:

1. Let $t_1 \cdots t_\ell = \langle t \rangle$. Select random $r \leftarrow \mathbb{Z}_q$.
2. Output $\langle t, C \rangle$ where:

$$C = (rP, rH_1(t_1t_2), \dots, rH_1(t_1 \cdots t_\ell), M \oplus H_2(\hat{e}(Q, H_1(t_1))^r)).$$

$\text{Dec}_{SK_{t,r}}(\langle t, C \rangle)$ does the following:

1. Parse $\langle t \rangle$ as $t_1 \cdots t_\ell$. Parse $SK_{t,r}$ as $(sk_{\langle t \rangle}, \dots)$ and $sk_{\langle t \rangle}$ as $(S_{\langle t \rangle}, Q_{\langle t \rangle})$ where $Q_{\langle t \rangle} = (Q_{t_1}, \dots, Q_{t_1 \cdots t_{\ell-1}})$. Parse C as $(U_0, U_1, \dots, U_\ell, V)$.
2. Compute

$$M = V \oplus H_2 \left(\frac{\hat{e}(U_0, S_{\langle t \rangle})}{\prod_{j=2}^{\ell} \hat{e}(Q_{t_1 \cdots t_{j-1}}, U_j)} \right).$$

We now verify that decryption is performed correctly. When encrypting, we have $\hat{e}(Q, H_1(t_1))^r = \hat{e}(P, H_1(t_1))^{rs_\varepsilon}$. When decrypting, we have $U_0 = rP$, $U_1 = rH_1(t_1t_2), \dots, U_\ell = rH_1(t_1 \cdots t_\ell)$ so that

$$\begin{aligned} \frac{\hat{e}(U_0, S_{\langle t \rangle})}{\prod_{j=2}^{\ell} \hat{e}(Q_{t_1 \cdots t_{j-1}}, U_j)} &= \frac{\hat{e}\left(rP, s_\varepsilon H_1(t_1) + \sum_{j=2}^{\ell} s_{t_1 \cdots t_{j-1}} H_1(t_1 \cdots t_j)\right)}{\prod_{j=2}^{\ell} \hat{e}(s_{t_1 \cdots t_{j-1}} P, rH_1(t_1 \cdots t_j))} \\ &= \frac{\hat{e}(P, H_1(t_1))^{rs_\varepsilon} \cdot \prod_{j=2}^{\ell} \hat{e}(P, H_1(t_1 \cdots t_j))^{rs_{t_1 \cdots t_{j-1}}}}{\prod_{j=2}^{\ell} \hat{e}(P, H_1(t_1 \cdots t_j))^{rs_{t_1 \cdots t_{j-1}}}} \\ &= \hat{e}(P, H_1(t_1))^{rs_\varepsilon} \end{aligned}$$

and thus decryption succeeds.

3.3 Efficiency

Our scheme enjoys the same parameters as the forward-secure scheme of [8]. In fact, our scheme is exactly the “intrusion-resilient” extension of their forward-secure scheme. In particular, our scheme has public key of size $O(1)$, and all other parameters are $O(\log N)$ including: key generation time, encryption/decryption time, ciphertext length, key update time and message length, key refresh time and message length, and user/base storage.

3.4 Extensions

The full version of [8] gives a number of extensions of their original forward-secure scheme. In particular, they show (1) a modification of the scheme which can be proven secure in the standard model under the *decisional* BDH assumption; and (2) efficiency improvements which achieve key generation time and key update time $O(1)$. Both of these results may be carried over to our setting, via appropriate (small) modifications of the scheme presented above.

3.5 Security of Our Scheme

We now provide a sketch of the proof of security for the above scheme.

Theorem 1. *Under the computational BDH assumption (and in the random oracle model), the scheme described above is an intrusion-resilient public-key encryption scheme achieving semantic security.*

Proof. We convert any adversary A which successfully attacks the encryption scheme into an algorithm A' which breaks the BDH assumption. On a high level, A' will try to simulate the view of A in the following way: A' will guess the time period t^* for which A will ask its query to the LR oracle. If this guess turns out to be incorrect — in particular, as soon as A asks a query to LR which is *not* for time t^* or as soon as the scheme becomes (t^*, Q) -compromised — A' aborts the simulation and fails. On the other hand, if A' is correct in its guess (which occurs with probability $1/N$), then A' will be able to perfectly simulate the view of A in attacking the scheme. As in [8], this will enable A' to break the BDH assumption with probability $O(\text{Adv}_A(k)/Nq_{H_2})$, where q_{H_2} is the number of hash queries A makes to H_2 .

In more detail, adversary A' is given $(\mathbb{G}_1, \mathbb{G}_2, \hat{e})$ as output by $\mathcal{IG}(1^k)$, and is additionally given random elements $P, Q = s_\epsilon P, P' = bP$, and $U_0 = cP$. The goal of A' is to output $\hat{e}(P, P)^{s_\epsilon bc}$. A' will simulate an instance of the encryption scheme for adversary A . First, A' sets $PK = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, Q)$ and gives PK to A . Next, A' guesses a random index $t^* \in \{0, \dots, N - 1\}$ (this represents a guess of the period for which A will query LR). Let $\langle t^* \rangle = t_1^* \cdots t_\ell^*$ and $t_0^* = \epsilon$.

To answer the hash queries of A , algorithm A' maintains lists H_1^{list} and H_2^{list} . To begin, H_2^{list} will be empty. H_1^{list} is prepared by first having A' select random $x_2, \dots, x_\ell \in \mathbb{Z}_q$ and then storing the tuples (t_1^*, P') , $(t_1^* t_2^*, x_2 P), \dots, (t_1^* \cdots t_\ell^*, x_\ell P)$ in H_1^{list} . Next, A' proceeds as follows:

1. Choose random $y_1 \in \mathbb{Z}_q$ and store $(\overline{t_1^*}, y_1 P)$ in H_1^{list} .
2. For $2 \leq k \leq \ell$, choose random $y_k, s_k \in \mathbb{Z}_q$ and then store the value $(t_1^* \cdots t_{k-1}^* \overline{t_k^*}, y_k P - s_k^{-1} P')$ in H_1^{list} .

A' will respond to hash queries of A in the obvious way. If A queries $H_b(X)$, then A' checks whether there is a tuple of the form (X, Y) in H_b^{list} . If so, the value Y is returned. Otherwise, A' chooses random Y from the appropriate range, stores (X, Y) in H_b^{list} , and returns Y .

We point out that fixing the output of the random oracle as above will allow A' to simulate the “local secret keys” for all nodes in the tree (as in [14, 8]) *except* those nodes on the path ρ from the root to leaf t^* . In particular, fixing the output as above will allow A' to simulate local secret keys for all nodes whose parent is on path ρ ; A' can then generate “real” local secret keys for the remaining nodes by following the legal description of the scheme.

Since all other values stored by the user and by the base are (individually) random, it should be clear that A' can simulate all queries of A to Osec *except* those for which the scheme becomes (t^*, Q) -compromised (where Q now represents the queries of A to Osec up to and including that point in time).

When a query to **Osec** results in the scheme’s becoming (t^*, Q) -compromised, A' simply aborts as its guess of t^* was incorrect; this will occur exactly with probability $(N - 1)/N$. Assuming the scheme is never (t^*, Q) -compromised, A' can simulate the **LR** oracle as in [14, 8]. Specifically, A' will return the value $(U_0, x_2 U_0, \dots, x_\ell U_0, V)$, where V is a random element from $\{0, 1\}^n$. Overall, with probability $1/N$, A' will be able to simulate the entire view of A . It is easy to see that the only way for A to get any advantage in the above simulation is to ask the random oracle H_2 the value $\hat{e}(P, P)^{s_\epsilon b_c}$, as otherwise the encrypted message is information-theoretically hidden from A . Thus, outputting a random input element from the H_2^{list} will have non-negligible probability of being the value A' needs to break the BDH assumption.

3.6 Security against Adaptive Chosen-Ciphertext Attacks

We briefly state our main results, and defer the details until the final version of our paper. To benefit from a modular approach (i.e., to avoid having to re-prove security every time a new scheme is constructed), we would like to have a generic transformation for securing intrusion-resilient public-key encryption schemes against adaptive chosen-ciphertext attacks. Such a transformation would take *any* intrusion-resilient public-key encryption scheme achieving semantic security and convert it to an intrusion-resilient public-key encryption scheme secure against adaptive chosen-ciphertext attacks. We call such a transformation a *CCA2-transformation*.

Much work along these lines has been done for the case of “standard” public-key encryption (e.g., [13, 22] and others), and many CCA2-transformations for “standard” public-key encryption schemes are known. The next theorem shows that we may leverage off this work for the case of intrusion-resilient encryption. Namely:

Theorem 2. *Any CCA2-transformation for “standard” public-key encryption schemes is also a CCA2-transformation for intrusion-resilient public-key encryption schemes.*

We note that this is different from the case of forward-secure public-key encryption, where such a result does not seem to hold [8].

Applying, e.g., the CCA2-transformation of [13, 22] to our scheme above, we obtain an intrusion-resilient public-key encryption scheme secure against adaptive chosen-ciphertext attacks.

References

- [1] M. Abdalla, S. Miner, and C. Namprempre. Forward-Secure Threshold Signature Schemes. RSA 2001.
- [2] M. Abdalla and L. Reyzin. A New Forward-Secure Digital Signature Scheme. Asiacrypt 2000. 21
- [3] R. Anderson. Two Remarks on Public-Key Cryptology. Invited lecture, CCCS ’97. Available at <http://www.cl.cam.ac.uk/users/rja14/>. 19, 20

- [4] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A Concrete Security Treatment of Symmetric Encryption. FOCS '97. [24](#)
- [5] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. Crypto '99. [19](#), [20](#), [21](#)
- [6] M. Bellare and A. Palacio. Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold. Available at <http://eprint.iacr.org>. [21](#)
- [7] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. Crypto 2001. Full version to appear in *SIAM J. Computing* and available at <http://eprint.iacr.org/2001/090/>. [21](#), [25](#)
- [8] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. Preliminary version available at <http://eprint.iacr.org/2002/060/>. [19](#), [21](#), [22](#), [25](#), [26](#), [29](#), [30](#), [31](#)
- [9] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to Share a Function Securely. STOC '94. [20](#)
- [10] Y. Desmedt and Y. Frankel. Threshold Cryptosystems. Crypto '89. [20](#)
- [11] Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public-Key Cryptosystems. Eurocrypt 2002. [19](#), [20](#), [21](#)
- [12] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. PKC 2003. [20](#), [21](#)
- [13] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. Crypto '99. [22](#), [31](#)
- [14] C. Gentry and A. Silverberg. Hierarchical ID-Based Cryptography. Asiacrypt 2002. [25](#), [30](#), [31](#)
- [15] G. Itkis. Intrusion-Resilient Signatures: Generic Constructions, or Defeating a Strong Adversary with Minimal Assumptions. SCN 2002. [19](#), [21](#)
- [16] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. Crypto 2001. [21](#)
- [17] G. Itkis and L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. Crypto 2002. [19](#), [21](#), [22](#)
- [18] A. Joux. The Weil and Tate Pairing as Building Blocks for Public-Key Cryptosystems. ANTS 2002. [25](#)
- [19] A. Joux and K. Nguyen. Separating Decision Diffie-Hellman from Diffie-Hellman in Cryptographic Groups. Manuscript, Jan. 2001. Available at <http://eprint.iacr.org>. [25](#)
- [20] H. Krawczyk. Simple Forward-Secure Signatures From any Signature Scheme. CCCS 2000. [21](#)
- [21] T. Malkin, D. Micciancio, and S. Miner. Efficient Generic Forward-Secure Signatures with an Unbounded Number of Time Periods. Eurocrypt 2002. [21](#)
- [22] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-Security Asymmetric Cryptosystem Transform. CT-RSA 2001. [22](#), [31](#)
- [23] R. Ostrovsky and M. Yung. How to Withstand Mobile Virus Attacks. PODC '91. [21](#), [22](#)

TMAC: Two-Key CBC MAC

Kaoru Kurosawa and Tetsu Iwata

Department of Computer and Information Sciences
Ibaraki University

4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
{kurosawa,iwata}@cis.ibaraki.ac.jp

Abstract. In this paper, we propose TMAC. TMAC is a refinement of XCBC such that it requires only two keys while XCBC requires three keys. More precisely, TMAC requires only $(k + n)$ -bit keys while XCBC requires $(k + 2n)$ -bit keys, where k is the key length of the underlying block cipher E and n is its block length. We achieve this by using a universal hash function and the cost is almost negligible.

Similar to XCBC, the domain is $\{0, 1\}^*$ and it requires no extra invocation of E even if the size of the message is a multiple of n .

Keywords: CBC MAC, block cipher, provable security.

1 Introduction

1.1 Background

The CBC MAC is a well-known method to generate a message authentication code (MAC) based on a block cipher. Bellare, Kilian, and Rogaway proved the security of the CBC MAC for fixed message length mn , where n is the block length of the underlying block cipher E [3].

It is also known, however, that the CBC MAC is *not* secure unless the message length is fixed. For variable length messages, Encrypted MAC (EMAC) was developed for the RACE project [4]. EMAC is obtained by encrypting the CBC MAC value by E again with a new key K_2 . That is,

$$\text{EMAC}_{E_{K_1}, E_{K_2}}(M) = E_{K_2}(\text{CBC}_{E_{K_1}}(M)) ,$$

where M is a message and K_1 is the key of the CBC MAC and $\text{CBC}_{E_{K_1}}(M)$ is the CBC MAC for M . Petrank and Rackoff [12] proved that EMAC is secure if the message length is a multiple of n , that is, if the domain is $(\{0, 1\}^n)^+$. Then Vaudenay showed a significantly simpler proof by using decorrelation theory [13, 14].

The simplest approach to deal with arbitrary length messages is to append the minimal 10^i to a message M as a padding so that the length is a multiple of n . In this method, however, the padding is appended even if the size of the message is already a multiple of n . This means that the underlying block cipher E is invoked once more if the size of the message is a multiple of n .

Hence EMAC has two problems if the domain is $\{0, 1\}^*$. (We call this scheme EMAC * .)

1. E is invoked once more if the size of the message is a multiple of n .
2. It needs two key schedulings for two keys K_1 and K_2 .

Black and Rogaway solved the above two problems [5]. They first considered FCBC to solve the first problem. FCBC requires three block cipher keys K_1, K_2, K_3 , where K_1 is used to generate the CBC MAC, and K_2 or K_3 is used to encrypt the last block, depending on whether the size of the message is a multiple of n or not. See Fig. 7. They finally proposed XCBC to solve the second problem as well. XCBC is obtained from FCBC by replacing the last block cipher encryption by K_2 or K_3 with XORing K_2 or K_3 before the encryption. More precisely, XCBC is described as follows.

1. If $M \in (\{0, 1\})^+$ then XCBC computes exactly the same as the CBC MAC, except for XORing an n -bit key K_2 before encrypting the last block.
2. If $M \notin (\{0, 1\})^+$ then minimal 10^i padding ($i \geq 0$) is appended to M so that the length is a multiple of n , and XCBC computes exactly the same as the CBC MAC, except for XORing another n -bit key K_3 before encrypting the last block.

The key length of XCBC is $(k + 2n)$ bits in total, where k is the key length of the underlying block cipher E .

1.2 Our Contribution

In this paper, we propose TMAC, Two-Key CBC Message Authentication Code. TMAC is a refinement of XCBC such that it requires only two keys while XCBC requires three keys. More precisely, TMAC requires only $(k + n)$ -bit keys while XCBC requires $(k + 2n)$ -bit keys.

Similar to XCBC, the domain is $\{0, 1\}^*$ and it requires no extra invocation of E even if the size of the message is a multiple of n .

We achieve this by using a universal hash function and the cost is almost negligible. We also prove the security of TMAC formally.

Table 1 shows a comparison with CBC MAC and its variants, where M is the message and E is a block cipher. In Table 1,

- “# E Invocations” denotes the number of invocations of E , assuming $|M| > 0$.

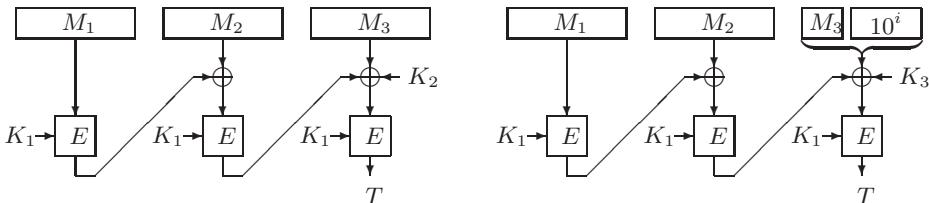


Fig. 1. Illustration of XCBC

- “ $\#E$ Keys” denotes the number of keys used for E , which is equivalent to the number of key schedulings of E .
- “ $\#M$ ” denotes the number of messages which the sender has maced.

1.3 Other Related Works

Jaulmes, Joux, and Valette proposed RMAC [11] which is an extension of EMAC. RMAC encrypts with $K_2 \oplus R$, where R is an n -bit random string and it is a part of the tag. That is,

$$\text{RMAC}_{E_{K_1}, E_{K_2}}(M) = (E_{K_2 \oplus R}(\text{CBC}_{E_{K_1}}(M)), R) .$$

They showed that the security of RMAC is beyond the birthday paradox limit. However, the tag is n bits longer than other CBC MAC variants. Also, for each message, an n -bit truly random string and one key scheduling are required.

Recently, some researchers proposed parallelizable MAC algorithms. Bellare, Guérin, and Rogaway proposed XOR MAC [2]. Gligor, and Donescu proposed XECB-MAC [9]. Black and Rogaway proposed PMAC [7].

2 Preliminaries

2.1 Notation

If A is a finite set then $\#A$ denotes the number of elements in A . For a set A , $x \xleftarrow{R} A$ means that x is randomly chosen from A . If $\alpha \in \{0, 1\}^*$ is a string then $|\alpha|$ denotes its length in bits. If $\alpha, \beta \in \{0, 1\}^*$ are equal-length strings then $\alpha \oplus \beta$ is their bitwise XOR.

2.2 CBC MAC

The block cipher E is a function $E : \mathcal{K}_E \times \{0, 1\}^n \rightarrow \{0, 1\}^n$, where each $E(K, \cdot) = E_K(\cdot)$ is a permutation on $\{0, 1\}^n$, \mathcal{K}_E is the set of possible keys and n is the block length.

Table 1. Comparison of CBC MAC and its variants

Name	Domain	# E Invocations	# E Keys	Key Length
CBC MAC [8, 10, 3]	$(\{0, 1\}^n)^m$	$ M /n$	1	k
EMAC* [4, 12, 13, 14]	$\{0, 1\}^*$	$1 + \lceil (M + 1)/n \rceil$	2	$2k$
RMAC [11]	$\{0, 1\}^*$	$1 + \lceil (M + 1)/n \rceil$	$1 + \#M$	$2k$
XCBC [5, 6]	$\{0, 1\}^*$	$\lceil M /n \rceil$	1	$k + 2n$
TMAC (Our proposal)	$\{0, 1\}^*$	$\lceil M /n \rceil$	1	$k + n$

The CBC MAC [8, 10] is the simplest and most well-known algorithm to make a MAC from a block cipher E . Let $M = M_1||M_2||\cdots||M_m$ be a message string such that $|M_1| = |M_2| = \cdots = |M_m| = n$. Then $\text{CBC}_{E_K}(M)$, the CBC MAC of M under key K , is defined as C_m , where

$$C_i = E_K(M_i \oplus C_{i-1})$$

for $i = 1, \dots, m$ and $C_0 = 0^n$.

Bellare, Kilian, and Rogaway proved the security of the CBC MAC for fixed message length mn [3]. It is well known, however, that the CBC MAC is *not* secure if the message length varies.

2.3 The Field with 2^n Points

For an n -bit string $\alpha = a_{n-1} \cdots a_1 a_0 \in \{0, 1\}^n$, let

$$\alpha \ll 1 = a_{n-2} a_{n-3} \cdots a_1 a_0 0 .$$

Similarly, let

$$\alpha \gg 1 = 0 a_{n-1} a_{n-2} \cdots a_2 a_1 .$$

We interchangeably think of a point a in $\text{GF}(2^n)$ in any of the following ways:

1. as an abstract point in a field;
2. as an n -bit string $a_{n-1} \cdots a_1 a_0 \in \{0, 1\}^n$;
3. as a formal polynomial $a(u) = a_{n-1}u^{n-1} + \cdots + a_1u + a_0$ with binary coefficients.

To add two points in $\text{GF}(2^n)$, take their bitwise XOR. We denote this operation by $a \oplus b$.

To multiply two points, fix some irreducible polynomial $f(u)$ having binary coefficients and degree n . To be concrete, choose the lexicographically first polynomial among the irreducible degree n polynomials having a minimum number of coefficients. We list some indicated polynomials.

$$\begin{cases} f(u) = u^{64} + u^4 + u^3 + u + 1 & \text{for } n = 64, \\ f(u) = u^{128} + u^7 + u^2 + u + 1 & \text{for } n = 128, \text{ and} \\ f(u) = u^{256} + u^{10} + u^5 + u^2 + 1 & \text{for } n = 256. \end{cases}$$

To multiply two points $a \in \text{GF}(2^n)$ and $b \in \text{GF}(2^n)$, regard a and b as polynomials $a(u) = a_{n-1}u^{n-1} + \cdots + a_1u + a_0$ and $b(u) = b_{n-1}u^{n-1} + \cdots + b_1u + b_0$, form their product $c(u)$ where one adds and multiplies coefficients in $\text{GF}(2)$, and take the remainder when dividing $c(u)$ by $f(u)$.

Note that it is particularly easy to multiply a point $a \in \{0, 1\}^n$ by u . We show a method for $n = 128$, where $f(u) = u^{128} + u^7 + u^2 + u + 1$. Then multiplying $a = a_{127} \cdots a_1 a_0$ by u yields a product $a_{n-1}u^n + a_{n-2}u^{n-1} + \cdots + a_1u^2 + a_0u$. Thus, if $a_{n-1} = 0$, then $a \cdot u = a \ll 1$. If $a_{n-1} = 1$, then we must add u^{128} to

$a \ll 1$. Since $u^{128} + u^7 + u^2 + u + 1 = 0$ we have $u^{128} = u^7 + u^2 + u + 1$, so adding u^{128} means to xor by $0^{120}10000111$. In summary, when $n = 128$,

$$a \cdot u = \begin{cases} a \ll 1 & \text{if } a_{127} = 0, \\ (a \ll 1) \oplus 0^{120}10000111 & \text{otherwise,} \end{cases} \quad (1)$$

where $a \cdot u = a(u) \cdot u \bmod f(u)$.

Also, note that it is easy to divide a point $a \in \{0, 1\}^n$ by u , meaning that one multiplies a by the multiplicative inverse of u in the field: $a \cdot u^{-1}$. We show a method for $n = 128$. Then multiplying $a = a_{127} \cdots a_1 a_0$ by u^{-1} yields a product $a_{n-1}u^{n-2} + a_{n-2}u^{n-3} + \cdots + a_2u + a_1 + a_0u^{-1}$. Thus, if $a_0 = 0$, then $a \cdot u^{-1} = a \gg 1$. If $a_0 = 1$, then we must add u^{-1} to $a \gg 1$. Since $u^{128} + u^7 + u^2 + u + 1 = 0$ we have $u^{127} = u^6 + u + 1 + u^{-1}$, so adding $u^{-1} = u^{127} + u^6 + u + 1$ means to xor by $10^{120}1000011$. In summary, when $n = 128$,

$$a \cdot u^{-1} = \begin{cases} a \gg 1 & \text{if } a_0 = 0, \\ (a \gg 1) \oplus 10^{120}1000011 & \text{otherwise.} \end{cases} \quad (2)$$

3 Basic Construction

3.1 Universal Hash Functions

Let H be a function $H : \mathcal{K}_H \times X \rightarrow \{0, 1\}^n$, where each $H(K, \cdot) = H_K(\cdot)$ is a function from X to $\{0, 1\}^n$, and \mathcal{K}_H is the set of possible keys of H .

Definition 3.1. We say $H : \mathcal{K}_H \times X \rightarrow \{0, 1\}^n$ is an (ϵ_1, ϵ_2) -almost XOR universal (AXU) hash function if

1. for any element $x \in X$ and any element $y \in \{0, 1\}^n$, there exist at most $\epsilon_1 \#\mathcal{K}_H$ keys $K \in \mathcal{K}_H$ such that $H_K(x) = y$, and
2. for any two distinct elements $x, x' \in X$ and any element $y \in \{0, 1\}^n$, there exist at most $\epsilon_2 \#\mathcal{K}_H$ keys $K \in \mathcal{K}_H$ such that $H_K(x) \oplus H_K(x') = y$.

We show an example. Let $\mathcal{K}_H = \{0, 1\}^n$, $X = \{0, 1\}^n \setminus \{0^n\}$, and $H_K(x) = K \cdot x$ over $\text{GF}(2^n)$. Then it is easy to see that H is a $(1/2^n, 1/2^n)$ -AXU hash function.

3.2 TMAC-Family

TMAC-family is defined by a block cipher E , an (ϵ_1, ϵ_2) -AXU hash function H and two distinct constants Cst_1 and Cst_2 , where Cst_1 and Cst_2 belong to the domain of H .

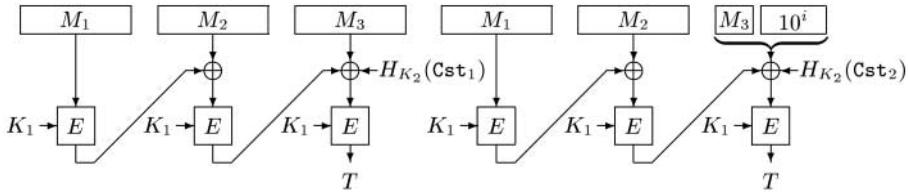
Let \mathcal{K}_E be the set of keys of E and let \mathcal{K}_H be the set of keys of H . Then the algorithm of TMAC-family is described in Fig. 2 and illustrated in Fig. 3. It takes two keys $K_1 \in \mathcal{K}_E$, $K_2 \in \mathcal{K}_H$ and a message $M \in \{0, 1\}^*$, and returns a string in $\{0, 1\}^n$. The key space \mathcal{K} is $\mathcal{K} = \mathcal{K}_E \times \mathcal{K}_H$.

The block cipher to be used is likely to be AES, but any other block cipher is fine.

```

Algorithm TMAC-fEK1, HK2, Cst1, Cst2(M)
if M ∈ {0, 1}n
    then K ← HK2(Cst1) and P ← M
    else K ← HK2(Cst2) and P ← M||10i, where i ← n − 1 − |M| mod n
Let P = P1||P2|| · · · ||Pm, where |P1| = |P2| = · · · = |Pm| = n
C0 ← 0n
for i ← 1 to m − 1 do
    Ci ← EK1(Pi ⊕ Ci−1)
return EK1(Pm ⊕ Cm−1 ⊕ K)

```

Fig. 2. Definition of TMAC-family**Fig. 3.** Illustration of TMAC-family

3.3 Comparison with XCBC

TMAC is obtained from XCBC by replacing (K_2, K_3) with $(H_{K_2}(\text{Cst}_1), H_{K_2}(\text{Cst}_2))$. Note that the number of keys is reduced from three (K_1, K_2, K_3) to two (K_1, K_2) .

4 Proposed Specification

In this section, we show the proposed specification of TMAC-family. Our choice is; $H_K(x) = K \cdot x$, $\text{Cst}_1 = u$, and $\text{Cst}_2 = 1$, where $K \in \{0, 1\}^n$ and \cdot denotes multiplication over $\text{GF}(2^n)$.

Equivalently $H_K(\text{Cst}_1) = K \cdot u$ and $H_K(\text{Cst}_2) = K$, where $K \cdot u$ can be computed with only one shift and one conditional XOR as shown in (1).

We call this algorithm TMAC specifically. The algorithm of TMAC is defined in Fig. 4 and illustrated in Fig. 5.

4.1 User Option

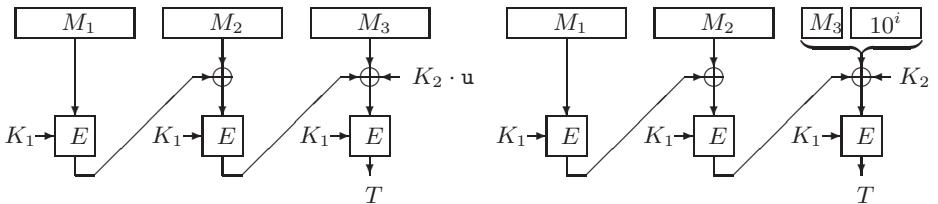
We have two options on the computation of $K_2 \cdot u$. The first option is to keep both K_2 and $K_2 \cdot u$ in the memory. It uses a memory of $2n$ bits.

The second option uses a memory of only n bits. We first keep K_2 in the memory. When $K_2 \cdot u$ is needed, we compute $K_2 \cdot u$ from K_2 . We then replace K_2 with $K_2 \cdot u$ in the memory. Next when K_2 is needed, we compute K_2 from $K_2 \cdot u$ and replace $K_2 \cdot u$ with K_2 in the memory. Repeat this process.

```

Algorithm TMACEK1, K2}(M)
if M ∈ ({0, 1}n)+
    then K ← K2 · u and P ← M
    else K ← K2 and P ← M||10i, where i ← n − 1 − |M| mod n
Let P = P1||P2||⋯||Pm, where |P1| = |P2| = ⋯ = |Pm| = n
C0 ← 0n
for i ← 1 to m − 1 do
    Ci ← EK1(Pi ⊕ Ci−1)
return T = EK1(Pm ⊕ Cm−1 ⊕ K)

```

Fig. 4. Definition of TMAC.**Fig. 5.** Illustration of TMAC

Note that it is easy to compute \$K_2\$ from \$K_2 \cdot u\$ since multiplication by \$u^{-1}\$ can be computed with only one shift and one conditional XOR as shown in (2).

4.2 Comparison with XCBC

TMAC is obtained from XCBC by replacing \$(K_2, K_3)\$ with \$(K_2 \cdot u, K_2)\$. The size of keys is reduced from \$(k + 2n)\$ bits to \$(k + n)\$ bits in this way.

5 Security of TMAC-Family

5.1 Security Definitions

An adversary \$\mathcal{A}\$ is an algorithm with an oracle (or oracles). The oracle computes some function. Without loss of generality, adversaries are assumed to never ask a query outside the domain of the oracle, and to never repeat a query.

A block cipher is a function \$E : \mathcal{K}_E \times \{0, 1\}^n \rightarrow \{0, 1\}^n\$ where \$\mathcal{K}_E\$ is a finite set and each \$E_K(\cdot) = E(K, \cdot)\$ is a permutation on \$\{0, 1\}^n\$. Let \$\text{Perm}(n)\$ denote the set of all permutations on \$\{0, 1\}^n\$. We say that \$P\$ is a random permutation if \$P\$ is randomly chosen from \$\text{Perm}(n)\$.

Note that \$\{E_K(\cdot) \mid K \in \mathcal{K}_E\}\$ should look like \$\text{Perm}(n)\$. For an adversary \$\mathcal{A}\$, we define

$$\text{Adv}_E^{\text{prp}}(\mathcal{A}) \stackrel{\text{def}}{=} \left| \Pr(K \xleftarrow{R} \mathcal{K}_E : \mathcal{A}^{E_K(\cdot)} = 1) - \Pr(P \xleftarrow{R} \text{Perm}(n) : \mathcal{A}^{P(\cdot)} = 1) \right| .$$

The adversary \mathcal{A} cannot distinguish $\{E_K(\cdot) \mid K \in \mathcal{K}_E\}$ from $\text{Perm}(n)$ if $\text{Adv}_E^{\text{prp}}(\mathcal{A})$ is negligible.

Similarly, a MAC function family from $\{0, 1\}^*$ to $\{0, 1\}^n$ is a map $F : \mathcal{K}_F \times \{0, 1\}^* \rightarrow \{0, 1\}^n$ where \mathcal{K}_F is a set with an associated distribution. We write $F_K(\cdot)$ for $F(K, \cdot)$. We say that $\mathcal{A}^{F_K(\cdot)}$ *forges* if \mathcal{A} outputs $(x, F_K(x))$ where \mathcal{A} never queried x to its oracle $F_K(\cdot)$. Then we define

$$\text{Adv}_F^{\text{mac}}(\mathcal{A}) \stackrel{\text{def}}{=} \Pr(K \xleftarrow{R} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot)} \text{ forges}) .$$

Let $\text{Rand}(*, n)$ denote the set of all functions from $\{0, 1\}^*$ to $\{0, 1\}^n$. This set is given a probability measure by asserting that a random element R of $\text{Rand}(*, n)$ associates to each string $x \in \{0, 1\}^*$ a random string $R(x) \in \{0, 1\}^n$. Then we define

$$\text{Adv}_F^{\text{viprf}}(\mathcal{A}) \stackrel{\text{def}}{=} \left| \Pr(K \xleftarrow{R} \mathcal{K}_F : \mathcal{A}^{F_K(\cdot)} = 1) - \Pr(R \xleftarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| .$$

Also we write

$$\text{Adv}_E^{\text{prp}}(t, q) \stackrel{\text{def}}{=} \max_{\mathcal{A}} \{\text{Adv}_E^{\text{prp}}(\mathcal{A})\},$$

where the maximum is over all adversaries who run in time at most t and make at most q queries. Further we write

$$\text{Adv}_F^{\text{mac}}(t, q, \mu) \stackrel{\text{def}}{=} \max_{\mathcal{A}} \{\text{Adv}_F^{\text{mac}}(\mathcal{A})\} \text{ and } \text{Adv}_F^{\text{viprf}}(t, q, \mu) \stackrel{\text{def}}{=} \max_{\mathcal{A}} \left\{ \text{Adv}_F^{\text{viprf}}(\mathcal{A}) \right\},$$

where the maximum is over all adversaries who run in time at most t , make at most q queries, each of which is at most μ -bits.

5.2 Theorem Statements

We give the following information-theoretic bound on the security of TMAC-family. A proof of this lemma is given in the next section.

We idealize a block cipher by a random permutation drawn from $\text{Perm}(n)$.

Lemma 5.1. *Suppose that a random permutation $P_1 \in \text{Perm}(n)$ is used in TMAC-family as the underlying block cipher. Let H be an (ϵ_1, ϵ_2) -AXU hash function, Cst_1 and Cst_2 be two distinct constants, where Cst_1 and Cst_2 belong to the domain of H . Let \mathcal{A} be an adversary which asks at most q queries, each of which is at most nm -bits. Assume $m \leq 2^n/4$. Then*

$$\left| \Pr(P_1 \xleftarrow{R} \text{Perm}(n); K_2 \xleftarrow{R} \mathcal{K}_H : \mathcal{A}^{\text{TMAC-f}_{P_1, H_{K_2}, \text{Cst}_1, \text{Cst}_2}(\cdot)} = 1) - \Pr(R \xleftarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| \leq \frac{(5m^2 + 2)q^2}{2 \cdot 2^n} + \frac{m^2q^2}{2} \epsilon ,$$

where $\epsilon = \max(\epsilon_1, \epsilon_2)$.

By substituting $\epsilon = \frac{1}{2^n}$, we obtain the following corollary.

Corollary 5.1. Suppose that a random permutation $P_1 \in \text{Perm}(n)$ is used in TMAC as the underlying block cipher. Let \mathcal{A} be an adversary which asks at most q queries, each of which is at most nm -bits. Assume $m \leq 2^n/4$. Then

$$\left| \Pr(P_1 \xleftarrow{R} \text{Perm}(n); K_2 \xleftarrow{R} \{0,1\}^n : \mathcal{A}^{\text{TMAC}_{P_1, K_2}(\cdot)} = 1) - \Pr(R \xleftarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| \leq \frac{(3m^2 + 1)q^2}{2^n} .$$

From the above corollary, it is standard to pass to the complexity-theoretic result. (For example, see [3, Section 3.2].) Then we have the following corollary.

Corollary 5.2. Let $E : \mathcal{K}_E \times \{0,1\}^n \rightarrow \{0,1\}^n$ be the underlying block cipher used in TMAC. Then

$$\text{Adv}_{\text{TMAC}}^{\text{viprf}}(t, q, nm) \leq \frac{(3m^2 + 1)q^2}{2^n} + \text{Adv}_E^{\text{prp}}(t', q') ,$$

where $t' = t + O(mq)$ and $q' = mq$.

The security of MAC is also derived in the usual way. (For example, see [3, Proposition 2.7].) Then we have the following theorem.

Theorem 5.1. Let $E : \mathcal{K}_E \times \{0,1\}^n \rightarrow \{0,1\}^n$ be the underlying block cipher used in TMAC. Then

$$\text{Adv}_{\text{TMAC}}^{\text{mac}}(t, q, nm) \leq \frac{(3m^2 + 1)q^2 + 1}{2^n} + \text{Adv}_E^{\text{prp}}(t', q') ,$$

where $t' = t + O(mq)$ and $q' = mq$.

5.3 Proof of Lemma 5.1

For a random permutation P , an (ϵ_1, ϵ_2) -AXU hash function H and two distinct constants Cst_1 and Cst_2 , where Cst_1 and Cst_2 belong to the domain of H , let

$$\begin{aligned} Q_1(x) &\stackrel{\text{def}}{=} P(H_K(\text{Cst}_1) \oplus x), \\ Q_2(x) &\stackrel{\text{def}}{=} P(H_K(\text{Cst}_2) \oplus x). \end{aligned}$$

We first show that $P(\cdot), Q_1(\cdot), Q_2(\cdot)$ are indistinguishable from three independent random permutations $P_1(\cdot), P_2(\cdot), P_3(\cdot)$.

Lemma 5.2. Let \mathcal{A} be an adversary which asks at most q queries. Then

$$\left| \Pr(P \xleftarrow{R} \text{Perm}(n); K \xleftarrow{R} \mathcal{K}_H : \mathcal{A}^{P(\cdot), P(H_K(\text{Cst}_1) \oplus \cdot), P(H_K(\text{Cst}_2) \oplus \cdot)} = 1) - \Pr(P_1, P_2, P_3 \xleftarrow{R} \text{Perm}(n) : \mathcal{A}^{P_1(\cdot), P_2(\cdot), P_3(\cdot)} = 1) \right| \leq \frac{q^2}{2} \left(\frac{1}{2^n} + \epsilon \right) ,$$

where $\epsilon = \max(\epsilon_1, \epsilon_2)$.

Algorithm $\text{FCBC}_{E_{K_1}, E_{K_2}, E_{K_3}}(M)$

if $M \in (\{0, 1\}^n)^+$

then $K \leftarrow K_2$, and $P \leftarrow M$

else $K \leftarrow K_3$, and $P \leftarrow M || 10^i$, where $i \leftarrow n - 1 - |M| \bmod n$

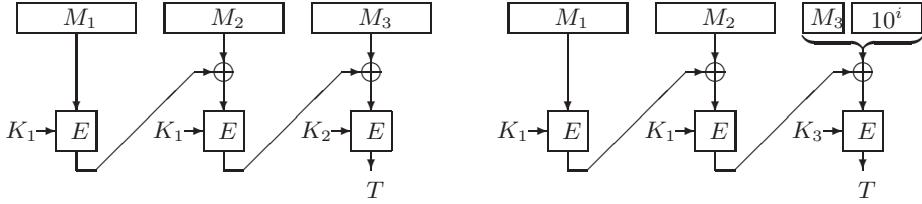
Let $P = P_1 || P_2 || \dots || P_m$, where $|P_1| = |P_2| = \dots = |P_m| = n$

$C_0 \leftarrow 0^n$

for $i \leftarrow 1$ **to** $m - 1$ **do**

$C_i \leftarrow E_{K_1}(P_i \oplus C_{i-1})$

return $E_K(P_m \oplus C_{m-1})$

Fig. 6. Definition of FCBC**Fig. 7.** Illustration of FCBC

A proof is given in the appendix.

Next we recall FCBC which appeared in the analysis of XCBC [5]. FCBC is a function taking three keys $K_1, K_2, K_3 \in \mathcal{K}_E$ and a message $M \in \{0, 1\}^*$, and returning a string in $\{0, 1\}^n$, where E is the underlying block cipher. The function is defined in Fig. 6 and illustrated in Fig. 7. Black and Rogaway showed the following result for FCBC [5].

Proposition 5.1 (Black and Rogaway [5]). *Suppose that random permutations $P_1, P_2, P_3 \in \text{Perm}(n)$ are used in FCBC as the underlying block ciphers. Let \mathcal{A} be an adversary which asks at most q queries, each of which is at most nm -bits. Assume $m \leq 2^n/4$. Then*

$$\left| \Pr(P_1, P_2, P_3 \xrightarrow{R} \text{Perm}(n) : \mathcal{A}^{\text{FCBC}_{P_1, P_2, P_3}(\cdot)} = 1) - \Pr(R \xrightarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| \leq \frac{(2m^2 + 1)q^2}{2^n} .$$

We finally give a proof of Lemma 5.1.

Proof (of Lemma 5.1). By the triangle inequality,

$$\left| \Pr(P_1 \xrightarrow{R} \text{Perm}(n); K_2 \xrightarrow{R} \mathcal{K}_H : \mathcal{A}^{\text{TMAC-f}_{P_1, H_{K_2}, \text{cst}_1, \text{cst}_2}(\cdot)} = 1) - \Pr(R \xrightarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| \quad (3)$$

is at most

$$\left| \Pr(P_1, P_2, P_3 \xrightarrow{R} \text{Perm}(n) : \mathcal{A}^{\text{FCBC}_{P_1, P_2, P_3}(\cdot)} = 1) - \Pr(R \xrightarrow{R} \text{Rand}(*, n) : \mathcal{A}^{R(\cdot)} = 1) \right| \quad (4)$$

$$+ \left| \Pr(P_1 \xleftarrow{R} \text{Perm}(n); K_2 \xleftarrow{R} \mathcal{K}_H : \mathcal{A}^{\text{TMAC-}\mathbf{f}_{P_1, H_{K_2}, \text{Cst}_1, \text{Cst}_2}(\cdot)} = 1) - \Pr(P_1, P_2, P_3 \xleftarrow{R} \text{Perm}(n) : \mathcal{A}^{\text{FCBC}_{P_1, P_2, P_3}(\cdot)} = 1) \right|. \quad (5)$$

Proposition 5.1 [5] gives us an upper bound on (4). We next bound (5). (5) is at most

$$\left| \Pr(P \xleftarrow{R} \text{Perm}(n); K \xleftarrow{R} \mathcal{K}_H : \mathcal{A}^{P(\cdot), P(H_K(\text{Cst}_1) \oplus \cdot), P(H_K(\text{Cst}_2) \oplus \cdot)} = 1) - \Pr(P_1, P_2, P_3 \xleftarrow{R} \text{Perm}(n) : \mathcal{A}^{P_1(\cdot), P_2(\cdot), P_3(\cdot)} = 1) \right| \quad (6)$$

since any adversary which does well in the setting (5) could be converted to one which does well in the setting (6), where we assume that \mathcal{A} in (6) makes at most mq total queries to her oracles. By applying Lemma 5.2, (5) is bounded by $\frac{m^2q^2}{2} \left(\frac{1}{2^n} + \epsilon \right)$. Therefore (3) is at most

$$\frac{(2m^2 + 1)q^2}{2^n} + \frac{m^2q^2}{2} \left(\frac{1}{2^n} + \epsilon \right) = \frac{(5m^2 + 2)q^2}{2 \cdot 2^n} + \frac{m^2q^2}{2} \epsilon.$$

□

6 Discussion

6.1 Advantages

Short Key. TMAC requires only $(k+n)$ -bit keys while XCBC uses $(k+2n)$ -bit keys.

Provable Security. We proved that TMAC is a variable input length ($\{0, 1\}^*$) pseudorandom function (VPRF) with fixed output length ($\{0, 1\}^n$) by assuming that the underlying block cipher is a pseudorandom permutation.

Efficiency. TMAC uses $\max\{1, \lceil |M|/n \rceil\}$ block cipher calls. The overhead beyond block cipher calls is almost negligible.

Arbitrarily Message Length. Any bit string $M \in \{0, 1\}^*$ can be computed, including the empty string. The length of the string need not be known in advance.

No Re-keying. Whereas some competing schemes (e.g., in [1, 4, 11]) would require invoking E with two or three different keys, TMAC requires only one key as XCBC. Therefore any key scheduling costs are minimized. This enhances efficiency in both software and hardware.

No decryption. As for any CBC MAC variant, TMAC does not use decryption of the block cipher.

Backwards Compatibility. TMAC with $K_2 = 0^n$ is backwards compatible with the CBC MAC.

Simplicity. Because TMAC is simple, it is easily implemented in both software and hardware.

6.2 Limitations

We note the following limitations. They apply to any CBC MAC variants and therefore none of them is specific to TMAC.

Sequential Block Cipher Calls. The CBC MAC and its variants, including TMAC, are not parallelizable.

Limited Pre-processing Capability. Key-scheduling of the underlying block cipher and $K_2 \cdot u$ can be pre-computed. Additional pre-computation is not possible without knowing the message.

7 Conclusion

Similar to XCBC, TMAC uses $\lceil |M|/n \rceil$ block cipher invocations for any non-empty message M . (The empty string is an exception; it requires one block cipher invocation.) Overhead beyond block cipher calls is almost negligible.

The size of secret keys is n bits smaller than XCBC. The cost for this short key is to use $K_2 \cdot u$. It is computed with only one shift and one conditional XOR.

References

- [1] ANSI X9.19. American national standard — Financial institution retail message authentication. ASC X9 Secretariat — American Bankers Association, 1986. [43](#)
- [2] M. Bellare, R. Guérin, and P. Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. *Advances in Cryptology — CRYPTO '95, LNCS 963*, pp. 15–28, Springer-Verlag, 1995. [35](#)
- [3] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *JCSS*, vol. 61, no. 3, 2000. Earlier version in *Advances in Cryptology — CRYPTO '94, LNCS 839*, pp. 341–358, Springer-Verlag, 1994. [33, 35, 36, 41](#)
- [4] A. Berendschot, B. den Boer, J. P. Boly, A. Bosselaers, J. Brandt, D. Chaum, I. Damgård, M. Dichtl, W. Fumy, M. van der Ham, C. J. A. Jansen, P. Landrock, B. Preneel, G. Roelofsen, P. de Rooij, and J. Vandewalle. Final Report of RACE Integrity Primitives. *LNCS 1007*, Springer-Verlag, 1995. [33, 35, 43](#)
- [5] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three key constructions. *Advances in Cryptology — CRYPTO 2000, LNCS 1880*, pp. 197–215, Springer-Verlag, 2000. [34, 35, 42, 43](#)
- [6] J. Black and P. Rogaway. Comments to NIST concerning AES modes of operations: A suggestion for handling arbitrary-length messages with the CBC MAC. *Second Modes of Operation Workshop*. Available at <http://www.cs.ucdavis.edu/~rogaway/>. [35](#)
- [7] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. *Advances in Cryptology — EUROCRYPT 2002, LNCS 2332*, pp. 384–397, Springer-Verlag, 2002. [35](#)
- [8] FIPS 113. Computer data authentication. Federal Information Processing Standards Publication 113, U. S. Department of Commerce/National Bureau of Standards, National Technical Information Service, Springfield, Virginia, 1994. [35, 36](#)

- [9] V. Gligor, and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. *Fast Software Encryption, FSE 2001, LNCS 2355*, pp. 92–108, Springer-Verlag, 2001. Full version is available at <http://csrc.nist.gov/encryption/modes/proposedmodes/>. 35
- [10] ISO/IEC 9797-1. Information technology — security techniques — data integrity mechanism using a cryptographic check function employing a block cipher algorithm. International Organization for Standards, Geneva, Switzerland, 1999. Second edition. 35, 36
- [11] É. Jaulmes, A. Joux, and F. Valette. On the security of randomized CBC-MAC beyond the birthday paradox limit: A new construction. *Fast Software Encryption, FSE 2002, LNCS 2365*, pp. 237–251, Springer-Verlag, 2002. Full version is available at <http://eprint.iacr.org/2001/074/>. 35, 43
- [12] E. Petrank and C. Rackoff. CBC MAC for real-time data sources. *J.Cryptology*, vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000. 33, 35
- [13] S. Vaudenay. Decorrelation over infinite domains: the encrypted CBC-MAC Case. *Selected Areas in Cryptography, SAC 2000, LNCS 2012*, pp. 57–71, Springer-Verlag, 2001. 33, 35
- [14] S. Vaudenay. Decorrelation over infinite domains: the encrypted CBC-MAC Case. *Communications in Information and Systems (CIS)*, vol. 1, pp. 75–85, 2001. 33, 35

A Proof of Lemma 5.2

Let $\{A^{(1)}, \dots, A^{(q)}\}$ be a set of n -bit strings, that is, $A^{(i)} \in \{0,1\}^n$ for $1 \leq \forall i \leq q$. We say $\{A^{(1)}, \dots, A^{(q)}\}$ are distinct as shorthand for $A^{(i)} \neq A^{(j)}$ for $1 \leq \forall i < \forall j \leq q$.

Before proving Lemma 5.2, we need the following lemma.

Lemma A.1. *Let q_1, q_2, q_3 be non-negative integers. Let*

$$x_1^{(1)}, \dots, x_1^{(q_1)}, x_2^{(1)}, \dots, x_2^{(q_2)}, x_3^{(1)}, \dots, x_3^{(q_3)}$$

be fixed n -bit strings such that $\{x_1^{(1)}, \dots, x_1^{(q_1)}\}$ are distinct, $\{x_2^{(1)}, \dots, x_2^{(q_2)}\}$ are distinct, and $\{x_3^{(1)}, \dots, x_3^{(q_3)}\}$ are distinct. Similarly, let

$$y_1^{(1)}, \dots, y_1^{(q_1)}, y_2^{(1)}, \dots, y_2^{(q_2)}, y_3^{(1)}, \dots, y_3^{(q_3)}$$

be fixed n -bit strings such that $\{y_1^{(1)}, \dots, y_1^{(q_1)}, y_2^{(1)}, \dots, y_2^{(q_2)}, y_3^{(1)}, \dots, y_3^{(q_3)}\}$ are distinct. Let $H : \mathcal{K}_H \times \{0,1\}^n \rightarrow \{0,1\}^n$ be an (ϵ_1, ϵ_2) -AXU hash function, and let Cst_1 and Cst_2 be two distinct constants, where Cst_1 and Cst_2 belong to the domain of H . Let $P \in \text{Perm}(n)$ and $K \in \mathcal{K}_H$. Then the number of (P, K) which satisfies

$$\begin{cases} P(x_1^{(i)}) = y_1^{(i)} & \text{for } 1 \leq \forall i \leq q_1, \\ P(H_K(\text{Cst}_1) \oplus x_2^{(i)}) = y_2^{(i)} & \text{for } 1 \leq \forall i \leq q_2, \text{ and} \\ P(H_K(\text{Cst}_2) \oplus x_3^{(i)}) = y_3^{(i)} & \text{for } 1 \leq \forall i \leq q_3 \end{cases} \quad (7)$$

is at least $(2^n - (q_1 + q_2 + q_3))! \cdot \#\mathcal{K}_H(1 - \epsilon(q_1 q_2 + q_1 q_3 + q_2 q_3))$, where $\epsilon = \max(\epsilon_1, \epsilon_2)$.

Let $q = q_1 + q_2 + q_3$. Then the above bound is at least $(2^n - q)! \cdot \#\mathcal{K}_H \left(1 - \frac{\epsilon q^2}{2}\right)$ since $q_1 q_2 + q_1 q_3 + q_2 q_3 = \frac{q^2 - q_1^2 - q_2^2 - q_3^2}{2}$.

Proof (of Lemma A.1). We first count the number of K .

Number of K . First, for any fixed i and j such that $1 \leq i \leq q_1$ and $1 \leq j \leq q_2$, we have at most $\epsilon_1 \#\mathcal{K}_H$ keys K such that $x_1^{(i)} = H_K(\text{Cst}_1) \oplus x_2^{(j)}$. Since there are $q_1 q_2$ choice of (i, j) , we have

$$\#\{K \mid x_1^{(i)} = H_K(\text{Cst}_1) \oplus x_2^{(j)} \text{ for } 1 \leq \exists i \leq q_1 \text{ and } 1 \leq \exists j \leq q_2\} \leq q_1 q_2 \epsilon_1 \#\mathcal{K}_H. \quad (8)$$

Next, for any fixed i and j such that $1 \leq i \leq q_1$ and $1 \leq j \leq q_3$, we have at most $\epsilon_1 \#\mathcal{K}_H$ keys K such that $x_1^{(i)} = H_K(\text{Cst}_2) \oplus x_3^{(j)}$. Since there are $q_1 q_3$ choice of (i, j) , we have

$$\#\{K \mid x_1^{(i)} = H_K(\text{Cst}_2) \oplus x_3^{(j)} \text{ for } 1 \leq \exists i \leq q_1 \text{ and } 1 \leq \exists j \leq q_3\} \leq q_1 q_3 \epsilon_1 \#\mathcal{K}_H. \quad (9)$$

Next, for any fixed i and j such that $1 \leq i \leq q_2$ and $1 \leq j \leq q_3$, we have at most $\epsilon_2 \#\mathcal{K}_H$ keys K such that $H_K(\text{Cst}_1) \oplus x_2^{(i)} = H_K(\text{Cst}_2) \oplus x_3^{(j)}$. Since there are $q_2 q_3$ choice of (i, j) , we have

$$\begin{aligned} \#\{K \mid H_K(\text{Cst}_1) \oplus x_2^{(i)} = H_K(\text{Cst}_2) \oplus x_3^{(j)} \text{ for } 1 \leq \exists i \leq q_2 \text{ and } 1 \leq \exists j \leq q_3\} \\ \leq q_2 q_3 \epsilon_2 \#\mathcal{K}_H. \end{aligned} \quad (10)$$

Then from (8), (9) and (10), we have at least $\#\mathcal{K}_H - \#\mathcal{K}_H(q_1 q_2 \epsilon_1 + q_1 q_3 \epsilon_1 + q_2 q_3 \epsilon_2) \geq \#\mathcal{K}_H(1 - \epsilon(q_1 q_2 + q_1 q_3 + q_2 q_3))$ choice of $K \in \mathcal{K}_H$ which satisfies the following three conditions:

$$\begin{cases} x_1^{(i)} \neq H_K(\text{Cst}_1) \oplus x_2^{(j)} & \text{for } 1 \leq \forall i \leq q_1 \text{ and } 1 \leq \forall j \leq q_2, \\ x_1^{(i)} \neq H_K(\text{Cst}_2) \oplus x_3^{(j)} & \text{for } 1 \leq \forall i \leq q_1 \text{ and } 1 \leq \forall j \leq q_3, \text{ and} \\ H_K(\text{Cst}_1) \oplus x_2^{(i)} \neq H_K(\text{Cst}_2) \oplus x_3^{(j)} & \text{for } 1 \leq \forall i \leq q_2 \text{ and } 1 \leq \forall j \leq q_3. \end{cases}$$

We now fix any K which satisfies these three conditions.

Number of P . Now K is fixed in such a way that

$$\begin{aligned} &\{x_1^{(1)}, \dots, x_1^{(q_1)}, \\ &H_K(\text{Cst}_1) \oplus x_2^{(1)}, \dots, H_K(\text{Cst}_1) \oplus x_2^{(q_2)}, \\ &H_K(\text{Cst}_2) \oplus x_3^{(1)}, \dots, H_K(\text{Cst}_2) \oplus x_3^{(q_3)}\} \end{aligned}$$

(which are inputs to P) are distinct. Also, the corresponding outputs

$$\begin{aligned} &\{y_1^{(1)}, \dots, y_1^{(q_1)}, \\ &y_2^{(1)}, \dots, y_2^{(q_2)}, \\ &y_3^{(1)}, \dots, y_3^{(q_3)}\} \end{aligned}$$

are distinct. In other words, for P , the above $q_1 + q_2 + q_3$ input-output pairs are determined. The remaining $2^n - (q_1 + q_2 + q_3)$ input-output pairs are undetermined. Therefore we have $(2^n - (q_1 + q_2 + q_3))!$ possible choice of P for any such fixed K .

This concludes the proof of the lemma. \square

We now prove Lemma 5.2.

Proof (of Lemma 5.2). Let $\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3$ be either $P_1(\cdot), P_2(\cdot), P_3(\cdot)$ or $P(\cdot), P(H_K(\text{Cst}_1) \oplus \cdot), P(H_K(\text{Cst}_2) \oplus \cdot)$. The adversary \mathcal{A} has oracle access to $\mathcal{O}_1, \mathcal{O}_2$ and \mathcal{O}_3 .

Since \mathcal{A} is computationally unbounded, there is no loss of generality to assume that \mathcal{A} is deterministic.

There are three types of queries \mathcal{A} can make: either $(1, x)$ which denotes the query “what is $\mathcal{O}_1(x)?$,” $(2, x)$ which denotes the query “what is $\mathcal{O}_2(x)?$,” or $(3, x)$ which denotes the query “what is $\mathcal{O}_3(x)?$.” For the i -th query \mathcal{A} makes to \mathcal{O}_j , define the query-answer pair $(x_j^{(i)}, y_j^{(i)}) \in \{0, 1\}^n \times \{0, 1\}^n$, where \mathcal{A} ’s query was $(j, x_j^{(i)})$ and the answer it got was $y_j^{(i)}$.

Suppose that we run \mathcal{A} with oracles. For this run, assume that \mathcal{A} made q_1 queries to $\mathcal{O}_1(x)$, q_2 queries to $\mathcal{O}_2(x)$, and q_3 queries to $\mathcal{O}_3(x)$, where $q_1 + q_2 + q_3 = q$. For this run, we define view v of \mathcal{A} as

$$v \stackrel{\text{def}}{=} \langle (y_1^{(1)}, \dots, y_1^{(q_1)}), (y_2^{(1)}, \dots, y_2^{(q_2)}), (y_3^{(1)}, \dots, y_3^{(q_3)}) \rangle . \quad (11)$$

For this view, we always have the following three conditions:

$$\begin{cases} \{y_1^{(1)}, \dots, y_1^{(q_1)}\} \text{ are distinct,} \\ \{y_2^{(1)}, \dots, y_2^{(q_2)}\} \text{ are distinct, and} \\ \{y_3^{(1)}, \dots, y_3^{(q_3)}\} \text{ are distinct.} \end{cases}$$

We note that since \mathcal{A} never repeats a query, for the corresponding queries, we have

$$\begin{cases} \{x_1^{(1)}, \dots, x_1^{(q_1)}\} \text{ are distinct,} \\ \{x_2^{(1)}, \dots, x_2^{(q_2)}\} \text{ are distinct, and} \\ \{x_3^{(1)}, \dots, x_3^{(q_3)}\} \text{ are distinct.} \end{cases}$$

Since \mathcal{A} is deterministic, the i -th query \mathcal{A} makes is fully determined by the first $i - 1$ query-answer pairs. This implies that if we fix some qn -bit string V and return the i -th n -bit block as the answer for the i -th query \mathcal{A} makes (instead of the oracles), then

- \mathcal{A} ’s queries are uniquely determined,
- q_1, q_2 and q_3 are uniquely determined,
- the parsing of V into the format defined in (11) is unique determined, and
- the final output of \mathcal{A} (0 or 1) is uniquely determined.

Let \mathbf{V}_{one} be a set of all qn -bit string V such that \mathcal{A} outputs 1. We let $N_{one} \stackrel{\text{def}}{=} \#\mathbf{V}_{one}$. Also, let \mathbf{V}_{good} be a set of all qn -bit string V such that for the corresponding parsing v , we have $\{y_1^{(1)}, \dots, y_1^{(q_1)}, y_2^{(1)}, \dots, y_2^{(q_2)}, y_3^{(1)}, \dots, y_3^{(q_3)}\}$ are distinct. We let $N_{good} \stackrel{\text{def}}{=} \#\mathbf{V}_{good}$. Note that $N_{good} = \frac{(2^n)!}{(2^n-q)!}$. Now observe that the number of V which is *not* in the set \mathbf{V}_{good} is at most $\binom{q}{2} \frac{2^{qn}}{2^n}$. Therefore, we have

$$\#\{V \mid V \in (\mathbf{V}_{one} \cap \mathbf{V}_{good})\} \geq N_{one} - \binom{q}{2} \frac{2^{qn}}{2^n}. \quad (12)$$

Evaluation of p_{rand} . We first evaluate

$$\begin{aligned} p_{rand} &\stackrel{\text{def}}{=} \Pr(P_1, P_2, P_3 \xleftarrow{R} \text{Perm}(n) : \mathcal{A}^{P_1(\cdot), P_2(\cdot), P_3(\cdot)} = 1) \\ &= \frac{\#\{(P_1, P_2, P_3) \mid \mathcal{A}^{P_1(\cdot), P_2(\cdot), P_3(\cdot)} = 1\}}{\{(2^n)!\}^3}. \end{aligned}$$

For each $V \in \mathbf{V}_{one}$, the number of (P_1, P_2, P_3) such that

$$\begin{cases} P_1(x_1^{(i)}) = y_1^{(i)} & \text{for } 1 \leq \forall i \leq q_1, \\ P_2(x_2^{(i)}) = y_2^{(i)} & \text{for } 1 \leq \forall i \leq q_2, \text{ and} \\ P_3(x_3^{(i)}) = y_3^{(i)} & \text{for } 1 \leq \forall i \leq q_3 \end{cases} \quad (13)$$

is exactly $(2^n - q_1)! \cdot (2^n - q_2)! \cdot (2^n - q_3)!$, which is at most $(2^n - q)! \cdot \{(2^n)!\}^2$. Therefore, we have

$$\begin{aligned} p_{rand} &= \sum_{V \in \mathbf{V}_{one}} \frac{\#\{(P_1, P_2, P_3) \mid (P_1, P_2, P_3) \text{ satisfying (13)}\}}{\{(2^n)!\}^3} \\ &\leq \sum_{V \in \mathbf{V}_{one}} \frac{(2^n - q)!}{(2^n)!} \\ &= N_{one} \cdot \frac{(2^n - q)!}{(2^n)!}. \end{aligned}$$

Evaluation of p_{real} . We next evaluate

$$\begin{aligned} p_{real} &\stackrel{\text{def}}{=} \Pr(P \xleftarrow{R} \text{Perm}(n); K \xleftarrow{R} \mathcal{K}_H : \mathcal{A}^{P(\cdot), P(H_K(\text{Cst}_1) \oplus \cdot), P(H_K(\text{Cst}_2) \oplus \cdot)} = 1) \\ &= \frac{\#\{(P, K) \mid \mathcal{A}^{P(\cdot), P(H_K(\text{Cst}_1) \oplus \cdot), P(H_K(\text{Cst}_2) \oplus \cdot)} = 1\}}{(2^n)! \cdot \#\mathcal{K}_H}. \end{aligned}$$

Then from Lemma A.1, we have

$$\begin{aligned} p_{real} &\geq \sum_{V \in (\mathbf{V}_{one} \cap \mathbf{V}_{good})} \frac{\#\{(P, K) \mid (P, K) \text{ satisfying (7)}\}}{(2^n)! \cdot \#\mathcal{K}_H} \\ &\geq \sum_{V \in (\mathbf{V}_{one} \cap \mathbf{V}_{good})} \frac{(2^n - q)!}{(2^n)!} \cdot \left(1 - \frac{\epsilon q^2}{2}\right). \end{aligned}$$

Completing the Proof. From (12) we have

$$\begin{aligned} p_{real} &\geq \left(N_{one} - \binom{q}{2} \frac{2^{qn}}{2^n} \right) \cdot \frac{(2^n - q)!}{(2^n)!} \cdot \left(1 - \frac{\epsilon q^2}{2} \right) \\ &= \left(N_{one} \cdot \frac{(2^n - q)!}{(2^n)!} - \binom{q}{2} \frac{2^{qn}}{2^n} \cdot \frac{(2^n - q)!}{(2^n)!} \right) \cdot \left(1 - \frac{\epsilon q^2}{2} \right) \\ &\geq \left(p_{rand} - \binom{q}{2} \frac{2^{qn}}{2^n} \cdot \frac{(2^n - q)!}{(2^n)!} \right) \cdot \left(1 - \frac{\epsilon q^2}{2} \right). \end{aligned}$$

Since $2^{qn} \cdot \frac{(2^n - q)!}{(2^n)!} \geq 1$, we have

$$\begin{aligned} p_{real} &\geq \left(p_{rand} - \frac{q(q-1)}{2 \cdot 2^n} \right) \cdot \left(1 - \frac{\epsilon q^2}{2} \right) \\ &\geq p_{rand} - \frac{q^2}{2} \left(\frac{1}{2^n} + \epsilon \right). \end{aligned} \tag{14}$$

Applying the same argument to $1 - p_{real}$ and $1 - p_{rand}$ yields that

$$1 - p_{real} \geq 1 - p_{rand} - \frac{q^2}{2} \left(\frac{1}{2^n} + \epsilon \right). \tag{15}$$

Finally, (14) and (15) give $|p_{real} - p_{rand}| \leq \frac{q^2}{2} \left(\frac{1}{2^n} + \epsilon \right)$. \square

Montgomery Prime Hashing for Message Authentication

Douglas L. Whiting¹ and Michael J. Sabin²

¹ Hifn, Carlsbad CA, USA

dwhiting@hifn.com

² Independent Consultant, Sunnyvale CA, USA

mike.sabin@att.net

Abstract. Montgomery Prime Hashing (MPH) is a scheme for message authentication based on universal hashing. In MPH, roughly speaking, the hash value is computed as the Montgomery residue of the message with respect to a secret modulus. The modulus value is structured in a way that allows fast, compact implementations in both hardware and software. The set of allowed modulus values is large, and as a result, MPH achieves good, provable security.

MPH performance is comparable to that of other high-speed schemes such as MMH. An advantage of MPH is that the secret key (i.e., the modulus) is small, typically 128–256 bits, while in MMH the secret key is typically much larger. In applications where MMH key length is problematic, MPH may be an attractive alternative.

Keywords: Universal hash, message authentication, MAC.

1 Introduction

This work was motivated by the need to find a message authentication check (MAC) algorithm for use in wireless LAN (WLAN) applications [6]. A MAC is a “secure checksum,” i.e., a string of bits appended to a message that enables a receiver to verify that an incoming message is authentic and has not been altered en route. The MAC is based on a secret key (the MAC key) shared by sender and receiver. With high probability, an attacker who does not know the MAC key cannot create the correct MAC value for any message; that is, an attacker cannot forge a message.

In WLANs, the main design considerations for the MAC algorithm are as follows:

- *Short messages.* Message lengths range from 64 to 1,500 bytes. The MAC computation must be well suited to this range.
- *Efficient in software and hardware.* WLAN equipment comes in a wide variety of implementations from many different vendors. Some implementations compute the MAC in software, either on a general-purpose computer or on an embedded CPU. Other implementations compute the MAC in hardware using custom silicon. Accordingly, the MAC computation must be efficient across a wide variety of platforms, in both software and hardware.

- *MAC computation more critical than MAC key derivation.* The MAC computation is critical: it is performed on each message sent or received, so it must be fast and simple in order to allow low cost, high-throughput implementations. The MAC key-derivation procedure is less critical: it is performed infrequently, as part of a relatively slow, high-level protocol. We are willing to make reasonable tradeoffs that improve the MAC computation’s efficiency at the expense of the key derivation. Unlike the MAC computation, the key-derivation algorithm is intended only for software; no hardware considerations apply.

These considerations are not unique to WLANs. Thus, the MAC algorithm we propose here is not specific to WLANs and may be useful in other applications with comparable considerations.

Our proposed MAC algorithm is based on universal hashing, an approach first developed by Carter and Wegman [3]. A family of universal hash functions is a collection of functions that map messages of arbitrary length into hash values of a small, fixed length. The family is characterized by a collision-rate bound, i.e., a bound on the probability that two messages produce the same hash value when a function is selected at random from the family. To authenticate a message, a hash function is selected from the family, with the selection determined by a secret key (the MAC key) shared by sender and receiver. The sender computes the hash of the message using the selected function, encrypts the resulting hash value, and sends the encrypted hash value (the MAC) along with the message. The receiver computes the hash value of the received message and verifies that it matches the received hash value. The collision-rate bound of the hash family results in a provable security bound, i.e., a bound on the probability that an attacker can forge a message. The security bound does not depend on the probability distribution of the messages being authenticated; it depends only on the distribution used to select a hash function from the family.

An example of message authentication based on universal hashing is MMH [5]. In MMH, roughly speaking, the hash function is computed as an inner product $\sum_i m_i x_i$, where $\{m_i\}$ are the words of the message and $\{x_i\}$ are the words of the MAC key. There are additional details we need not mention here. MMH is well-suited to both hardware and software. It is cleverly designed so that the time-critical part of the computation is an integer multiply/accumulate operation. With 32-bit words, [5] reports a throughput of 1.5 clocks per byte (cpb) on a Pentium-Pro machine, with a security bound of $2^{-29.4}$. For 64-bit words, the numbers are 3.2 cpb and $2^{-61.4}$, respectively. These are impressive numbers, and as a result, MMH is finding its way into various applications. One such application is Packetcable [2], which uses 32-bit MMH to authenticate voice-over-IP packets.

A disadvantage of MMH is that, in its simplest form, it requires the MAC key to be the length of the longest allowed message. In the case of WLANs, this means the MAC key must be 1,500 bytes long. This can be problematic for some devices, such as access points, that maintain security associations with hundreds or thousands of peers. Each security association requires a unique MAC key; at

1,500 bytes per key, this can add up to a lot of storage. There is a further complication. Each time the access point sends or receives a message, it must fetch the MAC key of the corresponding security association. When the MAC key is as long as the message, the need to fetch it effectively doubles the bus bandwidth needed by the MAC-computation device to maintain real-time throughput. On a CPU with high bus bandwidth, such as a Pentium, this is of little concern. But on simpler, embedded CPUs, and in hardware implementations, this fetching can be a significant penalty.

One workaround for the long-key issue in MMH is to use tree hashing. For example, with 32-bit MMH on a 1,500-byte message, a two-level tree requires a key size of about 155 bytes, and a three-level tree requires a key size of about 87 bytes. Using more than three levels reduces the key size further, but not by much. These key sizes are much smaller than 1,500 bytes, but they may still be long enough to be of concern in some devices. Furthermore, tree hashing adds significant complexity to the implementation, particularly in hardware.

Another workaround is to use a pseudorandom generator to expand a short key into a long one, with the expansion occurring in real time as a message's MAC is computed. However, on many CPUs, a suitable pseudorandom generator consumes significantly more cycles than MMH itself, swamping out MMH's good performance.

UMAC [1, 14] is a variation of MMH that achieves very high throughput when authenticating large messages in software. While UMAC appears to be the current performance champion for large messages, it is less well suited to short messages. For a message length of 1,500 bytes, on a Pentium III machine, [14] reports a throughput of 0.6 cpb for the 32-bit case and 1.1 cpb for the 64-bit case. These are good numbers relative to MMH. But for a message length of 43 bytes, the throughput numbers degrade to 4.9 cpb and 7.7 cpb, respectively; these are much slower than MMH. Numbers for a message length of 64 bytes are not given in [14]; we assume they are in the same ballpark as those for 43 bytes. To achieve these numbers, UMAC uses a key length in excess of 1,024 bytes. UMAC uses three levels of hashing, each level using a different algorithm; relative to MMH, this is much more complex to implement, particularly in hardware. In short, while UMAC is a superb performer in many applications, it does not appear well suited to WLANs, due to its complexity, long key size, and inefficiency on short messages.

In this paper, we propose Montgomery Prime Hashing (MPH) for message authentication. MPH is efficient in both hardware and software. In MPH, roughly speaking, the hash function is computed as the Montgomery residue of the message with respect to a secret modulus (the MAC key). The modulus value is constrained to have the structure needed for a fast implementation, but this still results in hash family that is large enough to achieve a good security bound. In the 32-bit case, we achieve a throughput of 1.9 cpb on a Pentium Pro or Pentium III machine using a key size of 16 bytes; in the 64-bit case, we achieve a throughput of 3.2 cpb using a key size of 32 bytes. These throughput numbers are slightly worse than those of MMH, but the key-size numbers are much more favorable.

The security bound achieved using MPH degrades as the length of the message increases. In this sense it is similar to polynomial hashing (i.e., algorithm ‘‘PolyCW’’ of [8, 9]) and to Krawczyk’s LFSR hash [7]. For the worst-case WLAN message length of 1,500 bytes, the security bound of MPH is $2^{-23.4}$ for the 32-bit case and $2^{-55.4}$ for the 64-bit case — about 6 bits less security than the corresponding numbers for MMH. We believe the reduced security is a desirable tradeoff in order to get the reduced key length.

For MPH, the key-derivation procedure consists of selecting a secret modulus. The structure we impose on the modulus includes the requirement that it have a large prime factor. This makes selecting an MPH key much slower than selecting an MMH key. However, as mentioned above, the run-time of the key-derivation procedure is only a secondary concern. We present a key-derivation algorithm that is optimum in the security sense, i.e., an algorithm that selects the modulus with the precise distribution that achieves the claimed security bounds. The algorithm is also reasonably fast; when coded in C on an 800 MHz Pentium III, the average run time was 0.3 msec for the 32-bit case and 1.3 msec for the 64-bit case.

Related Work. Message authentication based on universal hashing has received a lot of attention. See, e.g., [1, 5, 8, 9, 14] and the references therein. MPH has similarity to the LSFR-based scheme of [7]; to some extent, MPH is a $GF(p)$ analog of the $GF(2^n)$ scheme there. The IBC-Hash method of the RIPE project [11] uses the residue of a message with respect to a random prime; MPH differs in that it uses the Montgomery residue with respect to a random, structured composite, and this produces different characteristics for both implementation and security.

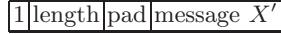
2 Definition

MPH computes the residue of a message with respect to a randomly selected modulus M . The selection of M is constrained by two positive integer constants, D and W , according to the following rules:

- R1** $2^D < M < 2^{D+W} - 1$;
- R2** $M \bmod 2^W = 2^W - 1$;
- R3** M has a prime factor $q > 2^D$.

W is typically selected to be the word size, in bits, of the CPU or hardware device of interest. D is a multiple of W , typically a small multiple. In the examples here, $W = 32$, and D is chosen as 32, 64, or 96. When we mention the ‘‘32-bit case’’ of MPH, we mean the case of $D = 32$; when we mention the ‘‘64-bit case’’ or ‘‘96-bit case’’, we mean $D = 64$ or $D = 96$, respectively.

M may be any integer in the range $(2^D, 2^{D+W} - 1)$ such that the least significant W bits of M are 1-bits (condition R2), and such that M has a large prime factor, where ‘‘large’’ means greater than 2^D (condition R3). We will see later that there are many values of M that meet these requirements.

**Fig. 1.** Format of padded message

The MPH-MAC of a message is computed in four steps: (1) form the padded message; (2) compute the MPH hash of the padded message; (3) compute the hash tag; (4) encrypt the hash tag. We now define each of these steps. Appendix A lists a numerical example that may be helpful to understanding the definition.

2.1 Form the Padded Message

Let the message string consist of L' bits, $\{x_0, x_1, \dots, x_{L'-1}\}$, with $0 \leq L' < 2^{W-1}$. We define a corresponding integer $X' = \sum_{j=0}^{L'-1} x_j 2^j$. The format of the padded message is shown in Figure 1. The padded message begins with a 1-bit, followed by a length field of $W - 1$ bits containing the binary encoded value of L' . The trailing L' bits of the padded message are the bits of the original message X' . A pad field of 0-bits is prepended to X' to make the total length of the padded message a multiple of W ; the length of the pad field ranges from 0 to $W - 1$ bits, inclusive.

The leading 1-bit and the length field delineate the precise location of the original message bits within the padded message. They also defend against certain attacks that involve prepending 0-bits or concatenating messages.

The padded message can be concisely described as the integer X , where

$$X = 2^{L-1} + 2^{L-W} L' + X' ,$$

$$L = W \lceil L'/W + 1 \rceil .$$

L is the length, in bits, of the padded message. Note that $2^{L-1} \leq X < 2^L$.

When forming integer X' from the message string, the order of message bits can be defined in any convenient way, as long as all parties have a clear specification of the ordering convention. For example, in an application where a message consists of a string of bytes $\{b_0, b_1, \dots, b_{K-1}\}$, the order of bits in the message string can be defined such that $X' = \sum_{j=0}^{K-1} b_j 2^{8j}$. These sorts of considerations are standard practice when adapting a MAC algorithm to a particular application.

2.2 Compute the MPH Hash

Given the padded message X of length L bits, the MPH hash value is computed using the well-known Montgomery modular reduction technique. Figure 2 gives a pseudocode definition of the computation. The simple implementation described in the pseudocode is only intended to define the computation; faster implementations are possible, as discussed in Section 3.

```

// Input:
//   L-bit integer  $X$ ,  $X > 0$ ,  $2^{L-1} \leq X < 2^L$ ;
//   integer  $M$ ,  $M > 0$ ,  $M$  odd.
// Return value:  $H_M(X)$ .
 $x \leftarrow X$ 
for  $j \leftarrow 1$  to  $L$  do
    if (least significant bit of  $x$  is 1)  $x \leftarrow x + M$ 
     $x \leftarrow x/2$ 
return  $x$ 

```

Fig. 2. Pseudocode definition of $H_M(X)$

One thing to note is that the pseudocode definition never returns a value of 0. Instead, if the padded message X is a multiple of the modulus M , then a value of M is returned. This has no effect on the security properties of MPH. Thus, even though it would be possible to check for a return value of M and map it to 0, there is simply no need to do so, and the pseudocode definition stands.

We can concisely state the definition of the MPH hash of the padded message X as

$$H_M(X) = \begin{cases} 2^{-L}X \bmod M, & \text{if } X \text{ is not a multiple of } M; \\ M, & \text{otherwise.} \end{cases}$$

2.3 Compute the Hash Tag

After the hash value $H_M(X)$ has been computed, an extra processing step is applied to generate the hash tag. This step serves two purposes. First, it maps the $(D + W)$ -bit hash value into a D -bit hash tag. This is desirable because MPH offers a collision rate slightly less than 2^D , and if we do the mapping right, we can achieve this collision rate using only D bits. Second, it allows us to achieve provable security by encrypting the hash tag with a stream cipher, i.e., by exclusive-oring the hash tag with a random pad value. If we skip the hash-tag computation and directly encrypt $H_M(X)$ with a stream cipher, it does not appear that we get provable security. There may be other provably-secure encryption methods that can be applied directly to $H_M(X)$, such as modulo-adding a random pad value, but we want results that apply to a conventional stream cipher.

The hash tag is computed using a construction similar to one proposed by Krovetz [9, Chapter 5]. In the construction here, we use W -bit integers in the range of $[0, 2^W]$. The hash value $H_M(X)$ is split into W -bit integers, h_0, h_1, \dots, h_d , with $d = D/W$. The hash tag consists of W -bit integers, f_0, f_1, \dots, f_{d-1} , computed as

$$f_j = \left(\left(\sum_{i=0}^d h_i a_{j+i} \right) \bmod p_W + b_j \right) \bmod 2^W .$$

The value p_W is the largest W -bit prime; for $W = 32$, $p_W = 2^{32} - 5$. The values $\{a_0, a_1, \dots, a_{2d-1}\}$ and $\{b_0, b_1, \dots, b_{d-1}\}$ are randomly picked W -bit integers in $[0, p_W)$ and $[0, 2^W)$, respectively; they are part of the MPH key and are picked at the same time the MPH modulus M is picked. The values f_j together constitute D bits that are used as the hash tag.

Note that computation of the hash tag requires the MPH-MAC key to include $3D$ random bits for the values of a_i and b_j . The key also includes the upper D bits of the modulus M . (The lower W bits of M are 1-bits and need not be included in the key.) Thus, the total size of the MPH-MAC key is $4D$ bits.

Computing the hash tag is similar to computing an MMH-MAC over $H_M(X)$. The difference is that, in MMH, the quantities b_j would be random pad values, freshly picked for each message. Here, the quantities b_j are part of the MPH key, picked once and used for all messages protected by that key.

The concatenation of the MPH-hash computation with the hash-tag computation bears some resemblance to a two-level tree hash. However, it is not a tree, since the input message is wholly digested (i.e., converted to a fixed-length string) by the first level (the MPH hash). The second level (the hash tag) thus has fixed complexity that does not vary with message size. This makes for a simpler implementation than a tree-hashing scheme, particularly in hardware.

2.4 Encrypt the Hash Tag

The MPH-MAC of the message is formed by encrypting the hash tag using a stream cipher. That is, with $\{f_0, f_1, \dots, f_{d-1}\}$ as the hash tag, the MPH-MAC $\{e_0, e_1, \dots, e_{d-1}\}$ is formed as $\{f_0 \oplus c_0, f_1 \oplus c_1, \dots, f_{d-1} \oplus c_{d-1}\}$, where each c_j is a W -bit random pad value generated by the stream cipher, and “ \oplus ” denotes exclusive-or. The pad values are freshly generated for each message. Since $d = D/W$, each message requires D bits from the stream cipher.

3 Implementation Considerations

The simple “bit at a time” logic described in the pseudocode of Figure 2 is intended to define the computation, not necessarily to guide implementation. Nevertheless, it may be practical to implement the pseudocode’s logic as is, if performance requirements are not too high. Hardware implementations, in particular, seem feasible for achieving throughput rates up to several hundred megabits/sec. The interim sum for x can be maintained in carry-save format until the end of the computation, making it unnecessary to implement a high-speed carry chain.

Faster implementations are possible, both in hardware and software. The basic idea is to right-shift x by V bits at a time, after adding to x a multiple of M carefully selected to cancel out the low V bits of x . The pseudocode in Figure 2 illustrates the principle for $V = 1$. For $V > 1$, the cancellation can be effected by either a V -bit lookup table that selects the appropriate multiple of M to add, or by multiplication of M by a V -bit scale factor.

The lookup-table method is suitable for relatively small values of V . It is attractive in hardware when a good multiplier circuit is not feasible, or in software when the CPU does not support a fast multiply operation. The looked-up multiple vM can be stored as $v(M + 1)$, which requires only $(D + V)$ bits since the lower W bits of $v(M + 1)$ are 0-bits. The required table size is thus $(D + V)2^V$ bits. For example, for $V = 4$ and $D = W = 32$, a table of 576 bits (72 bytes) is required. It is easy to trade off memory usage for computation time, typically using V in the range 2–8.

The scale-factor method is suitable for high-speed hardware and for CPUs that have a good multiply capability. If we restrict $V \leq W$, then the scale factor v to apply to M is simply the low V bits of x . That is, with $v = x \bmod 2^V$, then $(x + vM) \bmod 2^V = 0$, i.e., the lower V bits of x are cancelled out. This works because the lower W bits of M are 1-bits. The multiple vM can be computed as $v(M + 1) - v$, which requires only a $(V \times D)$ -bit multiplier since the lower W bits of $(M + 1)$ are 0-bits.

As an example of the obtainable performance, consider the two cases specified by $D = W = 32$ and $D = 2W = 64$. These are the “32-bit” and “64-bit” cases mentioned in Section 1. Using the scale-factor method, the MPH hash can be computed for $D = 32$ in less than 1.9 clocks per byte (cpb) on a Pentium Pro or Pentium III machine, with code plus data size less than 200 bytes. For $D = 64$, the corresponding numbers are 3.2 cpb and 350 bytes.

4 Security Bound

MPH-MAC has a provable security bound. Let Ω be the set of values M that satisfy conditions R1–R3. The MPH family of hash functions, $\{H_M(\cdot) : M \in \Omega\}$, is ε -Almost Universal (ε -AU). That is, there exists some $\varepsilon > 0$ such that, for any two distinct messages X and Y , $\Pr\{H_M(X) = H_M(Y)\} \leq \varepsilon$. The actual value of ε depends on the strategy for picking M ; more will be said about this later. Because of the Krovetz construction used to form the hash tag, and because the hash tag is encrypted using a stream cipher, the ε -AU property is sufficient to bound the probability that an attacker can find a collision [9, Chapter 5]. That is, with MPH-MAC, the probability that an attacker can forge a message is no greater than ε .¹

We consider the case that M is selected uniformly from Ω . This follows the usual convention for universal hashing, in which a particular hash function is selected uniformly from a family of hash functions.

As a preliminary step, we need to calculate N_Ω , the number of elements in Ω . This is done in Appendix B. The result from the appendix is

$$N_\Omega \approx 2^D \left[\ln \left(\frac{D + W}{D} \right) - \frac{1}{\ln(2)} \left(\frac{1}{D} - \frac{1}{D + W} \right) \right] .$$

¹ Actually, the hash-tag construction increases the collision probability slightly, but the increase is negligible compared to ε .

Table 1. N_Ω and ε_u versus D , for $W = 32$ and $N_x = 12000$

D	N_Ω	ε_u
32	$2^{31.42}$	$2^{-22.87}$
64	$2^{62.67}$	$2^{-55.12}$
96	$2^{94.18}$	$2^{-87.22}$

Table 1 lists numerical values for N_Ω for $W = 32$ and several values of D . The table shows that N_Ω is close to 2^D . This means that a sizeable fraction of the 2^D values of M that satisfy conditions R1 and R2 also satisfy condition R3, i.e., a significant fraction of them have a large prime factor.

We also need to establish that the prime factor q defined in condition R3 is unique across elements of Ω . To see this, let $M \in \Omega$, with $M = sq$ and q as defined in R3. Let $M' = s'q$ be a value that satisfies R2, with $M' > M$. Then by R2, $(M' - M) \bmod 2^W = 0$; in other words, $(M' - M)$ is a multiple of 2^W . Since $0 < M' - M = (s' - s)q$, and since q is odd, it follows that $(s' - s)$ is a positive multiple of 2^W . Hence $M' = M + (s' - s)q > 2^W q > 2^{D+W}$. Thus M' does not satisfy R1, so $M' \notin \Omega$. This establishes that no two elements of Ω can have the same prime factor q defined in R3.

We are now able to compute the collision bound when M is picked uniformly from Ω . Let X and Y be any two distinct, padded messages. Without loss of generality, assume $Y > X$. If $H_M(X) = H_M(Y)$, then $Y - X$ is a multiple of M . An attacker can maximize the probability that $Y - X$ is a multiple of M by picking $Y - X$ to be a multiple of as many elements of Ω as possible. Let N_x be the size, in bits, of the largest allowed padded message. Then $Y - X < 2^{N_x}$. Each element of Ω has a unique prime factor greater than 2^D . Thus, $Y - X$ can be a multiple of at most N_x/D elements of Ω . The value of M is picked uniformly from Ω , so the probability of any particular value of M is $1/N_\Omega$. Thus, the probability that $Y - X$ is a multiple of M is at most $(N_x/D)/N_\Omega$. This leads to the bound

$$\Pr\{H_M(X) = H_M(Y)\} \leq \varepsilon_u ,$$

$$\varepsilon_u = \frac{N_x}{DN_\Omega} . \quad (1)$$

Notice that the collision bound is proportional to N_x . In other words, the longer the message, the greater the probability that an attacker can forge a message. Thus, MPH-MAC is only useful in environments where messages are limited to some reasonable, maximum size. An example of such an environment is WLANs, where messages are no longer than 1,500 bytes (12,000 bits).

Table 1 lists numerical values for ε_u for $N_x = 12000$, $W = 32$, and several values of D .

5 Modulus Selection

To make MPH-MAC work, we need a practical key-derivation algorithm, i.e., an algorithm for randomly selecting a modulus value M uniformly from Ω . The topic of modulus-selection algorithms is an interesting one all its own (see [12]), but space limitations prevent us from discussing it here. Instead, we present a single method, the “SQ algorithm,” and briefly describe its properties.

5.1 Definition of SQ Algorithm

Note: All random integers are picked uniformly on the indicated interval.

1. Pick a random integer k on $[0, W]$.
2. Pick a random integer s' on $[0, 2^k]$.
3. Set $s = 2^k + s'$. If s is even, go to step 1.
4. Compute $q_l \in (0, 2^W)$ such that $sq_l \bmod 2^W = 2^W - 1$.
5. Pick a random integer q_h on $[0, 2^{D-k}]$.
6. Set $q = 2^W q_h + q_l$.
7. If $q \leq 2^D$ or $sq \geq 2^{D+W} - 1$, go to step 1.
8. Test q for primality. If q is not prime, go to step 1.

The final pair (s, q) yields an MPH modulus $M = sq$ with $M \in \Omega$.

Step 1 picks a random integer k of m bits, where $W = 2^m$. Step 2 picks a random integer s' of k bits, defined as 0 if $k = 0$.

Step 3 combines s' and k into s . If s is even, it is rejected, and the procedure starts again. Thus, at the start of step 4, s is odd-valued, and $2^k \leq s < 2^{k+1}$.

Step 4 computes q_l , the negative of the multiplicative inverse of s , modulo 2^W . This is a fast computation commonly performed as part of Montgomery reduction. Appendix C describes the computation.

Step 5 picks a random integer q_h of length $D - k$ bits.

Step 6 combines q_h and q_l into q . There are 2^{D-k} possible values for q , corresponding to the possible values for q_h . The value of q satisfies $sq \bmod 2^W = 2^W - 1$.

Steps 7 and 8 check that $M = sq$ is a valid MPH modulus with q as the large prime factor. Step 7 checks that q and sq are within the required range. Step 8 checks that q is prime. If either of the checks fail, s and q are rejected, and the procedure starts again. If both checks pass, then the algorithm is done, with $M = sq$ a valid MPH modulus.

When testing q for primality in Step 8, any appropriate primality test may be used. A good choice is the Miller-Rabin test [10, Section 4.2.3], preceded by a small-factor test that quickly rejects q if it is a multiple of a small prime.

Note that the SQ algorithm is well suited to fixed point arithmetic. This makes it feasible even on simple CPUs that do not provide good support for floating point arithmetic.

5.2 Uniformity

In this section we show that the SQ algorithm selects M uniformly over Ω . This means that the SQ algorithm is optimum for the uniform distribution, i.e., it achieves the collision bound ε_u described in Section 4.

First consider the value of s after step 3 has executed and passed. For a given value of k , s takes on an odd value in the range $2^k \leq s < 2^{k+1}$. For $k = 0$, s takes on only a single value, 1. For $k > 0$, s takes on one of 2^{k-1} possible values. Step 3 always passes for $k = 0$; it passes half the time for $k > 0$. Thus, for any given k , the probability of a particular value of s given that step 3 has passed is $\alpha 2^{-k}$, where α is the probability that $k = 0$ given that step 3 has passed. It is easy to compute $\alpha = 2/(W + 1)$.

Next consider the value of q selected in step 6. For a given value of k , the value of q is picked uniformly from one of 2^{D-k} possible values. Thus, the probability of any particular value of q is 2^{k-D} . Since a value of s has probability $\alpha 2^{-k}$, it follows that the probability of a pair (s, q) is $(\alpha 2^{-k})(2^{k-D}) = \alpha 2^{-D}$.

Note that this value, $\alpha 2^{-D}$, does not depend on s or q . Thus, any pair (s, q) produced by step 6 occurs with probability $\alpha 2^{-D}$; i.e., the pairs are uniformly distributed. Steps 7 and 8 reject some of the pairs selected by step 6, but this does not affect the uniformity of the distribution. Thus, the output pairs (s, q) produced by the SQ algorithm are uniformly distributed.

It is easy to show that the set of output pairs produced by the SQ algorithm has a one-to-one correspondence with Ω . That is: if $M \in \Omega$, then there is exactly one output pair (s, q) for which $M = sq$; and if (s, q) is an output pair, then there is exactly one $M \in \Omega$ for which $M = sq$. Since the output pairs are uniformly distributed, we conclude that the SQ algorithm selects a value M uniformly over Ω .

5.3 Run Time

The run time of the SQ algorithm is well suited to analysis. The analysis is an interesting topic (see [12]), but space limitations prevent us from describing it here. Instead, we present empirical results drawn from an implementation.

The SQ algorithm with $W = 32$ was implemented in the C programming language on an 800 MHz Pentium III. No assembly language was used, but care was taken with time-critical inner loops to ensure that the compiler produced good code. The implementation was used to generate 1,000 modulus values for each of several choices of D . The total number of machine cycles was measured and divided by 1,000 to get an average value. The results are listed in Table 2 in the column labelled “Cycles”. To get a time measurement, divide the “Cycles” entry by the machine cycle time; e.g., on our 800 MHz machine, the run time for $D = 32$ is $0.23/0.800 = 0.29$ msec, and the run time for $D = 64$ is $1.05/0.800 = 1.31$ msec. The test was also run on a Pentium Pro machine; the number of machine cycles was identical to that for the Pentium III.

The column in Table 2 labelled “ t_{mr} ” indicates the number of Miller-Rabin iterations performed during the primality test of Step 8. The value of t_{mr} was

Table 2. SQ run-time measurements and associated parameters versus D , for $W = 32$

D	Cycles	t_{mr}	%-MR	b	Bits
32	0.23×10^6	16	78.2%	9	1,908
64	1.05×10^6	33	88.9%	10	6,105
96	3.09×10^6	49	92.8%	11	12,081

picked sufficiently large to ensure that the false-prime probability of the Miller-Rabin test produces negligible degradation of the security bound ε_u shown in Table 1. The value of t_{mr} is calculated in Appendix D. The column labelled “%-MR” shows the percentage of machine cycles consumed by Miller-Rabin iterations. Not surprisingly, the percentage is quite high, meaning that the Miller-Rabin iterations dominate the run time.

We suspect that the values of t_{mr} shown in Table 2 are larger than necessary, since they are based on a relatively simple analysis of Miller-Rabin. In [4], much smaller values of t_{mr} are found for the special case of testing a k -bit integer picked uniformly on $(2^{k-1}, 2^k)$. Unfortunately, the analysis of [4] does not apply here because we are not picking uniformly on an interval. If we pretend the analysis applies, and we compute t_{mr} using the methods of [4], we get values that are 30–40% lower than those in Table 2; these would significantly improve run time. We conjecture that the reduced values of t_{mr} are sufficient for achieving the required false-prime probability, but it remains an open question if this is in fact the case. So we stick to the values of t_{mr} in the table.

The column in Table 2 labelled “ b ” indicates the size of the small-factor test that preceded Miller-Rabin. Factors up to 2^b were considered; that is, for each candidate q , the residue $q \bmod f$ was computed for each prime $f < 2^b$, with q rejected if any residue was zero. The value of b was optimized experimentally to get the best average run time. Note that this value is optimum only for our implementation; another implementation will have its own optimum, depending on the run time of the residue computation versus that of Miller-Rabin.

The column in Table 2 labelled “Bits” shows the average number of random bits consumed. These include the random bits consumed by steps 1–7 of the SQ algorithm as well as the random bits consumed by the Miller-Rabin test of step 8. In our implementation, we used an RC4 encryptor [13, p. 397] to generate pseudorandom bits. The encryptor was initialized using a 128-bit random seed for the encryption key. The resulting keystream produced by the encryptor was used as a source of pseudorandom bits. RC4 is attractive for this purpose because of its efficiency; in our implementation, its contribution to total run time was negligible.

The run-time numbers for the SQ algorithm may come as a surprise to readers familiar with RSA key generation, which takes much longer than the numbers mentioned here. The reason for the difference in run time is the size of the primes — RSA uses large primes, MPH uses short primes, and that makes a huge

difference. We have noticed a tendency among crypto engineers (including the authors) to initially dismiss MPH modulus selection as too time-consuming, based on experience with RSA key generation. This is simply not the case.

5.4 Use with Key Agreement Schemes

Some key-agreement schemes, such as those based on Diffie-Hellman, do not allow the parties to exchange a specific MAC key. Instead, these schemes provide the parties with a shared, random “key blob.” The blob serves as input to a key-derivation algorithm that each party runs to get the MAC key.

The SQ algorithm is suitable for use with such a key-agreement scheme. The blob produced by the scheme is used to seed a pseudorandom generator (PRG), and the PRG is used to generate the random bits needed by the SQ algorithm. The parties need a clear specification of the PRG (e.g., the RC4 PRG described above) and of the details of the primality test in step 8 (e.g., the number of Miller-Rabin iterations). Once these are specified, the SQ algorithm is a deterministic function of the PRG seed and is thus suitable for use with a key-agreement scheme.

6 Nonuniform Distributions

The collision-rate bound ε_u calculated in Section 4 is based on selecting the modulus M uniformly from Ω . This follows the usual practice for universal hashing, in which a particular hash function is selected uniformly from a family of hash functions. In the case of MPH, however, a uniform distribution is not the best choice, because some modulus values offer better collision resistance than others. Specifically, the collision resistance of a modulus M increases with the size of its large prime factor q .² We get a better collision-rate bound by using a distribution that favors values of M that have large values of q .

In [12], we show that a good choice is to make $\Pr\{M\}$ proportional to $\log(q)$. We call this the “log-q” distribution. The collision-rate bound ε_l produced by the log-q distribution is slightly better than ε_u . Table 3 shows the improvement by listing $\varepsilon_l/\varepsilon_u$ for $W = 32$ and several values of D . The improvement — 0.2 to 0.5 bits — is not of much practical significance, but it is interesting nevertheless. The improvement is better for small D than for large D .

It is easy to incorporate a log-q distribution into the SQ algorithm. Just add the following step before or after step 8:

- Pick a random, real number t uniformly on $[0, D + W]$. If $t > \log_2(q)$, go to step 1.

Since t is uniform on $[0, D + W]$, then $\Pr\{t \leq \log_2(q)\} = \log_2(q)/(D + W)$. Thus, the probability that the step passes is proportional to $\log_2(q)$. This imparts the

² To be pedantic, the collision resistance *as approximated by the collision-rate bound* increases with the size of the large prime factor.

Table 3. $\varepsilon_l/\varepsilon_u$ and $\varepsilon_o/\varepsilon_l$ versus D for $W = 32$

D	$\varepsilon_l/\varepsilon_u$	$\varepsilon_o/\varepsilon_l$
32	$2^{-0.53}$	$2^{-1.01}$
64	$2^{-0.30}$	$2^{-1.60}$
96	$2^{-0.21}$	$2^{-2.00}$

log-q distribution onto the selection of M and thus achieves the collision bound ε_l . The failure rate of the step is fairly low, roughly 25% for $D = W$ and 12% for $D = 3W$, so adding the step does not increase the run time by much.

Since log-q gives a better result than uniform, it is natural to wonder what the optimum distribution is for picking M . Unfortunately, this seems to be a hard question to answer. In [12], however, we derive a lower-bound ε_o on the collision rate of the optimum distribution. Table 3 lists $\varepsilon_o/\varepsilon_l$ for $W = 32$ and several values of D . The table shows that ε_o is only about 1–2 bits better than ε_l . In other words, the log-q distribution provides security that is no more than 1–2 bits worse than optimum. It remains an open question as to exactly what the optimum distribution is, but the fact that the improvement is minor makes the question of little practical importance.

Acknowledgment

The authors thank David Wagner of U.C. Berkeley and David McGrew of Cisco Systems for their help and insight. The authors also thank the anonymous reviewers for valuable feedback on the original manuscript.

References

- [1] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: fast and secure message authentication. In *Proc. of CRYPTO 99*, LNCS 1666, page 216ff. Springer-Verlag, 1999. [52](#), [53](#)
- [2] Cable Television Laboratories, Inc. *PacketCable Security Specification*, October 2002. <http://www.packetcable.com/downloads/specs/PKT-SP-SEC-I06-021018.pdf>. [51](#)
- [3] L. Carter and M. Wegman. Universal hash functions. *J. of Computer and System Sciences*, 18:143–154, 1979. [51](#)
- [4] Ivan Damgard, Peter Landrock, and Carl Pomerance. Average case error estimates for the strong probable prime test. *Mathematics of Computation*, 61:177–194, July 1993. [61](#)
- [5] Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the Gbit/second rates. In *Proc. of the 4th Workshop on Fast Software Encryption*, LNCS 1267, pages 172–189. Springer-Verlag, 1997. [51](#), [53](#)
- [6] IEEE. *IEEE Std 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 1999 edition. [50](#)

- [7] Hugo Krawczyk. LFSR-based hashing and authentication. In *Proc. of CRYPTO 94*, LNCS 839, pages 129–139. Springer-Verlag, 1994. [53](#)
- [8] Ted Krovetz and Phillip Rogaway. Fast universal hashing with small keys and no preprocessing: the PolyR. In *Proc. of ICISC 2000*, LNCS 2015, page 73 ff. Springer-Verlag, 2000. [53](#)
- [9] Theodore D. Krovetz. *Software-Optimized Universal Hashing and Message Authentication*. PhD thesis, University of California, Davis, 2000.
<http://www.cs.ucdavis.edu/~krovetz/papers>. [53](#), [55](#), [57](#)
- [10] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997. [59](#), [65](#), [66](#), [67](#)
- [11] RIPE Consortium. *RIPE Integrity Primitives: Final Report of RACE Integrity Primitives Evaluation (R1040)*, LNCS 1007. Springer-Verlag, 1995. [53](#)
- [12] Michael J. Sabin and Douglas L. Whiting. Selecting an MPH modulus.
http://www.hifn.com/support/crypto MPH_modulus.pdf. [59](#), [60](#), [62](#), [63](#)
- [13] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, second edition, 1996. [61](#)
- [14] UMAC: Fast and provably secure message authentication.
<http://www.cs.ucdavis.edu/~rogaway/umac/>. [52](#), [53](#)

A MPH-MAC Example

The following listing illustrates an example MPH-MAC calculation. All numbers in the listing are in hexadecimal format.

Key material:

```
M: 2F7E4143 8297A730 FFFFFFFF
a3, a2, a1, a0: 96E6967C FFE862F5 BB1CC8C8 4C395E7D
b1, b0: 5212B33B 0C8A1B24
```

Message to authenticate ("Now is the time for all"):

```
X': 6C6C61 20726F66 20656D69 74206568 74207369 20776F4E
```

Padded message:

```
X: 800000B8 006C6C61 20726F66 20656D69 74206568 74207369 20776F4E
```

Computation of MPH hash:

```
j=20, x: 800000B8 006C6C61 20726F66 266B5DE8 719936FE D79AA357
j=40, x: 800000B8 006C6C61 48722876 A32B74A5 1D113BA5
j=60, x: 800000B8 05D0EC38 F923A215 69B3823A
j=80, x: 939C16ED 38548172 92A9652F
j=A0, x: 1B356E57 B195684F 587B8871
j=C0, x: 106A5168 B4EFFADB 13E33CF0
j=E0, x: 03B0871A OFA16130 91BF34CB
```

Hash tag computation:

```
h2, h1, h0: 03B0871A OFA16130 91BF34CB
f1, f0: 83B65D7A 02825E41
```

Hash tag encryption:

c1, c0:	36EA0F3C 607285AE
e1, e0:	B55C5246 62F0DBEF

In this example, $W = 32$ and $D = 64$. The key material consists of the modulus M and the hash tag quantities $\{a_0, \dots, a_3\}$ and $\{b_0, b_1\}$.

The message to be authenticated is the string “Now is the time for all”. The bytes of the string are assembled into integer X' least-significant first; i.e., “N” (0x4E) is the least-significant byte of X' and “l” (0x6C) is the most significant.

The padded message X is formed by concatenating: a 1-bit; a 31-bit encoding of the length in bits of X' (0xB8); and X' , padded with leading 0-bits so that its length in bits is a multiple of W .

The listing shows some intermediate values in the computation of the MPH hash $H_M(X)$. Referring to the definition in Figure 2, the listing shows the value of j and X at the bottom of the “for” loop in the figure, sampled every 32 (0x20) iterations. The final value of x , which occurs when j equals the length in bits of the padded message (0xE0), is $H_M(X)$.

To compute the hash tag, $H_M(X)$ is partitioned into three 32-bit words $\{h_0, h_1, h_2\}$. Then the hash tag is computed as two 32-bit words $\{f_0, f_1\}$ using the procedure of Section 2.3.

To encrypt the hash tag, two 32-bit random pad values $\{c_0, c_1\}$ are generated by the stream cipher. These are exclusive-or'd with the hash tag to produce the MPH-MAC $\{e_0, e_1\}$.

B Calculating N_Ω

Let I be the interval $(2^D, 2^{D+W} - 1)$. The prime number theorem [10, Fact 2.95] states that the number of prime integers in I is approximately

$$\frac{2^{D+W} - 1}{\ln(2^{D+W} - 1)} - \frac{2^D}{\ln(2^D)} .$$

We can write this as

$$\int_I \frac{\ln(x) - 1}{\ln^2(x)} dx .$$

Let q be a prime integer in I . The number of multiples of q in I is $\lfloor (2^{D+W} - 2)/q \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function; we can approximate this value as $2^{D+W}/q$. For simplicity we can also approximate I as $(2^D, 2^{D+W})$. So, the number N_1 of integers in I that have a prime factor in I is given by

$$\begin{aligned} N_1 &\approx \int_{2^D}^{2^{D+W}} \frac{\ln(x) - 1}{\ln^2(x)} \frac{2^{D+W}}{x} dx \\ &= 2^{D+W} \left[\ln\left(\frac{D+W}{D}\right) - \frac{1}{\ln(2)} \left(\frac{1}{D} - \frac{1}{D+W} \right) \right] . \end{aligned}$$

Of these N_1 integers, it is reasonable to assume that 1 in 2^W of them have lower W bits that are all 1-bits. Thus,

$$N_\Omega \approx 2^D \left[\ln \left(\frac{D+W}{D} \right) - \frac{1}{\ln(2)} \left(\frac{1}{D} - \frac{1}{D+W} \right) \right]. \quad (2)$$

C Calculating q_l

Here is an algorithm for calculating q_l :

```

Algorithm get_ql(s)
// Input: s, odd-valued, 0 < s < 2W
// Output: ql, 0 < ql < 2W, sql mod 2W = 2W-1
ql ← 1
t ← s
for i ← 1 to W − 1 do
    if (bit i of t is 0)
        t ← (t + s2i) mod 2W
        ql ← ql + 2i
return ql

```

Bit i corresponds to 2^i . For example, bit 0 is the least significant bit.

D MR Iterations

The collision bound ε_u computed in Section 4 assumes that the modulus $M = sq$ is valid, i.e., it assumes that $M \in \Omega$. If q is composite and the Miller-Rabin (MR) test mistakenly declares it to be prime, then M may not be valid, i.e., it may be that $M \notin \Omega$. Let ε' be the collision bound that takes this false-prime possibility into account. Then

$$\begin{aligned} \varepsilon' = & \Pr\{H_M(X) = H_M(Y)|M \in \Omega\} \Pr\{M \in \Omega\} + \\ & \Pr\{H_M(X) = H_M(Y)|M \notin \Omega\} \Pr\{M \notin \Omega\}. \end{aligned}$$

Since $\varepsilon_u = \Pr\{H_M(X) = H_M(Y)|M \in \Omega\}$, we can bound $\varepsilon' \leq \varepsilon_u + \Pr\{M \notin \Omega\} \leq \varepsilon_u + \rho$, where ρ is the MR false-prime probability, i.e., the probability that MR declares q to be prime when it is actually composite. Let's say we do not want to degrade the collision bound by more than 0.01 bits, i.e., we want $\varepsilon' \leq 2^{0.01}\varepsilon_u$. Thus we want $\rho/\varepsilon_u \leq 2^{0.01} - 1$. This degradation is small enough that the numerical values in Table 1 do not change, since the precision of the table is 0.01 bits.

Let t be the number of MR iterations performed. Then $\rho \leq 2^{-2t}/\mu$, where μ is the probability that the value of q submitted to the MR test is prime [10, Note 4.47]. Let C be the set of values for q that might be submitted to the MR test. That is, $q \in C$ means that q is a value that may occur at step 8 of the SQ algorithm, and q would survive the small-factor test that precedes MR. Let N_{mr}

be the number of elements in C . The number of primes in C is N_Ω . Since q is uniformly picked from C , it follows that $\mu = N_\Omega/N_{mr}$. Combined with (1), this yields

$$\frac{\rho}{\varepsilon_u} \leq \frac{2^{-2t} DN_{mr}}{N_x} . \quad (3)$$

Let λ be the proportion of odd integers that are not divisible by any factor checked in the small-factor test. Then $\lambda = \prod_f (1 - 1/f)$, where f ranges over the factors used in the small-factor test, i.e., ranges over the prime values in $(2, 2^b]$. The value of λ may be computed directly; it is approximately $1.12/\ln(2^b)$ [10, p. 146]. Using a logic similar to that for calculating N_Ω in Appendix B, we can compute N_{mr} as

$$N_{mr} \approx \frac{1}{2^W} \int_{2^D}^{2^{D+W}} \frac{\lambda}{2} \frac{2^{D+W}}{x} dx = \lambda 2^{D-1} \ln(2^W) .$$

Substituting this into (3) gives

$$\frac{\rho}{\varepsilon_u} \leq \frac{\lambda D 2^{D-1-2t} \ln(2^W)}{N_x} . \quad (4)$$

The values for t_{mr} in Table 2 are the smallest integer values for t such that the right-hand side of (4) is less than $2^{0.01} - 1$.

An Analysis of Proxy Signatures: Is a Secure Channel Necessary?

Jung-Yeon Lee¹, Jung Hee Cheon¹, and Seungjoo Kim²

¹ IRIS (International Research center for Information Security)

ICU (Information and Communications Univ.), Korea

{bushman, jhcheon}@icu.ac.kr

² Korea Information Security Agency, Korea

skim@kisa.or.kr

Abstract. A proxy signature enables the original signer to delegate her signing capability to a proxy entity, who signs a message on behalf of the original signer. In this paper, we discuss the necessity of a secure channel in proxy signatures. Though establishing a secure channel has much influence on the efficiency of the scheme, to the best of our knowledge, this topic has not been discussed before. All known proxy signatures used a secure channel to deliver a signed warrant except one which used a 3-pass weak blind signature. However, the KPW scheme [2] appeared to be secure without the secure channel. We think that our result can contribute to designing more efficient proxy signature scheme.

1 Introduction

An employee in a company needs to go on a business trip to someplace which has no computer network access. During the trip he will receive e-mails, and may be expected to respond to some messages urgently. A solution for this situation is a *proxy signatures*. Before the trip, he delegates his signing capability to his secretary (called a proxy signer), and instructs his secretary to respond to the e-mails in place of him according to a prearranged plan. Then the secretary responds to the e-mails using the proxy signature.

In order to create this kind of signature securely, it should satisfy the following requirements [5, 3].

R1. Verifiability: From the proxy signature a verifier can be convinced of the original signer's agreement on the signed message.

R2. Strong unforgeability: A designated proxy signer can create a valid proxy signature for the original signer. But the original signer and other third parties who are not designated as a proxy signer cannot create a valid proxy signature.

R3. Strong identifiability: Anyone can determine the identity of the corresponding proxy signer from the proxy signature.

R4. Strong undeniability: Once a proxy signer creates a valid proxy signature of an original signer, he cannot repudiate the signature creation.

R5. Prevention of misuse: The proxy signer cannot use the proxy key for other purposes than generating a valid proxy signature. That is, he cannot sign, with the proxy key, messages that have not been authorized by the original signer.

The basic idea to implement a proxy signature scheme is that the original signer creates a signature (called a proxy) on the delegation information (the identity of the designated proxy signer, valid period, instruction for signing, or any warrant information) and then the proxy signer uses it to generate a proxy private key and signs on the delegated message. Since the proxy key pair is generated using the original signer's signature on delegation information, any verifier can check the original signer's agreement from the proxy signature.

We should note here that, in [5], Mambo *et al.* emphasized that the secure channel between the original signer and the proxy signer is necessary in the proxy delivery step. Otherwise, anyone who obtained the proxy can create the valid proxy signature. In this paper, we will take a closer look at the necessity of secure channel in the proxy signatures. Because establishing a secure channel has much influence on the efficiency of the scheme, it is desirable to construct a proxy signature scheme without secure channel. Furthermore, a proxy signature scheme without secure channel does not employ encryption. Thus we don't need to consider the current debate about cryptographic policy as to whether the law enforcement should be given when authorized surreptitious access to the plaintext of encrypted messages.

Related Works. A proxy signature (MUO scheme) was first introduced by Mambo *et al.* [5]. Since its warrant does not include the information of the proxy signer, a secure channel is required in the proxy delivery step. Petersen and Horster [7] proposed a proxy-protected scheme (PH scheme) using a 3-pass weak blind signature, where the proxy private key is an ordinary signature on the identity of the proxy signer using Schnorr's signature scheme. The KPW scheme [2] specified the warrant so as to contain the identity information and the limit of the signing capability of the proxy signer in order to prevent the misuse of the signing capability by the proxy signer. Lee *et al.* [3] proposed a scheme (LKK scheme) based on the KPW scheme, where the original signer and the proxy signer do not play the same role in the generation of a proxy signature and so the verifier can identify both of them without seeing the warrant information.

Our Contribution. We analyze the necessity of the secure channel to deliver a proxy certificate for all known proxy signature schemes [5, 2, 6, 3] to make use of secure channel. We show that all schemes but the KPW scheme are insecure without the secure channel. Further we provide a heuristic proof for the security of the KPW scheme without the secure channel, and propose the modified schemes for the MUO scheme and the LKK scheme in order to be secure without the secure channel. Finally, we show that the LKK scheme does not satisfy the

strong unforgeability even with the secure channel which is different from the author's claim.

Organization. In Section 2, 3 and 5, we will review the proxy signature scheme the MUO scheme, the PH scheme and the LKK scheme respectively and analyze the functions of each scheme and the necessity of a secure channel. Then, in Section 4 we show that the KPW scheme is secure although we remove the secure channel in the proxy delivery step. In Section 6, We compare the functions and efficiency of each scheme and revise the MUO scheme and the LKK scheme so that their schemes are secure although we remove the secure channel. Finally, we conclude this paper in Section 7.

2 The MUO Scheme and Its Analysis

We review the proxy-protected proxy signature proposed in [5], which is based on the Discrete Logarithm Problem (DLP). Throughout this paper, p is a large prime with $2^{511} < p < 2^{512}$, q is a large prime with $q \mid p - 1$ and g is a generator of a multiplicative subgroup of \mathbb{Z}_p^* with order q . $h()$ denotes a collision resistant hash function. In addition, it is assumed that Alice is an original signer with a key pair $(x_A, y_A (= g^{x_A} \bmod p))$ and Bob is a proxy signer with $(x_B, y_B (= g^{x_B} \bmod p))$. We denote by m_w the warrant which contains the information of the proxy signer and by m_P the delegated message.

[Basic Protocol]

1. Generation of the proxy key: Bob gets the proxy key pair (x_P, y_P) through the following steps.

1. The original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \bmod p$. Then, she calculates

$$s_A = x_A + k \cdot K \bmod q.$$

and sends (s_A, K) to Bob in a secure manner.

2. Bob checks

$$g^{s_A} \stackrel{?}{=} y_A \cdot K^K \bmod p.$$

If it is passed, Bob computes the proxy private key as

$$x_P = s_A + x_B \cdot y_B.$$

2. Proxy signature generation: When Bob signs a document m_p for the sake of Alice, he executes the DLP-based ordinary signing operation with the proxy private key x_P . The created proxy signature σ is

$$(m_p, \text{Sign}_\sigma(m_p), K, y_A, y_B).$$

3. Verification: To verify the proxy signature σ , first the verifier computes the proxy public key as

$$y_P = g^{x_P} = y_A \cdot K^K \cdot y_B^{y_B}.$$

The verification is carried out by the same checking operation as in the original signature scheme.

This scheme is a proxy-protected one and satisfies all requirements except R5 of the security requirements. The requirement is not satisfied since the warrant (s_A, K) does not include the identity information and the limit of the capability of the designated proxy signer. The proxy signer also can transfer the warrant to someone else. To avoid these problems, one can manage the revocation list which represents the possession of the warrant and the limit of the signing capability of proxy signer.

As they mentioned, this scheme needs a secure channel. If we remove the secure channel, anyone who intercepts the warrant (s_A, K) can be the proxy signer of Alice. Furthermore, if he has a warrant (s_C, K') made by another user Charlie, he can change the original signer from Alice to Charlie as follows: Let s_P be a proxy signature generated by Bob on behalf of Alice using Schnorr's scheme, i.e.,

$$\begin{aligned} s_P &= k_P + x_P \cdot h(m_p, r_P) \\ &= k_P + (x_B \cdot y_B + s_A) \cdot h(m_p, r_P) \\ &= k_P + x_B \cdot y_B \cdot h(m_p, r_P) + \underbrace{s_A \cdot h(m_p, r_P)}_{(1)}. \end{aligned}$$

Here, the underlined term (1) can be extracted from the proxy signature σ . Through the algebra

$$s_{P'} = s_P - (1) + s_C \cdot h(m_p, r_P),$$

the attacker can create the proxy signature $s_{P'}$ in which Charlie is the original signer and Bob is proxy signer. Hence, if this scheme is used without the secure channel, the proxy signer can repudiate that he/she generated the proxy signature on behalf of a specific person.

3 The PH Scheme and Its Analysis

We review the proxy-protected proxy signature proposed in [7].

[Basic Protocol]

1. **Generation of the proxy key:** Bob gets a proxy private key x_P through the 3-pass weak blind signature protocol, where $x_P = s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Namely, Bob uses Schnorr's signature on Bob's ID of Alice as the proxy private key.
2. **Proxy signature generation:** When Bob signs a document m_p for the sake of Alice, he executes the DLP-based ordinary signing operation with the proxy private key x_p . The created proxy signature σ is

$$(m_p, \text{Sign}_\sigma(m_p), ID_B, r_A)$$

3. Verification: To verify the proxy signature σ , first the verifier computes the proxy public key as

$$y_P = g^{x_P} = y_A^{h(ID_B, r_A)} \cdot r_A \mod p.$$

The verification of the proxy signature is carried out by the checking operation of the same signature scheme.

Alice does not know the proxy private key since they used a blind signature scheme at the time of the proxy signature generation. However, consequently the proxy private key is an ordinary Schnorr's signature on the identity ID_B of Bob. So it does not contain the private information of Bob. It makes several weaknesses of the PH scheme as follows:

- Since an attacker can generate Schnorr's signature on Bob's ID and then create the proxy signature in which Bob is a proxy signer regardless of Bob's will. Thus, this scheme does not satisfies the requirements R2 and R4 and is not proxy-protected.
- Like the MUO scheme, the warrant of this scheme does not contain the limit of the signing capability and so does not protect the misuse by the proxy signer [3].

We remark that the PH scheme does not use a secure channel, however it requires a 3-pass weak blind signature protocol, which increases the communication load.

4 The KPW Scheme and Its Analysis

Kim *et al.* [2] introduced the notion of the partial delegation performed by inserting the warrant into the proxy signature, i.e., the proxy signer generates proxy signatures using his private key and the warrant signed by the original signer.

[Basic Protocol]

1. Generation of the proxy key: To delegate the signing capability to proxy signer, the original signer Alice uses Schnorr's scheme to make the signed warrant m_w . If the following process is finished successfully, Bob gets a proxy key pair (x_P, y_P) .

1. Alice chooses $k_A \in \mathbb{Z}_q^*$ at random and computes $r_A = g^{k_A} \mod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \mod q$, and then sends (m_w, r_A, s_A) to a proxy signer Bob secretly.
2. Bob verifies the validity of the signature on m_w . If the signature is valid, Bob computes the proxy key pair (x_P, y_P) as $x_P = h(m_w, r_A) \cdot x_B + s_A$.

2. Proxy signature generation: Bob uses any signature scheme based on the difficulty of DLP with the key pair (x_P, y_P) and obtains a signature (r_P, s_P) for the delegated message m_P . The valid proxy signature will be the tuple

$$(m_P, r_P, s_P, m_w, r_A).$$

- 3. Verification:** A recipient can verify the validity of the proxy signature by checking that both the proxy signer and the message conform to m_w and the verification with the proxy public key

$$y_P = (y_A \cdot y_B)^{h(m_w, r_A)} \cdot r_A \mod p.$$

This proxy signature scheme uses a warrant containing the identity information and the limit of the delegated signing capability and so satisfies the security requirements R1, R3 and R4. It is a proxy-protected one in the strict sense and satisfies the requirement R2 (strong unforgeability). They said that their scheme needs the secure channel in the proxy delivery step, but their scheme is still secure without the secure channel.

In order to show that the KPW scheme satisfies the second requirement without the secure channel, we classify the security problem as follows:

1. [Attack for the role of the original signer]
 - (a) An attacker creates the signature of some user on the warrant information and so impersonate the original signer.
 - (b) An attacker replaces the signed warrant by other valid warrant signed by different user. Therefore, he change the original signer.
2. [Attack for the role of proxy signer] An attacker converts a normal signature into a proxy signature.

The first problem 1-(a) is overcome easily by using the signature scheme which is secure against the existential forgery. Since the KPW scheme uses Schnorr's signature scheme whose security is proved, we ignore this problem. Next we show that the KPW scheme is secure for the two cases 1-(b) and 2.

We noted that the previous schemes have several weaknesses when the warrants are revealed. the one of those weaknesses results from the divisibility of the roles of original signer and proxy signer in the proxy signature. Namely, the attacker can remove the part of original signer from that of the proxy signature and insert the warrants created by another original signer. Therefore, he changes the original signer. In order to protect this problem, the proxy signature scheme must include a part which binds the private key of proxy signer with the public parameter used in the signing process on the warrant by original signer. Fortunately, the proxy signature by the KPW scheme has those parts and so their scheme does not need a secure channel.

More precisely, we assume that an attacker Charlie wants to change the original signer from Alice to Charlie himself and Charlie has a signed warrant (s_A, r_A) and a proxy signature (m, r_P, s_P, m_w, r_A) , where

$$s_P = \underbrace{k_P + x_B \cdot h(m_p, r_P) \cdot h(m_w, r_A)}_{(1)} + \underbrace{s_A \cdot h(m_p, r_P)}_{(2)}.$$

Charlie must modify the term (2).

First, let's consider the situation that the modification of the term (2) has no effect on the term (1). In this case, he must sign on the warrant information with the same random number r_A as s_A . In order to do that, he must compute the integer k_A such that $r_A = g^{k_A} \bmod n$, i.e., the DLP itself. Hence, he can not use this method.

Next, we consider the situation that Charlie chooses a random number k_C independently from r_A and signs on the warrant information m'_w . Then, Charlie must modify the first term (1) and consequently obtains the $k_P + x_B \cdot h(m_p, r_P) \cdot h(m'_w, r_C)$. In order to get this term, he has two approaches. One is to use the term (1) and another is to compute directly by himself. We know that the difficulty of the latter case is equal to solving DLP through the same way as [8].

Let $A = k_P + x_B \cdot h(m_p, r_P) \cdot h(m_w, r_A)$ and $B = k_P + x_B \cdot h(m_p, r_P) \cdot h(m_w, r_C)$. Suppose that he obtained B from A . Let $H_1 = h(m_w, r_A) \cdot h(m_P, r_P)$ and $H_2 = h(m'_w, r_C) \cdot h(m_P, r_P)$. To obtain B from the algebra

$$\frac{H_2}{H_1} \cdot A = \frac{H_2}{H_1} \cdot k_P + H_2 \cdot x_B,$$

the following equation must be satisfied

$$h(m_P, g^{\frac{H_2}{H_1} \cdot k_P}) \cdot h(m'_w, r_C) = H_2.$$

However, since h is a collision resistant hash function, it is computationally infeasible to find H_2 satisfying the above equation.

5 The LKK Scheme and Its Analysis

It is based on the KPW scheme implementing the delegation with the warrant, with the difference that the warrant information signed by the original signer need not explicitly include either his identity or the identity of the proxy signer since the original signer and proxy signer do not play the same role in the generation of a proxy signature. Thus, the verifier can identify the roles of them just from the signature.

[Basic Protocol]

1. Generation of the proxy key: Bob gets a proxy key pair (x_P, y_P) through the following steps.

1. Alice chooses $k_A \in \mathbb{Z}_q^*$ at random, computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$, and then sends (m_w, r_A, s_A) to a proxy signer Bob secretly.
2. Bob verifies the validity of the signature on m_w . If the signature is valid, Bob computes the proxy key pair (x_P, y_P) as $x_P = x_B + s_A$ and $y_P = g^{x_P} \bmod p$.

2. Proxy signature generation: The proxy signer Bob signs on the delegated message m_P with the proxy private key x_P using Schnorr's signature scheme.

- 3. Verification:** A recipient checks if the proxy signer and the message conform to m_w and then verifies the validity of the proxy signature with the proxy public key

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B \mod p.$$

If both verifications hold, the proxy signature is valid. Any verifier can check the original signer's agreement on m_w , identify the proxy signer from the proxy public key, and check the validity of the proxy signer's signature on m_P .

Because this proxy signature scheme uses the warrant containing the identity information of the proxy signer and the limit of the signing capability, it overcomes the weaknesses of the MUO scheme. However, this scheme is not a proxy-protected one and either does not provide R2 (strong unforgeability) in the strict sense.

When Lee *et al.* [3] analyze the security of their scheme, they only consider the situation that first the original signer delegates the signing capability and then the proxy signer signs the delegated message. However, we found the fact that the proxy signature can also be generated by reversing the signing order. That is, first the proxy signer Bob signs on the message m_P and then the original signer Alice adds some factor $s_A \cdot h(m_P, r_P)$ to the signature and therefore created the proxy signature. Let's see the algebra:

$$\begin{aligned} s_P &= k_P + x_P \cdot h(m_P, r_P) \\ &= k_P + (x_B + s_A) \cdot h(m_P, r_P) \\ &= k_P + x_B \cdot h(m_P, r_P) + s_A \cdot h(m_P, r_P) \\ &= s_B + s_A \cdot h(m_P, r_P), \end{aligned}$$

where s_B is a signature on the message m_P by Bob and the second term is generated by the original signer by herself. Using this method, every signature generated by the Schnorr's signature scheme can be converted into a proxy signature in which the signer is regarded as the proxy signer by the verifier. In addition, the original signer can remove the same term from the valid proxy signature and obtain a plain Schnorr's signature. Consequently, this scheme is not a proxy-protected one. See Fig. 1 for the more detail scenario.

Now we discuss the necessity of a secure channel. From the above approach, we can see why the LKK scheme needs the secure channel to deliver the original signer's signature on the warrant to the proxy signer. If the signed warrant is delivered through an insecure channel, an attacker who intercepts the delegation information can convert Bob's signature on any message conforming the warrant into the proxy signature in which Bob is the proxy signer regardless of Bob's will. Conversely, he can get a proxy signer's valid signature on the message m_P by removing the last term of the proxy signature. As another reason to establish the secure channel, if some attacker intercepts all the signed warrant delivered to Bob and then changes the term $(h(m_P, r_P) \cdot s_A)$ into the one among the intercepted warrants, he can change the original signer.

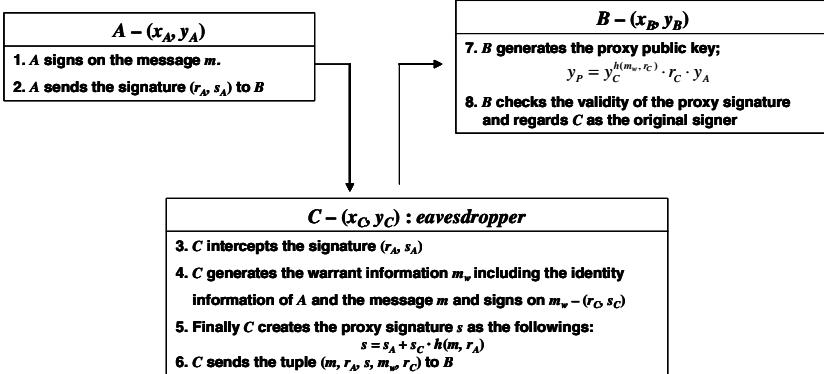


Fig. 1. Attack scenario against proxy protection

6 Comparison and Revisions

We compare the above four schemes and the OTO scheme [6] with respect to the providing functions and efficiency in Table 1. The providing functions mean the security requirements satisfied by each scheme and the efficiency means the necessity of the secure channel. In the MUO scheme the proxy signer can misuse the unlimited signing capability and transfer to others since the warrant does not include the identity information of the proxy signer or the limit of the delegated signing capability. The PH scheme uses a 3-pass blind signature scheme to protect the proxy private key from the original signer. However, since the proxy private key is the ordinary signature on the identity of the proxy signer using Schnorr's scheme, a malicious one can generate the proxy signature where the original signer is himself and the proxy signer is anyone who he wants. Thus their scheme does not satisfy the requirements R2 and R4 and so is not proxy-protected. The proxy signer can also misuse the signing capability. the OTO scheme follows the same process as the PH scheme, i.e., the proxy signer uses an ordinary signature as a proxy private key, with a difference that it uses a secure channel and the warrant contains the limit of the delegated rights. The KPW scheme is a proxy-protected one and satisfies all the security requirements in the strict sense as mentioned above. Furthermore, in case that we remove the secure channel, their scheme has no effect on the security while other schemes break out several problems. Actually, the removal of the secure channel improves the efficiency greatly. The LKK scheme is not a proxy-protected one since an attacker is able to change any valid signature into the proxy signature and the malicious original signer can obtain the valid signature from the proxy signature.

We propose the revisions of the above schemes [5, 3] to prevent the weaknesses arising in the situation removing the secure channel and provide the same functions with the original scheme with the same computational complexity.

Table 1. Comparison of proxy signatures

Features	MUO	PH	KPW	OTO	LKK
Secure Channel	Need	Not Need	Not Need	Need	Need
Proxy-Protected	O	X*	O	X	X*
R1 Verifiability	O	O	O	O	O
R2 Unforgeability	O	X	O	X	X*
R3 Identifiability	O	O	O	O	O
R4 Undeniability	O	X	O	X	O
R5 Prevention of Misuse	X	X	O	O	O
Non-Transferability	X	O	O	O	O

* Indicates the different assertion from the author's claim.

6.1 Revision of the MUO Scheme

In order that the original signer designates the proxy signer in advance, we modify the proxy key generation stage as follows:

1'. Generation of the proxy key: Bob gets a proxy key pair (x_P, y_P) through the following steps.

1. An original signer Alice generates a random number $k \in \mathbb{Z}_q^*$ and computes $K = g^k \pmod p$. After that, she calculates

$$s_A = x_A + k \cdot y_B \pmod q,$$

and then sends (s_A, K) to a proxy signer Bob.

2. Bob checks

$$g^{s_A} \stackrel{?}{=} y_A \cdot K^{y_B} \pmod p.$$

If it is passed, Bob computes the proxy private key as

$$x_P = s_A + x_B \cdot y_A \pmod q.$$

The proxy public key, which is used in the verification stage, is generated as follows:

$$y_P = g^{x_P} = y_A \cdot K^{y_B} \cdot y_B^{y_A}.$$

In this revision having no secure channel, the original signer can designate the proxy signer and so once the proxy signature is generated by the proxy signer anyone cannot change the original signer.

6.2 Revision of the LKK Scheme

The weakness of the LKK scheme results from the characteristic of the proxy private key and Schnorr's signature scheme. We revise the scheme at the proxy key generation stage as follows:

1'. Generation of the proxy key: Bob gets a proxy key pair (x_P, y_P) through the following steps.

1. Alice chooses at random $k_A \in \mathbb{Z}_q^*$ and computes $r_A = g^{k_A} \bmod p$ and $s_A = k_A + x_A \cdot h(m_w, r_A) \bmod q$. Then she sends (m_w, r_A, s_A) to Bob.
2. Bob verifies the validity of the signature on m_w . If the signature is valid, Bob chooses a random number k_P and computes his proxy key pair (x_P, y_P) such that $x_P = r_P \cdot x_B + s_A$ and $y_P = g^{x_P} \bmod p$ for $r_P = g^{k_P} \bmod p$.

Then the proxy public key is generated as follows:

$$y_P = y_A^{h(m_w, r_A)} \cdot r_A \cdot y_B^{r_P}.$$

In order to show that this scheme overcomes the weakness of the LKK scheme, we must show that Alice and Bob cannot create the proxy signature by the reversing-order method. Let's see the following two equations:

1. $s_1 = k_P + x_B \cdot h(m_P, r_P)$
2. $s_2 = k_P + r_P \cdot x_B \cdot h(m_P, r_P)$

where the conditions for each parameter are the same as the above scheme. s_1 is the Schnorr's signature on the message m_P and we cannot obtain s_1 in the polynomial time [8]. Through the comparison between the first equation and the second, we can know that finding s_2 is as difficult as the first and anyone cannot induce s_2 from s_1 . Here, in order that an attacker creates the proxy signature generated by our revised scheme from the valid signature generated by the Schnorr's signature scheme, the attacker should induce the second equation from the first. Thus, the proxy signature is not generated by the reversing-order method.

Consequently, our revised scheme overcomes the weakness of the previous schemes and we do not require the secure channel any more for the delivery of the signed warrant.

7 Conclusion

In this paper, we analyzed and compared several proxy signature schemes. The comparison was given in Table 1. We also discussed the necessity of secure channel in proxy signatures. All known proxy signatures used a secure channel to deliver a proxy certificate except one which used a 3-pass weak blind signature. However, one of them appeared to be secure without the secure channel. As a further work, it would be interesting to devise a security model on proxy signatures and give a rigorous proof based on this.

References

- [1] Javier Herranz and German Saez “Fully Distributed Proxy Signature Schemes”, <http://eprint.iacr.org/>, 2002.
- [2] S. Kim, S. Park, and D. Won, “Proxy signatures, revisited”, In *Pro. of ICICS'97, International Conference in Information and Communications Security*, Springer, Lecture Notes in Computer Science, LNCS1334, pages 223-232, 1997. **68, 69, 72**
- [3] Byoungcheon Lee, Heesun Kim, Kwangjo Kim, “Strong Proxy Signature and its Applications”, SCIS2001, vol 2/2 pp 603-608, Jan.23 26, 2001. **68, 69, 72, 75, 76**
- [4] Byoungcheon Lee, Heesun Kim, and Kwangjo Kim “Secure Mobile Agent using Strong Non-designated Proxy Signature”, Proc. of ACISP2001, LNCS, Springer Verlag Vol.2119, pp.474-486, 2001.
- [5] M. Mambo, K. Usuda, and E. Okamoto, “proxy signature: Delegation of the power to sign messages”, In *IEICE Trans. Fundamentals*, Vol. E79-A, No. 9, Sep., pp. 1338-1353, 1996. **68, 69, 70, 76**
- [6] Takeshi Okamoto, Mitsuru Tada, and Eiji Okamoto, “Extended Proxy Signatures for Smart Cards”, In *Pro. of ISW'99* Springer, Lecture Notes in Computer Science, LNCS 1729, pp. 247-258, 1999. **69, 76**
- [7] H. Petersen and P. Horster, “Self-certified keys - Concepts and Applications”, In *Proc. Communications and Multimedia Security'97*, pages 102-116, Chapman and Hall, 1997. **69, 71**
- [8] D. Pointcheval and J. Stern, “Security proofs for signatures”, In *Advances in Cryptology: Eurocrypt'96*, pages 387-398, Springer, 1996. **74, 78**

Invisibility and Anonymity of Undeniable and Confirmer Signatures

Steven D. Galbraith^{1,*} and Wenbo Mao^{2,**}

¹ Mathematics Department, Royal Holloway University of London
Egham, Surrey TW20 0EX, UK
Steven.Galbraith@rhul.ac.uk

² Mathematics, Cryptography and Security Group
Hewlett-Packard Laboratories, Bristol
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK
wm@hplb.hpl.hp.com

Abstract. Traditionally, the strongest notion of security for undeniable and confirmer signatures is invisibility under adaptive attacks. This security property was promoted by Camenisch and Michels and they provided schemes with this property. Gennaro, Krawczyk and Rabin (GKR) developed an RSA-based scheme which is much more efficient than the schemes of Camenisch and Michels, but it does not have invisibility. We give an RSA-based scheme which is as efficient as the GKR scheme, and which has invisibility.

We suggest that anonymity is the most relevant security property for undeniable and confirmer signatures. We give a precise definition of anonymity for undeniable and confirmer signatures in the multi-user setting and show that anonymity and invisibility are closely related. Finally, we show that anonymity can be achieved even when the parties use completely different cryptographic primitives.

Keywords: Undeniable signatures, confirmer signatures, RSA, invisibility, anonymity.

1 Introduction

Undeniable signatures [4, 5] are public key digital signatures which cannot be verified without interacting with the signer. Confirmer signatures [7] are undeniable signatures where signatures may also be verified by interacting with an entity called the confirmer who has been designated by the signer. Implicit in these notions is the principle that validity of a signature cannot be determined without some interaction. Undeniable and confirmer signatures have been used in various applications, including auctions [16, 17].

* This author thanks Hewlett-Packard Laboratories, Bristol and the EPSRC for support.

** This author's research is partially funded by the EU Fifth Framework Project IST-2001-324467 "CASENET".

The strongest notion of security in the literature for undeniable and confirmer signatures is that of ‘invisibility’, which was introduced by Chaum, van Heijst and Pfitzmann [6]. This is essentially the inability to determine whether a given message-signature pair is valid for a given user. In [6] invisibility is defined in terms of simulatability. In [3] this notion is phrased in terms of distinguishing whether a signature s corresponds to a message m_0 or m_1 .

The history of undeniable and confirmer signatures is no different from the history of other public key techniques: the strong notions of security were introduced after the early schemes had been invented, and the original schemes did not provide these security properties. Camenisch and Michels [3] gave a general method to obtain a secure confirmer signature scheme out of any sufficiently secure signature scheme and any sufficiently secure encryption scheme. Their methods can be used to obtain secure schemes based on RSA or discrete logarithms. The drawback of [3] is that efficiency is sacrificed to obtain security.

Much more efficient RSA-based undeniable and confirmer signatures were developed by Gennaro, Krawczyk and Rabin [10] and Galbraith, Mao and Paterson [9]. However, these schemes were not developed to provide invisibility. One of the main aims of this paper is to present a new RSA-based scheme which has the invisibility security property (i.e., is as secure as the scheme of [3]) and yet which is essentially as efficient as the schemes of [9, 10]. Our scheme is the most efficient in the literature among those which provide invisibility.

We believe that anonymity rather than invisibility should be considered as the main security property for undeniable and confirmer signatures in the multi-user setting. Informally, this security property is as follows. Imagine a system with n users and suppose an adversary is given a valid message-signature pair and is asked to determine which user generated the signature. By running signature confirmation or denial protocols with a given user (or their designated confirmer) one can determine whether or not the user generated the signature. An undeniable or confirmer signature scheme has the anonymity property if it is infeasible to determine whether a user is or is not the signer of the message without interacting with that user or with the $n - 1$ other users. A more precise definition of anonymity is given in Definitions 3 and 4 where the problem is distilled down to the case of two users. We show that anonymity and invisibility are equivalent notions for schemes with certain additional properties.

General techniques show that anonymity can be achieved (e.g., by combining the results of Okamoto [14] and Bellare, Boldyreva, Desai and Pointcheval [2]). However, our schemes are significantly more efficient than schemes which would arise using these general techniques.

For confirmer signature schemes we stress that we do not study the issue of whether the signature reveals who the designated confirmer is (though our solutions do provide anonymity for the confirmer).

1.1 Plan of the Paper

In Section 2 we clarify what we mean by undeniable and confirmer signature schemes. In Section 3 we generalise the notion of invisibility to one which we believe is more natural in the multi-user case. In Section 4 we give a precise

definition for anonymity and prove two of our main results: that anonymity and generalised invisibility are essentially equivalent.

Sections 5, 6 and 7 form the technical heart of the paper. We give two attacks on the anonymity of the RSA-based undeniable and confirmor signature schemes of Gennaro, Krawczyk and Rabin [10] and Galbraith, Mao and Paterson [9]. We give a new RSA-based undeniable/confirmor signature scheme. Our main result is Theorem 5 which shows that our scheme has the anonymity property. The security of our scheme against forgery is shown in Appendix C.

In Section 8, we discuss anonymity in the extremely general situation where participants may use completely different undeniable and confirmor signature schemes. We argue that anonymity can be obtained even in this setting.

2 Undeniable and Confirmor Signature Schemes

An *undeniable signature scheme* consists of two algorithms, namely Gen and Sign, and two protocols, namely Confirm and Deny. For every choice of the security parameter k there is a public-key space \mathcal{K} , a message space \mathcal{M} and a signature space \mathcal{S} . For our applications we stress that the space \mathcal{S} must depend only on the security parameter k and not on a specific public key.

Gen is a randomised algorithm which takes as input a security parameter k and outputs a public key $pk \in \mathcal{K}$ and a corresponding secret key sk .

Sign is an algorithm (in our case it must be randomised) which takes as input a secret key sk and a message $m \in \mathcal{M}$ and outputs a signature $\text{Sign}_{sk}(m) \in \mathcal{S}$.

In general, there are many valid signatures for any pair $(pk, m) \in \mathcal{K} \times \mathcal{M}$.

Confirm is a protocol between a signer and a verifier which takes as input a message $m \in \mathcal{M}$, a signature $s \in \mathcal{S}$ and a (certified) public key pk and allows the signer to prove to a verifier that the signature s is valid for the message m and the key pk . If the verifier has a suitable public key then the proof may be taken to be a non-interactive, designated-verifier proof [11].

Deny is a protocol which takes $m \in \mathcal{M}$, $s \in \mathcal{S}$ and $pk \in \mathcal{K}$ and allows a signer to prove to a verifier that the given signature is not valid for that key. In some schemes the denial protocol is the same as the confirmation protocol.

A *confirmor signature scheme* is essentially the same as above, except the role of confirmation and denial can also be performed by a third party called a ‘confirmor’. The significant modification is that the algorithm Gen now produces a confirmation key ck which is needed for the Confirm and Deny protocols.

The literature on confirmor signatures is inconsistent on whether the original signer has the ability to confirm and/or deny signatures. Camenisch and Michels [3] claim that it is undesirable for signers to be able to confirm or deny their signatures. We have a contrary opinion, that it is important for signers to be able to confirm and/or deny signatures (e.g., to clear their name by denying signatures that are not genuine). The schemes of [3, 7, 13] do not allow users to deny signatures, whereas the schemes of [4, 5, 9, 10] do allow this. In any

case, these distinctions have no bearing on the discussion of the invisibility and anonymity of the schemes.

The following properties will be used later. Property A is essentially that, for a fixed key and varying messages, signatures look random. Property B is essentially that, for a fixed message and varying keys, signatures look random.

Property A: Let k be any value of the security parameter. Let (pk, sk) be any output of the Gen algorithm for the security parameter k . Consider the uniform distribution on \mathcal{M} . Then the distribution on \mathcal{S} corresponding to the random variable $\text{Sign}_{sk}(m)$ is indistinguishable from uniform.

Property B: Let k be any value of the security parameter and let $m \in \mathcal{M}$ be an arbitrary message. Consider the distribution on \mathcal{K} induced by the randomised algorithm Gen. Then for pk chosen at random from \mathcal{K} according to this distribution (with the corresponding secret key sk), the distribution on \mathcal{S} corresponding to the random variable $\text{Sign}_{sk}(m)$ is indistinguishable from uniform.

A typical example in our situation of a distribution which is indistinguishable from uniform is the following. Let \mathcal{S} be the set of all binary strings of length $2k$, let N be a k -bit RSA modulus, and consider the uniform distribution on the set $\mathcal{S}' = \{s \in \mathcal{S} : \gcd(s, N) = 1\}$ (where a binary string is interpreted as an integer in the natural way). Then, given N and an algorithm which outputs polynomially many randomly sampled elements from \mathcal{S} or \mathcal{S}' , it is infeasible to determine which of the two sets is being sampled.

3 Generalised Invisibility

We give a definition of invisibility which is more suitable for our purposes. There is one subtlety in this definition: For the schemes we will consider, signatures are padded to a fixed bitlength in a way which is malleable by an adversary. For example, with our RSA-based scheme, a signature is naturally a number modulo N (where N is part of a user's public key) and this number is extended to a longer bitstring by adding a multiple of N . Now, this process is malleable in the sense that an adversary can add or subtract N from a given bitstring to obtain another, equally valid, signature. This leads to a trivial adaptive attack on the scheme.

More precisely, signatures lie in equivalence classes whose structure is known to all users, and which depend on the particular public key. Hence we cannot allow an adaptive adversary to initiate confirm or denial protocols on any representative of the equivalence class of the challenge signature. Fortunately, these equivalence relations are always easily computed and so one can recognise if an adversary tries to make a query of this form.

Definition 1. *Let $(\text{Gen}, \text{Sign}, \text{Confirm}, \text{Deny})$ be an undeniable or confirmer signature scheme. An adversary \mathbf{D} is said to be an **invisibility distinguisher** under an adaptive chosen message attack if it behaves as follows.*

Fix a value of the security parameter k (this fixes the signature space \mathcal{S}). Let $(pk, sk) \leftarrow \text{Gen}(1^k)$ be a key pair. The input to \mathbf{D} is pk . The distinguisher \mathbf{D} is permitted to interact with the hash function oracle(s), to obtain signatures on messages of its choice and to run signature verification and denial protocols (with the signer or a confirmor as appropriate) on elements of \mathcal{S} of its choice. At some point \mathbf{D} constructs a message m and requests a challenge s . The challenge s depends on the outcome of a hidden coin toss. If the hidden bit b is 0 then $s = \text{Sign}_{sk}(m)$ and if the hidden bit is 1 then s is chosen uniformly at random from the signature space \mathcal{S} . The interaction with the cryptosystem continues with the exception that verification and denial protocols cannot be executed on any pair (m, s') in the equivalence class of the challenge message-signature pair (m, s) . The output of \mathbf{D} is a guess b' for the hidden bit b .

A distinguisher \mathbf{D} with output b' is said to have **advantage** $\text{Adv}(\mathbf{D}) = \epsilon(k)$, if with probability at least $1/2 + \epsilon(k)$, we have $b' = b$.

Definition 2. An undeniability or confirmor signature scheme has **invisibility** if there is no polynomial time distinguisher \mathbf{D} as in Definition 1 which has non-negligible advantage (i.e., given any polynomial $p(k)$ we have $\epsilon(k) < 1/p(k)$ for sufficiently large k).

We now show that, for certain schemes, generalised invisibility is equivalent to the definition of invisibility given in [3]. First we recall the definition of invisibility given in [3]: the distinguisher is almost identical to Definition 1 above except that it presents two messages m_0, m_1 and the challenge is $s = \text{Sign}_{sk}(m_b)$ where b is the hidden bit.

The proofs of these results are given in Appendix A. Note that these results hold in the standard model of computation and that the security reductions require essentially no added computation.

Theorem 1. Let \mathcal{U} be an undeniability or confirmor signature scheme. If \mathcal{U} has invisibility in the sense of Definition 2, then \mathcal{U} has invisibility in the sense of [3].

Theorem 2. Let \mathcal{U} be an undeniability or confirmor signature scheme which satisfies Property A above. If \mathcal{U} has invisibility in the sense of [3], then \mathcal{U} has invisibility in the sense of Definition 2.

4 Anonymity

We give a rigorous definition for anonymity. The first step is to distill the problem down to the case of just two users (a scheme with the anonymity property for two users can easily be shown to be secure in the case of n users¹). Note that if a signature is known to be valid for some user then the identity of the signer can be obtained by executing a signature confirmation protocol with that user, or by executing a signature denial protocol with the other user.

¹ For general results n must be bounded by a polynomial in the security parameter, but for some specific schemes this constraint may be relaxed.

Definition 3. Let $(\text{Gen}, \text{Sign}, \text{Confirm}, \text{Deny})$ be an undeniable or confirming signature scheme. An adversary \mathbf{D} is said to be an **anonymity distinguisher** under an adaptive chosen message attack if it behaves as follows.

Fix a value of the security parameter k (this fixes the signature space \mathcal{S}). Let $(pk_0, sk_0) \leftarrow \text{Gen}(1^k)$ and $(pk_1, sk_1) \leftarrow \text{Gen}(1^k)$ be two key pairs. The input to \mathbf{D} is the pair (pk_0, pk_1) . The distinguisher \mathbf{D} is permitted to interact with the hash function oracle(s), to obtain signatures on messages of its choice and to run signature verification and denial protocols (with the signer or a confirming as appropriate) with respect to either of the two public keys on elements of \mathcal{S} of its choice. At some point \mathbf{D} constructs a message m and requests a challenge signature $s \leftarrow \text{Sign}_{sk_b}(m)$ where the bit $b \in \{0, 1\}$ is hidden from \mathbf{D} . The interaction with the cryptosystem continues with the exception that verification and denial protocols cannot be executed on any pair (m, s') in either of the two equivalence classes (i.e., with respect to pk_0 or pk_1) of the challenge message-signature pair (m, s) . The output of \mathbf{D} is a guess b' for the hidden bit b .

A distinguisher \mathbf{D} with output b' is said to have **advantage** $\text{Adv}(\mathbf{D}) = \epsilon(k)$, if with probability at least $1/2 + \epsilon(k)$, we have $b' = b$.

Definition 4. An undeniable or confirming signature scheme has **anonymity** if there is no polynomial time distinguisher \mathbf{D} as in Definition 3 which has non-negligible advantage (i.e., given any polynomial $p(k)$ we have $\epsilon(k) < 1/p(k)$ for sufficiently large k).

We now show that anonymity and invisibility are essentially equivalent. Note that both Theorems below are proved in the standard model of computation and that the security reductions require essentially no added computation.

Theorem 3. Let \mathcal{U} be an undeniable or confirming signature scheme which has Property B. If \mathcal{U} has anonymity, then \mathcal{U} has invisibility.

Proof. First assume that the distribution in Property B is precisely uniform. Let \mathbf{D}_I be an invisibility adversary, we will create an anonymity adversary \mathbf{D}_A from \mathbf{D}_I . The input to \mathbf{D}_A is (pk_0, pk_1) and we run \mathbf{D}_I on pk_0 . The queries made by \mathbf{D}_I can all be passed on as \mathbf{D}_A queries. Note that pk_1 is independent of the view of \mathbf{D}_I .

Eventually \mathbf{D}_I produces a message m and requests a challenge. The message m is output by \mathbf{D}_A and the challenge s is received. Recall that s is $\text{Sign}_{sk_0}(m)$ in the case $b = 0$ and is $\text{Sign}_{sk_1}(m)$ in the case $b = 1$. In the case $b = 1$, since pk_1 is uniformly chosen at random from \mathcal{K} , our assumption implies that s is a uniformly random element of \mathcal{S} . In other words, the value s is compatible with the game \mathbf{D}_I is designed to play.

Further queries by \mathbf{D}_I are passed on by \mathbf{D}_A . Finally, \mathbf{D}_I outputs a guess b' for b . This bit is used by \mathbf{D}_A as its guess for the hidden bit b . It is clear that

$$\text{Adv}(\mathbf{D}_A) = \text{Adv}(\mathbf{D}_I).$$

Now consider the case where the distribution in Property B is indistinguishable from uniform. Let \mathbf{D}_I be an invisibility distinguisher in the sense of Definition 1. If the advantage of \mathbf{D}_I in the game described earlier in the proof is

non-negligibly different from the advantage of \mathbf{D}_I in a real attack then \mathbf{D}_I can be easily transformed into a distinguisher for Property B. It follows that

$$\text{Adv}(\mathbf{D}_A) \approx \text{Adv}(\mathbf{D}_I).$$

where \approx means ‘equal up to negligible terms’. This completes the proof. \square

Theorem 4. *Let \mathcal{U} be an undeniable or confirmer signature scheme which has invisibility, then \mathcal{U} has anonymity.*

Proof. Let \mathbf{D}_A be an anonymity distinguisher. We will construct an invisibility distinguisher \mathbf{D}_I from \mathbf{D}_A . The input to \mathbf{D}_I is a key pk_0 . We execute Gen to construct another public key pk_1 (note that the secret key sk is known to \mathbf{D}_I).

Flip a coin to obtain a bit b' . If $b' = 0$ then run \mathbf{D}_A on the pair (pk_0, pk_0) and if $b' = 1$ then run \mathbf{D}_A on the pair (pk_1, pk_0) .

Queries made by \mathbf{D}_A are answered in the obvious way: queries with respect to pk_0 are passed on by \mathbf{D}_I , and queries with respect to pk_1 are handled using knowledge of the secret key.

Eventually \mathbf{D}_A produces a message $m \in \mathcal{M}$ and requests a challenge. We use m as the challenge for \mathbf{D}_I and receive a value $s \in \mathcal{S}$ which is $\text{Sign}_{pk_0}(m)$ in the case $b = 0$ or is a uniformly random element of \mathcal{S} in the case $b = 1$. The challenge s is passed to \mathbf{D}_A . In the case $b = 0$ we have that s is a valid signature on m for pk_0 and with overwhelming probability is not a valid signature on m for pk_1 . In the case $b = 1$ we have that s is a random element of \mathcal{S} and so, with overwhelming probability, s is not a valid signature on m for either public key. The queries by \mathbf{D}_A continue to be handled as before.

Finally, \mathbf{D}_A outputs a guess b'' . If $b'' = b'$ then output 0 as the guess for the hidden bit b , and if $b'' \neq b'$ then output 1.

It remains to compute the advantage of \mathbf{D}_I . First note that in the case $b = 0$

$$\text{Adv}(\mathbf{D}_A) = \Pr(b'' = b') = \frac{1}{2} + \epsilon.$$

Now, the advantage of \mathbf{D}_I is

$$\text{Adv}(\mathbf{D}_I) = \Pr(b'' = b'|b = 0) \Pr(b = 0) + \Pr(b'' \neq b'|b = 1) \Pr(b = 1)$$

and $\Pr(b'' \neq b'|b = 1) \approx \frac{1}{2}$ (where, again, \approx means ‘equal up to negligible factors’ since there is the negligible chance that the random s is a valid signature on m) since, in the case $b = 1$, the hidden bit b' is independent of s . It follows that

$$\text{Adv}(\mathbf{D}_I) \approx \left(\frac{1}{2} + \epsilon\right) \frac{1}{2} + \frac{1}{2} \frac{1}{2} - \frac{1}{2} = \frac{1}{2}\epsilon$$

and the result follows. \square

We emphasise that Theorems 1 and 4 mean that it is enough to prove that an undeniable or confirmer signature scheme has invisibility in the sense of Definition 2 to deduce both anonymity, and invisibility in the sense of Camenisch and Michels [3].

5 Undeniable Signatures Based on RSA

Gennaro, Krawczyk and Rabin [10] described an undeniable/confirming signature scheme based on RSA. In their case the signature for a message m is s where $s \equiv \overline{m}^d \pmod{N}$ and \overline{m} is a one-way encoding. The signature may be verified by proving that $s^e \equiv \overline{m} \pmod{N}$ where the verification exponent e is known to the confirming party. The verification exponent e is fixed by publishing values $g = h^e \pmod{N}$. The original scheme required moduli which are products of safe primes and it was generalised to arbitrary RSA moduli by Galbraith, Mao and Paterson [9], who also gave a more efficient denial protocol.

To handle adaptive attacks on invisibility/anonymity it is clear that the one-way encoding must also be randomised. Hence, a signature becomes a pair (r, s) where r is random and $s \equiv H(m, r)^d \pmod{N}$ where $H(m, r)$ is the randomised one-way encoding.

5.1 Attacks on Invisibility and Anonymity

We show that the schemes of [9, 10] do not have invisibility or anonymity even under passive attacks. Note that these security properties were not claimed for either scheme.

Since d is odd it follows that the Jacobi symbols $(\frac{s}{N})$ and $(\frac{H(m,r)}{N})$ are equal. Hence, given a pair $(H(m, r), s)$ and a user's public key N , if $(\frac{s}{N}) \neq (\frac{H(m,r)}{N})$ then the signature is not valid for that user. This shows that the scheme does not have invisibility. A similar attack in the multi-user setting (testing the condition for all users' keys N_i) shows that the scheme does not have anonymity.

Another attack on anonymity of RSA-based schemes arises since all users must have different moduli N . If a signature s satisfies $s \geq N_i$ for some modulus N_i in the system then user i is not the signer of the message. This reduces the number of possibilities for the signer.

5.2 Preventing the Jacobi Symbols Attack

To prevent the Jacobi symbols attack it is tempting to restrict to choices for r such that $H(m, r)$ is a quadratic residue in \mathbb{Z}_N^* (and so s would also be a quadratic residue in \mathbb{Z}_N^*). This does not work since anonymity could still be broken by computing $(\frac{s}{N})$ and eliminating those for which the value is -1 .

Another unsuccessful solution is the following. Since we do not want $(\frac{H(m,r)}{N})$ and $(\frac{s}{N})$ to be correlated a natural approach is to set $s = \xi H(m, r)^d \pmod{N}$ where ξ is a random element of order 2. The problem with this approach is that the confirming party (who has e) can recover ξ and hence factorise N and forge signatures.

Instead, our solution involves square roots, as these can be chosen to have arbitrary Jacobi symbol. We require that N is a Blum integer (i.e., a product $N = pq$ where $p \equiv q \equiv 3 \pmod{4}$) and so for every $a \in \mathbb{Z}_N^*$ with $(\frac{a}{N}) = +1$ it follows that either a or $-a$ is a square. The idea is to obtain a value $H'(m, r)$

which depends on both $H(m, r)$ and N , and is such that $\pm H'(m, r)$ is a square modulo N . We define signatures by

$$s = \left(\sqrt{\pm H'(m, r)} \right)^d \pmod{N}$$

where the square-root is randomly chosen from among the four possibilities. The verification operation is to check that $s^{2e} \equiv \pm H'(m, r) \pmod{N}$ where the verification exponent e is only known to the signer and confirmers.

At first sight one might think this protocol to be insecure since two signatures s and s' on the same message leak a square-root of unity s/s' . However, this scenario never arises, as the value $H(m, r)$ is randomised.

We give further details on how $H'(m, r)$ is constructed. First we construct a one-way randomised padding $H(m, r)$ of the message using the method of Bellare and Rogaway [1]. One consequence of using a randomised padding scheme in the context of undeniable signatures is that it is necessary to transmit the value r as part of the signature. This is because, unlike with standard RSA signatures, the value $H(m, r)$ is not recovered by the verifier as part of the signature verification process.

We now must associate a value $H'(m, r)$ which is a square (or minus a square). To achieve this we insist that N be a Blum integer. We now provide a deterministic algorithm which takes $H(m, r)$ and N and produces an element $H'(m, r) \pmod{N}$ with Jacobi symbol +1. Let F be a hash function from k -bit strings to k -bit strings.² Set $i = 0$ and $v_0 = H(m, r)$. If $(\frac{v_0}{N}) \neq +1$ then iterate $v_{i+1} := F(v_i)$ and continue until an element v_t with Jacobi symbol +1 is obtained. Define $H'(m, r) = v_t$. This process is expected to terminate within a few iterations in practice.

5.3 Ensuring that Signature Length Does Not Reveal the Signer

A solution to this problem is for users to enlarge any values $s \pmod{N}$ to a fixed bitlength by adding a suitable multiple of N (see Section 3.1 of Desmedt [8] and also [2]). This padding removes any information about the size of N and does not interfere with the reduction of the value modulo N .

More precisely, let k be the bitlength of the modulus N . We will extend signatures to be bitstrings of length $2k$. Since N is an RSA modulus, s is indistinguishable from being uniformly distributed in \mathbb{Z}_N . Uniformly choose an integer in the range $0 \leq t < (2^{2k} - s)/N$ and let $s' = s + tN$ (there are n or $n - 1$ such t where $n = \lfloor (2^{2k} - 1)/N \rfloor$). Then $0 \leq s' < 2^{2k}$ and it follows that s' is indistinguishable from a random $2k$ -bit string.

² In the security proof we model F as a random oracle to ensure that the simulation is indistinguishable from a real game. In practice we believe that F could be the function $x \mapsto x + 1$ without loss of security.

6 The New RSA-Based Scheme

We present the scheme in the case of moduli which are products of safe primes (a safe prime is a prime p such that $p' = (p - 1)/2$ is also prime). The modification of the scheme to general Blum integer moduli is straightforward following [9], but there are subtle reasons why our proof of invisibility does not apply in the general case (see the footnote in the proof of Theorem 5). The scheme is as follows for security parameter k .

System parameters: Let k be the security parameter and let k_0 be derived from k (e.g., $k_0 = \lfloor k/3 \rfloor$). The space \mathcal{S} is the set of bitstrings of length $k_0 + 2k$. Hash functions G_0, G_1 and G_2 as in [1] must be specified. A hash function F which takes k -bit strings to k -bit strings must be specified.

The key space \mathcal{K} is a subspace of the set of binary strings of length $3k$ corresponding to triples (N, g, h) with certain properties given below.

Key generation: A signer chooses two primes p and q whose product is a k -bit integer, such that $p' = (p - 1)/2$ and $q' = (q - 1)/2$ are prime. The signer sets $N = pq$ and chooses $e, d \in \mathbb{Z}$ such that $ed \equiv 1 \pmod{\varphi(N)}$. The signer chooses $g \in \mathbb{Z}_N^*$ (such that, with overwhelming probability, $\{a^2 : a \in \mathbb{Z}_N^*\} \subseteq \langle g \rangle \subseteq \mathbb{Z}_N^*$) and sets $h = g^d \pmod{N}$.

The signer registers the public key (N, g, h) with a certificate authority. The signer sends e to the designated confirmor (if there is one) via a secure channel.

Signing: To sign a message m the signer constructs the randomised padding value $H(m, r)$ as in [1] (i.e., chooses a k_0 -bit string r at random and computes $w = G_0(m, r), r^* = r \oplus G_1(w), \gamma = G_2(w)$ and $H(m, r) = w \| r^* \| \gamma$). Set $v = H(m, r)$. While $(\frac{v}{N}) \neq +1$ then set $v = F(v)$ and repeat. Set $H'(m, r) = v$. The signer computes

$$s = \left(\sqrt{\pm H'(m, r)} \right)^d \pmod{N}$$

where the sign is chosen so that $\pm H'(m, r)$ is a square modulo N and where the square-root is chosen randomly among the four possibilities. The signer enlarges s to a bitstring s' of length $2k$ by adding a suitable random multiple of N as in section 5.3. The signature on m is the $k_0 + 2k$ bit binary string (r, s') .

Confirm/Deny: To confirm or deny a signature the signer or confirmor executes non-interactive, designated verifier versions of proofs³ like those in [9] which prove knowledge of an integer e such that $g \equiv h^e \pmod{N}$ and $s^{2e} \stackrel{?}{=} \pm H'(m, r) \pmod{N}$. Note that all users can compute $H'(m, r)$ given m, r and N .

Equivalence classes: Given a valid signature (r, s) then $(r, \pm s + tN)$ for values $t \in \mathbb{Z}$ such that $0 \leq \pm s + tN < 2^{2k}$ is also a valid signature. This equivalence

³ These proof transcripts must be encrypted when sent to the verifier if anonymity is to be preserved.

class is used in Definition 3. In case of confusion we stress that this equivalence class is smaller than $\{(r, \xi s + tN) : \xi^2 \equiv 1 \pmod{N}\}$ which is the set of all valid signatures for message m with randomness r .

To obtain the security result it is necessary that executions of the confirm and deny protocols can be simulated in the random oracle model. This is not possible with interactive proofs so we must use non-interactive proofs. To maintain the security of the system (i.e., so that proofs cannot be transferred to other users) it is necessary to use designated-verifier proofs [11]. It is standard that such proofs can be simulated in the random oracle model. For further details see Jakobsson et al [11, 12].

We note that this scheme is secure against forgery under adaptive attacks (see Appendix C).

7 Invisibility of Revised RSA-Based Undeniable and Confirming Signatures

We now prove that the scheme described in the previous section has the invisibility property. Our proof only applies to the case of RSA moduli which are a product of safe primes. We write $\text{ord}(g)$ for the order of an element modulo N (i.e., the smallest positive integer n such that $g^n \equiv 1 \pmod{N}$) and we write $\langle g_1, \dots, g_m \rangle$ for the subgroup generated by g_1, \dots, g_m .

We will show that the invisibility (and hence, anonymity) of the system depends on the hardness of the following computational problem:

Composite Decision Diffie-Hellman Problem (CDDH): Let N be a product of two safe primes (i.e., $N = pq$ with p and q both primes such that $p' = (p - 1)/2$ and $q' = (q - 1)/2$ are both prime). Consider the two sets

$$\mathcal{T} = \{(g, h, u, v) \in (\mathbb{Z}_N^*)^4 : \text{ord}(g) = \text{ord}(h) = 2p'q', h \in \langle g \rangle, \langle g, v \rangle = \mathbb{Z}_N^*\}$$

and

$$\begin{aligned} \mathcal{T}_{\text{CDDH}} = & \{(g, h, u, v) \in \mathcal{T} : h \equiv g^d \pmod{N} \text{ for some } d \text{ coprime to } \varphi(N), \\ & v \equiv \xi u^d \pmod{N} \text{ for some } \xi \in \mathbb{Z}_N^* \text{ of order 2}\} \end{aligned}$$

with the uniform distribution on each. The CDDH problem is to distinguish these two distributions. More precisely, a CDDH oracle with advantage ϵ is an algorithm \mathcal{A} with input (N, g, h, u, v) such that

$$\begin{aligned} & \Pr(\mathcal{A}(N, g, h, u, v) = 1 | (g, h, u, v) \in \mathcal{T}_{\text{CDDH}}) \\ & - \Pr(\mathcal{A}(N, g, h, u, v) = 1 | (g, h, u, v) \in \mathcal{T}) = \epsilon. \end{aligned}$$

The CDDH assumption is that there is no CDDH oracle which runs in polynomial time and which has non-negligible advantage.

Belief in the CDDH assumption is given by the fact that if the factorisation of N is known then the problem reduces to the DDH problem modulo the prime factors of N .

Theorem 5. *Consider the RSA-based undeniable/confirmer signature scheme above with moduli which are products of safe primes. In the random oracle model (i.e., G_0, G_1, G_2 and F are random oracles) then the scheme has invisibility if the composite decision Diffie-Hellman problem is hard.*

The proof of this theorem is given Appendix B.

Corollary 1. *The RSA-based undeniable/confirmer signature scheme above has anonymity in the random oracle model.*

Note that it remains an open problem to obtain invisibility and anonymity in the case of moduli which are not necessarily safe prime products.

8 Anonymity between Signatures of Different Schemes

We have shown that our RSA-based scheme has invisibility when \mathcal{S} is the space of binary strings of length $k_0 + 2k$. In the full version of the paper we give a finite field based scheme which has invisibility with the same signature space.

Theorem 4 implies that a given undeniable/confirmer signature scheme has anonymity in the case of two users. In fact, the proof of Theorem 4 relies only on the fact that both users share the same signature space \mathcal{S} . One can easily modify Definitions 3 and 4 and the statement of Theorem 4 so that they only reference \mathcal{S} and not the particular cryptosystem. Hence we obtain anonymity in the more general case where one user has an RSA scheme and the other a finite field scheme. In other words, we find ourselves in the strong position of preserving anonymity even when different users base their systems on quite different public key mechanisms.

Acknowledgements

The authors would like to thank Simon Blackburn, Markus Jakobsson, Arjen Lenstra, Kenny Paterson and the anonymous referees for helpful comments. The authors particularly thank an anonymous referee for pointing out a weakness in an earlier version of the paper.

References

- [1] M. Bellare and P. Rogaway, The exact security of digital signatures - how to sign with RSA and Rabin, in U. Maurer (ed.), EUROCRYPT '96, Springer LNCS 1070, (1996) 399–416. [88](#), [89](#)
- [2] M. Bellare, A. Boldyreva, A. Desai and D. Pointcheval, Key-privacy in public key encryption, in C. Boyd (ed.) ASIACRYPT 2001, Springer LNCS 2248 (2001) 566–582. [81](#), [88](#)

- [3] J. Camenisch and M. Michels, Confirmer signature schemes secure against adaptive adversaries, in B. Preneel (ed.), EUROCRYPT 2000, Springer LNCS 1870 (2000) 243–258. [81](#), [82](#), [84](#), [86](#), [93](#)
- [4] D. Chaum and H. van Antwerpen, Undeniable signatures, in G. Brassard (ed.), CRYPTO '89, Springer LNCS 435 (1990) 212–216. [80](#), [82](#)
- [5] D. Chaum, Zero-knowledge undeniable signatures, in I. B. Damgaard (ed.), CRYPTO '90, Springer LNCS 473 (1991) 458–464. [80](#), [82](#)
- [6] D. Chaum, E. van Heijst and B. Pfitzmann, Cryptographically strong undeniable signatures, unconditionally secure for the signer, in J. Feigenbaum (ed.), CRYPTO '91, Springer LNCS 576 (1992) 470–484. [81](#)
- [7] D. Chaum, Designated confirmer signatures, in A. de Santis (ed.), EUROCRYPT 94, Springer LNCS 950 (1995) 86–91. [80](#), [82](#)
- [8] Y. Desmedt, Securing traceability of ciphertexts: Towards a secure software escrow scheme, in Guillou et al. (eds.), EUROCRYPT '95, Springer LNCS 921 (1995) 147–157. [88](#)
- [9] S. D. Galbraith, W. Mao, and K. G. Paterson, RSA-based undeniable signatures for general moduli, in B. Preneel (ed.), Topics in Cryptology – CT-RSA 2002, Springer LNCS 2271 (2002) 200–217. [81](#), [82](#), [87](#), [89](#), [97](#)
- [10] R. Gennaro, H. Krawczyk and T. Rabin, RSA-based undeniable signatures, in W. Fumy (ed.), CRYPTO '97, Springer LNCS 1294 (1997) 132–149. [81](#), [82](#), [87](#), [97](#)
Also in *Journal of Cryptology* (2000)13:397–416.
- [11] M. Jakobsson, K. Sako and R. Impagliazzo, Designated verifier proofs and their applications, in U. Maurer (ed.) EUROCRYPT '96, Springer LNCS 1070 (1996) 143–154. [82](#), [90](#)
- [12] M. Jakobsson, Efficient oblivious proofs of correct exponentiation, in B. Preneel (ed.), Communications and multimedia security, Kluwer (1999) 71–84. [90](#)
- [13] M. Michels and M. Stadler, Generic constructions for secure and efficient confirmer signature schemes, in K. Nyberg (ed.) EUROCRYPT '98, Springer LNCS 1403 (1998) 406–421. [82](#)
- [14] T. Okamoto, Designated confirmer signatures and public key encryption are equivalent, in Y. G. Desmedt (ed.), CRYPTO '94, Springer LNCS 839 (1994) 61–74. [81](#)
- [15] T. Okamoto and D. Pointcheval, The Gap-problems: a new class of problems for the security of cryptographic schemes, in K. Kim (ed.) PKC '2001, Springer LNCS 1992 (2001) 104–118. [96](#)
- [16] K. Sakurai and S. Miyazaki, A bulletin-board based digital auction scheme with bidding down strategy - towards anonymous electronic bidding without anonymous channels nor trusted centers, In Proc. International Workshop on Cryptographic Techniques and E-Commerce, City University of Hong Kong Press, (1999) 180–187. [80](#)
- [17] K. Sakurai and S. Miyazaki, An anonymous electronic bidding protocol based on a new convertible group signature scheme, in E. Dawson et al. (eds.), ACISP 2000, Springer LNCS 1841 (2000) 385–399. [80](#)

A Proofs of Theorems 1 and 2

Here is the proof of Theorem 1.

Proof. Let \mathcal{A} be a distinguisher in the sense of [3]. We build a distinguisher \mathbf{D} using \mathcal{A} .

The input to \mathbf{D} is pk and we execute \mathcal{A} on this value. Queries made by \mathcal{A} are passed on by \mathbf{D} . Eventually \mathcal{A} outputs a pair (m_0, m_1) . At this point flip a coin to obtain a bit b' and take $m_{b'}$ as the output of \mathbf{D} . The challenge s received by \mathbf{D} is $\text{Sign}_{sk}(m_{b'})$ in the case $b = 0$ or is a random element of \mathcal{S} in the case $b = 1$. Pass s to \mathcal{A} as the challenge. Continue to answer queries by \mathcal{A} as before.

Finally, \mathcal{A} outputs its guess b'' . If $b'' = b'$ then output $b''' = 0$ by \mathbf{D} , and if $b'' \neq b'$ then output $b''' = 1$.

We have

$$\begin{aligned} \text{Adv}(\mathbf{D}) &= \Pr(b''' = b) - \frac{1}{2} \\ &= \Pr(b''' = 0 | b = 0) \Pr(b = 0) + \Pr(b''' = 1 | b = 1) \Pr(b = 1) - \frac{1}{2} \\ &= \Pr(b'' = b' | b = 0) \frac{1}{2} + \Pr(b'' \neq b' | b = 1) \frac{1}{2} - \frac{1}{2}. \end{aligned}$$

Now, when $b = 0$ then the game is indistinguishable from a real attack, and so $\Pr(b'' = b' | b = 0) = \frac{1}{2} + \text{Adv}(\mathcal{A})$.

On the other hand, when $b = 1$ then, with overwhelming probability, s is not a valid signature for either m_0 or m_1 . It follows that the hidden bit b' is independent of the view of \mathcal{A} . Hence, $\Pr(b'' \neq b' | b = 1) \approx \frac{1}{2}$ where the notation \approx denotes equality up to negligible terms⁴. Therefore

$$\text{Adv}(\mathbf{D}) \approx \left(\frac{1}{2} + \text{Adv}(\mathcal{A}) \right) \frac{1}{2} + \frac{1}{2} \frac{1}{2} - \frac{1}{2} = \frac{1}{2} \text{Adv}(\mathcal{A})$$

and the result follows. \square

We now give the proof of Theorem 2.

Proof. We first assume that the distribution on \mathcal{S} mentioned in Property A is exactly uniform. Let \mathbf{D} be an distinguisher as in Definition 1. We must transform \mathbf{D} into a distinguisher \mathcal{A} .

The input to \mathcal{A} is the key pk and so we execute \mathbf{D} on this key. Queries made by \mathbf{D} are passed on as queries made by \mathcal{A} . When \mathbf{D} produces a message m then select another message $m' \in \mathcal{M}$ uniformly at random and use m, m' (in either order) as the output of \mathcal{A} . The challenge received is s which is passed back

⁴ The negligible quantity is the probability, over random $pk \in \mathcal{K}, \{m_0, m_1\} \subseteq \mathcal{M}$ and $s \in \mathcal{S}$, that s is a valid signature on at least one of the m_i for pk . This probability depends on the scheme under consideration and so it is impossible to be more precise here. This probability must be negligible for any scheme which is secure against forgery of signatures.

to \mathbf{D} . Note that, s is a signature on either m or m' , and by our assumption, the distribution of $s \in \mathcal{S}$ in the latter case is uniform.

The simulation then proceeds in the obvious way and is indistinguishable from a real game. We clearly have $\text{Adv}(\mathbf{D}) = \text{Adv}(\mathcal{A})$.

Now, we consider the case when the distribution is merely indistinguishable from uniform. Suppose we are given an adversary \mathbf{D} as in Definition 1 and suppose that the advantage of \mathbf{D} in the game played in the first half of the proof is non-negligibly different from the advantage of \mathbf{D} in a real attack. Then it is straightforward to convert \mathbf{D} into an algorithm which distinguishes the two distributions on \mathcal{S} . By Property A we conclude that $\text{Adv}(\mathbf{D}) \approx \text{Adv}(\mathcal{A})$. \square

B Proof of Theorem 5

Proof. Suppose we have an adversary \mathbf{D} to the scheme. We will transform \mathbf{D} into a CDDH algorithm \mathcal{A} . Let (N, g, h, u, v) be the input CDDH challenge problem to \mathcal{A} . Note that we can use random-self-reducibility of CDDH tuples if required (let a, b and c be coprime to the order of \mathbb{Z}_N^* and let $g' = g^a, h' = h^a, u' = u^b g^c$ and $v' = v^b h^c$).

We execute \mathbf{D} on the key (N, g, h) . The distinguisher expects to perform hash queries, to obtain signatures on messages of its choice, and to run confirm and denial protocols. We now show how these will be simulated.

Hash query: A hash query could be with respect to any of the random oracles G_0, G_1, G_2 or F . We first analyse how to respond to a query $G_0(m, r)$ where m is a message and r is a random k_0 -bit string.

If the value $G_0(m, r)$ has not been queried before then perform the following simulation algorithm:

- Choose x, y at random between 1 and N^2 and compute a k -bit string α which reduces modulo N to $\pm(g^x u^y)^2$.
Store the values (x, y) as state information.
- Flip a coin.
- While tails do
 - Choose a random bitstring β of length k such that β corresponds to an element of \mathbb{Z}_N^* with Jacobi symbol -1 .
 - Define $F(\beta) = \alpha$ and store this as state information.
 - Set $\alpha = \beta$ and flip coin again.
- Parse the string α as $w \| r^* \| \gamma$, define $G_0(m, r)$ to be w , $G_1(w)$ to be $r \oplus r^*$, $G_2(w)$ to be γ and store all values as state information.

If there are any conflicts with existing definitions of the functions then the simulation algorithm halts (or retry with different (x, y)); this happens with negligible probability if k, k_0 and k_1 are sufficiently large.

The expected number of iterations of the while loop in the simulation algorithm is one.

A query on G_1 , G_2 or F can be answered using the state information when the inputs are the result of a previous query on $G_0(m, r)$, and can be answered with a random bitstring otherwise.

Sign query: To sign m we choose a random r and construct $H'(m, r)$ using the simulation algorithm. The value for $H'(m, r)$ is an element of the form $\pm(g^x u^y)^2$ where (x, y) is known. Compute a bitstring of length $2k$ which reduces modulo N to

$$s = h^x v^y \pmod{N}.$$

The signature is (r, s) .

For the challenge signature we choose the hidden bit b . If $b = 0$ we employ the above signing process. If $b = 1$ we choose s to be a random $2k$ -bit string.

Confirm/Deny: First test whether the signature (r, s) is in the equivalence class of the challenge signature. If so then return an error message and halt, otherwise proceed.

It is necessary to decide whether the signature should be considered valid or not within the simulation. To do this, consider the hash value $H'(m, r) = \pm(g^x u^y)^2$ and check if $(s/(h^x v^y))^2 \equiv 1 \pmod{N}$ (if not then the signature is declared to be invalid). Note that the only signatures which are considered valid are signatures in the same equivalence class as signatures formed from a sign query (the probability that \mathbf{D} constructs a valid signature any other way is negligible).

Once we have determined whether to respond positively or negatively then an appropriate proof can be simulated in the random oracle model.

The distinguisher \mathbf{D} will eventually output a guess b' for the hidden bit b . If $b' = b$ then \mathcal{A} outputs 1 (valid) as the answer to the CDDH problem and when $b' \neq b$ then \mathcal{A} outputs 0 (invalid).

We now analyse the simulation. First, we claim that the simulated functions G_0, G_1, G_2 and F are indistinguishable from random oracles: The set of squares in \mathbb{Z}_N^* is generated by g^2 , hence every value $\alpha = \pm a^2$ for $a \in \mathbb{Z}_N^*$ can be written in the form $\pm(g^x u^y)^2$. Furthermore, the distribution of numbers of this form is indistinguishable from uniform if $1 \leq x, y \leq N^2$ (actually, we could take $1 \leq x, y \leq M$ for some smaller bound $M > N$ and still obtain a good result).

When the input tuple (g, h, u, v) lies in $\mathcal{T}_{\text{CDDH}}$ then the Sign and Confirm/Deny oracles all behave perfectly. Hence, in this case, the simulation is identical to a real attack on the system, and so our CDDH algorithm \mathcal{A} satisfies

$$\Pr(\mathcal{A}(N, g, h, u, v) = 1 | (g, h, u, v) \in \mathcal{T}_{\text{CDDH}}) = \Pr(b' = b) = \frac{1}{2} + \text{Adv}(\mathbf{D}).$$

We now consider the case where the input tuple is a random element of \mathcal{T} . In this case signatures generated by oracle queries are (with high probability) invalid. The simulation is therefore not indistinguishable from a real attack, and we cannot predict the outcome of the distinguisher \mathbf{D} . However, as we now show, the hidden bit b is independent of the game in this case.

Let $\alpha \in \mathbb{Z}_N^*$ be chosen arbitrarily such that $\pm\alpha = \beta^2$ is a square and let $s \in \mathbb{Z}_N^*$. We will show that one can choose integers $1 \leq x, y < N$ such that

$\alpha = \pm(g^x u^y)^2$ and $s = h^x v^y$. First we consider the projection of all values into the cyclic subgroup of squares in \mathbb{Z}_N^* . By abuse of notation we may write $h = g^d, u = g^{c_1}, v = g^{c_2}, \beta = g^{c_3}$ and $s = g^{c_4}$. Solving the equations $\alpha = \pm(g^x u^y)^2$ and $s = h^x v^y$ is equivalent to solving

$$\begin{pmatrix} 1 & c_1 \\ d & c_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \equiv \begin{pmatrix} c_3 \\ c_4 \end{pmatrix} \pmod{p'q'}.$$

The matrix is invertible when $c_2 - dc_1$ is coprime to $p'q'$ and this happens with overwhelming probability⁵ when (g, h, u, v) is chosen uniformly at random from \mathcal{T} . Finally, we consider the squares: The value of α places no restriction on elements of order 2 in β , and h and v together generate \mathbb{Z}_N^* so we can solve the projection of the equation $s = h^x v^y$ in the subgroup of elements of order 2.

This proves that the hidden bit is independent of the simulation when the input (g, h, u, v) is a random 4-tuple. Hence the output of \mathcal{A} in this case differs negligibly from a coin-toss.

It follows that

$$\text{Adv}(\mathcal{A}) \approx \frac{1}{2} + \text{Adv}(\mathbf{D}) - \frac{1}{2} = \text{Adv}(\mathbf{D})$$

(where \approx means equality up to negligible factors). \square

C Other Security Properties

We now prove the security against forgery of the RSA-based undeniable signature scheme presented in Section 5.

C.1 Unforgeability

We show that our scheme resists forgery in the random oracle model if factoring is hard (this is much easier than the equivalent result for the Chaum and van Antwerpen scheme, which is due to Okamoto and Pointcheval [15]). Our proof handles both the case where the adversary is a designated confirmer (i.e., the adversary knows the decryption exponent e) and the general case.

Theorem 6. *In the random oracle model, if factoring integers which are products of safe primes is hard then the RSA-based scheme is secure.*

Proof. Let \mathcal{F} be an adversary which forges an RSA-based undeniable signature for our scheme in a CPA, CCA1 or CCA2 attack scenario. We show how to build a factoring algorithm using \mathcal{F} .

Let N be the challenge integer and choose a random odd integer e (since N is a product of safe primes then e is coprime to $\varphi(N)$ with overwhelming probability).⁶ Choose a random $t \in \mathbb{Z}_N^*$ and define $g = t^e \pmod{N}$ and $h = t$. The

⁵ This is where the difficulties arise if N is not a product of safe primes.

⁶ To generalise this result to moduli which are not products of safe primes it suffices to choose e to be a medium-sized prime.

input to \mathcal{F} is the triple (N, g, h) and if the forger is the confirmor then they are also given e .

The forger \mathcal{F} will make various queries to the simulation and this is how we respond to them:

Hash query: These are answered similarly to the proof of Theorem 5. The basic step is to define $H'(m, r)$ as follows: Choose a random element $a \in \mathbb{Z}_N^*$ and set $H'(m, r) \equiv \pm a^{2e} \pmod{N}$.

Sign query: If \mathcal{F} asks for an undeniable signature on m then construct $H'(m, r) = \pm a^{2e}$. The response is the pair (r, a) . Since a was chosen randomly then it has random Jacobi symbol.

Confirm/Deny: We determine the validity of a given signature using the verification exponent e and then simulate the proof accordingly in the random oracle model.

One can easily show that this simulation is indistinguishable from a real game. Eventually the forger outputs a triple (m, r, s) such that

$$s^{2e} \equiv \pm H'(m, r) \pmod{N}.$$

Since N is a Blum integer and s^{2e} is a square then we have a minus in the above formula if and only if $H'(m, r)$ was defined to be $-a^{2e}$. Hence we have $s^{2e} \equiv a^{2e} \pmod{N}$ and so

$$(s^e/a^e)^2 \equiv 1 \pmod{N}.$$

Since a was chosen randomly then, with probability $1/2$, we have a nontrivial root of unity which gives a factorisation of N . \square

C.2 Convertibility

In [9, 10] it is shown how to obtain an undeniable/confirming signature scheme based on RSA which allows conversion to ordinary RSA signatures. The same approach applies to our scheme.

The basic idea is to have values e, d and c such that $edc \equiv 1 \pmod{\varphi(N)}$. The value e is now public, while d is known only to the signer and c is used for confirming/converting. The scheme of Section 5 is now verified using the relation $s^{2ce} \stackrel{?}{\equiv} \pm H'(m, r) \pmod{N}$. Signatures may be selectively converted to public key signatures by raising to the power c (this should be accompanied by a zero-knowledge proof of correctness). All signatures may be converted by publishing the value c .

A Secure Signature Scheme from Bilinear Maps

Dan Boneh^{1,*}, Ilya Mironov^{1,**}, and Victor Shoup²

¹ Computer Science Department, Stanford University
`{dabo,mironov}@cs.stanford.edu`

² Courant Institute, New York University
`shoup@cs.nyu.edu`

Abstract. We present a new class of signature schemes based on properties of certain bilinear algebraic maps. These signatures are secure against existential forgery under a chosen message attack in the standard model (without using the random oracle model). Security is based on the computational Diffie-Hellman problem. The concrete schemes that we get are the most efficient provable discrete-log type signature schemes to date.

1 Introduction

Provably secure signature schemes can be constructed from the most basic cryptographic primitive, one-way functions [NY89, Rom90]. As is often the case with cryptographic schemes designed from elementary blocks, this signature scheme is somewhat impractical. Over the years several signature schemes were proposed based on stronger complexity assumptions. The most efficient schemes provably secure in the standard model are based on the Strong RSA assumption [GHR99, CS99].

Surprisingly, no scheme based on any discrete logarithm problem comes close to the efficiency of the RSA-based schemes. We give a partial solution to this open problem using bilinear maps. A bilinear map is a function $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ that is consistent with group operations in both of its arguments, as described in the next section. Our construction gives an existentially unforgeable signature whenever the Computational Diffie-Hellman (CDH) assumption holds in $\mathbb{G}_0 \times \mathbb{G}_1$, that is, no efficient algorithm can compute $g^\alpha \in \mathbb{G}_1$ given the three values $h, h^\alpha \in \mathbb{G}_0$, and $g \in \mathbb{G}_1$. Precise definitions are given in the next section.

Our signature scheme is based on a signature authentication tree with a large branching factor. At a high level, our construction bears some resemblance to the signature schemes of Dwork-Naor [DN94] and Cramer-Damgård [CD96]. Both these schemes are based on the hardness of factoring whereas our scheme is based on CDH.

We obtain a concrete signature scheme by instantiating the bilinear map with the modified Weil or Tate pairings. This is the only known provably secure signature scheme based on the CDH assumption that is more efficient than

* Supported by NSF Career Award, DARPA, and Packard Foundation.

** Supported by Microsoft Fellowship.

the most general constructions of [CD95]. It is interesting to note that recently, Lysyanskaya [Lys02] constructed a verifiable unpredictable function (VUF) secure under the Generalized Diffie-Hellman assumption in the standard model. Such functions (defined in [MRV99]) provide a special type of secure signatures called unique signatures. This construction gives an excellent VUF, but as a signature scheme it compares poorly with the construction of [CD95]. We review other known signature schemes in Section 4.

Bilinear maps such as the Weil or Tate pairing have recently been used to construct a number of new cryptosystems, including three-party key exchange [Jou00], identity based encryption [BF01], short signatures [BLS01], credential systems [Ver01], hierarchical identity based encryption [HL02, GS02], and others. In this paper we show how bilinear maps can be used to construct efficient signature schemes secure in the standard model.

Efficient discrete log based signature schemes are known to exist in the random oracle model [PS96, BLS01]. Security in the random oracle model does not imply security in the real world. In this paper we only study signature schemes secure in the standard complexity model.

2 Mappings with Algebraic Properties

We consider binary maps between groups that are consistent with the group structure of their arguments. Such binary maps are called bilinear. Their formal definition follow.

Definition 1 (Bilinear map). *A function $e: \mathbb{G}_0 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is bilinear if for any four elements $g_1, g_2 \in \mathbb{G}_0$, $H_1, H_2 \in \mathbb{G}_1$ the following holds:*

$$e(g_1 \circ g_2, H_1) = e(g_1, H_1) \circ e(g_2, H_1) \quad \text{and} \quad e(g_1, H_1 \circ H_2) = e(g_1, H_1) \circ e(g_1, H_2).$$

In this paper we intentionally limit our scope to finite cyclic groups, which allows us to give more efficient constructions.

Throughout the paper we use the following notation. Small Roman letters f, g, h, \dots from the lower part of the alphabet denote elements of the group \mathbb{G}_0 ; capital Roman letters F, G, H, \dots stand for elements of \mathbb{G}_1 ; elements of \mathbb{G}_2 are denoted by letters from the end of the alphabet x, y, z .

Our constructions are based on the Computational Diffie Problem (CDH) defined below. However to simplify the exposition we define the notion of a secure bilinear map. We then show that this notion is equivalent to CDH. Informally, we say that a bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is secure if, given $g \in \mathbb{G}_0$ and $G, H \in \mathbb{G}_1$, it is hard to find $h \in \mathbb{G}_0$ such that $e(h, H) = e(g, G)$. More precisely, we define secure bilinear maps as follows.

Definition 2 (Secure bilinear map). *A bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is (t, ε) -secure if for all t -time adversaries \mathcal{A}*

$$\text{AdvBLM}_{\mathcal{A}} = \Pr \left[e \left(\mathcal{A}(g, G, H), H \right) = e(g, G) \mid g \xleftarrow{R} \mathbb{G}_0; G, H \xleftarrow{R} \mathbb{G}_1 \right] < \varepsilon.$$

The probability is taken over the coin tosses of the algorithm \mathcal{A} and random choice of g, G, H .

We give two simple examples of bilinear maps that are believed to be secure.

- $e(\cdot, \cdot): \mathbb{G}_0 \times \mathbb{G}_1 \mapsto \mathbb{F}_{p^r}^*$ is the Weil or Tate pairings on an elliptic curve E/\mathbb{F}_p and $\mathbb{G}_0, \mathbb{G}_1$ are distinct subgroups of $E[q]$ for some prime q (recall that $E[q]$ is the subgroup containing all points of order dividing q on some elliptic curve E/\mathbb{F}_p). For certain curves E/\mathbb{F}_p one can slightly modify the Weil pairing (as in [BF01]) so that we may take $\mathbb{G}_0 = \mathbb{G}_1$. At any rate, the security of the bilinear map $e(\cdot, \cdot)$ is equivalent to the Computational Diffie-Hellman assumption (CDH) in $\mathbb{G}_0 \times \mathbb{G}_1$. Informally, the CDH assumption in $\mathbb{G}_0 \times \mathbb{G}_1$ means that:

It is infeasible to find G^a given random group elements $h, h^a \in \mathbb{G}_0$, and $G \in \mathbb{G}_1$. When $\mathbb{G}_0 = \mathbb{G}_1$ this is the standard CDH assumption in \mathbb{G}_0 .

- Another bilinear map believed to be secure is $r(\cdot, \cdot): \mathbb{Z}_N^* \times \mathbb{Z}_{\varphi(N)}^+ \mapsto \mathbb{Z}_N^*$ defined as $r(g, H) = g^H$, where N is a product of two primes. This map is secure under the Strong RSA assumption [BP97]. Briefly, the Strong RSA assumption says that the following problem is difficult to solve:

For a random $x \in \mathbb{Z}_N^*$ find $r > 1$ and $y \in \mathbb{Z}_N^*$ such that $y^r = x$.

We give a short argument why the security of the map $r(\cdot, \cdot)$ is reducible to the Strong RSA assumption. Suppose the map $r(\cdot, \cdot)$ is insecure. Then, given (G, H, g) it is feasible to find an $h \in \mathbb{G}_0$ such that $r(g, G) = r(h, H)$, i.e. $g^G = h^H$. This solution yields (through application of the Extended Euclidean algorithm) a $z \in \mathbb{Z}_N^*$ satisfying $z^a = g$, where $a = H/\gcd(G, H)$.

For random $G, H \xleftarrow{R} \mathbb{Z}_{\varphi(N)}^+$ the probability that $a = 1$ is negligible. Therefore breaking the security of the bilinear map r amounts to breaking the Strong RSA assumption. It is not known whether the converse is true.

Next, we show that for finite order groups a bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is secure if and only if the CDH assumption holds in $\mathbb{G}_0 \times \mathbb{G}_1$. We first precisely define the CDH assumption.

Definition 3 (Computational Diffie-Hellman problem). *The Computational Diffie-Hellman problem is (t, ε) -hard if for all t -time adversaries \mathcal{A} we have*

$$\text{AdvCDH}_{\mathcal{A}} = \Pr \left[\mathcal{A}(g, H, H^a) = g^a \mid g \xleftarrow{R} \mathbb{G}_0; H \xleftarrow{R} \mathbb{G}_1; a \xleftarrow{R} \mathbb{Z}_{|\mathbb{G}_1|} \right] < \varepsilon.$$

Claim. Suppose that $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2$ are cyclic groups of prime order p . Suppose the map $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is non-degenerate in the following sense: $e(h, H) \neq 1$ for some $h \in \mathbb{G}_0$ and $H \in \mathbb{G}_1$. Then the map e is (t, ε) -secure if and only if the CDH problem is (t, ε) -hard.

Proof. First, suppose CDH can be solved in time t with probability at least ε . We give an algorithm to show that the map is not (t, ε) -secure. Let $g \in \mathbb{G}_0$ and $G, H \in \mathbb{G}_1$ where both $G, H \neq 1$. We wish to find $h \in \mathbb{G}_0$ such that

$e(g, G) = e(h, H)$. Since \mathbb{G}_1 is cyclic of prime order p there exists an $a \in \mathbb{Z}_p$ such that $G = H^a$. Let $h = g^a$. Then h satisfies

$$e(h, H) = e(g^a, H) = e(g, H^a) = e(g, G).$$

Therefore, $h = g^a$, which is the solution to the CDH problem (g, H, G) , is the required h . Hence, if the map is (t, ε) -secure then CDH is (t, ε) -hard.

Conversely, suppose there is a t -time algorithm that given random (g, G, H) outputs $h \in \mathbb{G}_0$ such that $e(g, G) = e(h, H)$ with probability at least ε . We show how to solve CDH. Let (g, H, G) be a random instance of the CDH problem, where $H \neq 1$. Write $G = H^a$ for some $a \in \mathbb{Z}_p$. Let h be such that $e(g, G) = e(h, H)$. Then

$$e(h, H) = e(g, G) = e(g, H^a) = e(g^a, H)$$

and hence $e(h/g^a, H) = 1$. Since $H \neq 1$ it follows that $h = g^a$, since otherwise the map e would be degenerate. Hence, if CDH is (t, ε) -hard then the map is (t, ε) -secure. \square

3 Security for Signature Schemes

We recall the standard definition of secure signature schemes stated in terms of exact security, in the spirit of [BR94]. This notion of existential unforgeability under adaptive chosen-message attack is due to [GMR88].

A signature scheme is a triple of probabilistic algorithms: a key generation algorithm KeyGen , signer $\text{Sign}(SK, Msg)$, and verifier $\text{Verify}(Msg, Sig, PK)$. By convention, Verify outputs 1 if it accepts the signature and 0 otherwise. We use the oracle notation, where $A^{B(\cdot)}(\cdot)$ means A supplied with oracle access to B .

Definition 4. A signature scheme is (t, ε, n) -existentially unforgeable under adaptive chosen-message attack, if for all pairs of algorithms $\mathcal{F}_1, \mathcal{F}_2$ running in time at most t

$$\begin{aligned} \text{AdvSig}_{\mathcal{F}_1, \mathcal{F}_2} &= \Pr[\text{Verify}(\mathcal{F}_2(T), PK) = 1 \mid \\ &\quad (SK, PK) \leftarrow \text{KeyGen}(1^k); T \leftarrow \mathcal{F}_1^{\text{Sign}(SK, \cdot)}(1^k)] < \varepsilon. \end{aligned}$$

\mathcal{F}_1 requests no more than n signatures from Sign and the message output by \mathcal{F}_2 is different from the messages signed by Sign . The probability is taken over the coin tosses of KeyGen , Sign , \mathcal{F}_1 and \mathcal{F}_2 . Here $\mathcal{F}_2(T)$ outputs a message/signature pair.

4 Previous Work

The seminal paper [GMR84] formulated a strong notion of security for signature schemes: existential unforgeability under adaptive chosen-message attacks. Since then there have been many proposals for signature schemes meeting this notion

Table 1. Summary of provably secure signature schemes. k is the security parameter, ℓ is the depth of the authentication tree, n is the branching factor of the tree, and m is the message length. The O -notation refers to asymptotics as a function of the security parameter k

Reference	Signature length	Public key length	Max number of signatures	Security assumption
[GMR84]	$O(k\ell)$	$O(k)$	2^ℓ	claw-free trapdoor permutations
[NY89]	$O(k^2\ell)$	$O(k)$	2^ℓ	UOWHF
[CD95]	$O(k\ell)$	$O(k)$	2^ℓ	one-way homomorphism
[DN94]	$O(k\ell)$	$O(kn)$	n^ℓ	RSA assumption
[CD96]	$O(k\ell)$	$O(k+n)$	n^ℓ	RSA assumption
[GHR99],[CS99]	$O(k)$	$O(k)$	∞	Strong RSA assumption
[Lys02]	$O(km)$	$O(km)$	2^m	Generalized Diffie-Hellman
this paper	$O(k\ell)$	$O(kn)$	n^ℓ	Computational Diffie-Hellman (secure bilinear maps)

of security based on different assumptions and of varying efficiency. Some of these results are summarized in Table 1. With an exception of GMR, all schemes have running time of the signing and verification algorithms proportional to the signature length. This information is omitted from the table.

A signature scheme may be based on the most general cryptographic assumption, namely that one-way functions exist [NY89, Rom90]. The proof proceeds via constructing an authentication tree and results in a scheme in which the signature length is proportional to the binary logarithm of the total number of messages signed with the same public key. More efficient (in terms of the signature length) signature schemes can be based on the Strong RSA assumption or its variants [CS99, GHR99]. Important steps in constructing these schemes were the Dwork-Naor scheme [DN94] later improved by [CD96]. Both schemes use trees with a large branching factor, which are therefore very shallow.

The Dwork-Naor trick is crucial for understanding this paper. In a nutshell, the trick allows to increase the tree's branching factor without blowing up the signature size. An authentication-tree scheme produces signatures that represent paths connecting messages and the root of the tree. Messages are usually placed in the very bottom level of the tree, though [CD95] puts messages in the leaves hanging from the inner nodes. The authentication mechanism works inductively: the root authenticates its children, they authenticate their children, and so on, down to the message authenticated by its parent. If the authentication mechanism allows attaching children to a node only when some secret information is known, then the adversary cannot forge signatures without knowing that secret.

[GMR84] and [CD95] take similar approaches in designing the authentication mechanism (hence the identical asymptotic of their signature length). They concatenate bit representations of a node's children and compute a value that authenticates this string in respect to the node. To verify a node's authenticity

we must know all siblings of the node. If the tree is binary, the signature contains twice as many nodes as the the depth of the tree. Indeed, each node must be accompanied by its sibling.

Since the signature length is proportional to the depth of the tree, one may wonder whether increasing the tree's branching factor is a good idea. The following simple computation shows why this is counterproductive.

Suppose one wants to sign N messages. If the authentication tree is binary, its depth must be at least $\log_2 N$. The signature length is roughly $2k \log_2 N$, where k is the security parameter that accounts for the nodes' size. When the branching factor of the tree is increased from 2 to d , the depth of the tree goes down to $\log_d N$. The signatures, however, must include all siblings of the nodes on the authentication path (ancestors of the message-leaf). Thus the signature size becomes $dk \log_d N$, which is actually *more* than in the binary case.

The improvement achieved by [DN94] is due to the way the sibling nodes are authenticated. Using a stronger complexity assumption than in the GMR scheme, authenticity of a node can be verified given its parent and its authentication value, whose size is independent of the branching factor. Each node has the authentication value different from those of its siblings and their authenticity can be verified independently. It allows to increase the number of a node's children and decrease the depth of the tree without inflating the signature length. The public key size, being the branching factor *times* the security parameter and, because of that, the main drawback of [DN94], was reduced in the Cramer-Damgård scheme [CD96]. Finally, two independent methods proposed in [CS99] and [GHR99] make the tree flat by letting the signer (but not the adversary) add branches on the fly.

We do not distinguish between statefull and stateless schemes. All schemes can be made memoryless at the price of doubling the tree depth [Gol86]. To do so [Gol86] suggested using a pseudo-random function (PRF) to generate all secret values in internal nodes of the authentication tree. The key for the PRF is kept secret at the signer. Furthermore, instead of sequentially stepping through the leaves of the tree (one leaf per signature) we pick a random leaf for every signature. To make sure the same leaf is not chosen at random for two different signatures we need to square the number of leaves and hence double the depth of the tree.

In this paper we implement the Dwork-Naor method using a secure bilinear map. Improving the scheme further in the direction of [CD96, CS99, GHR99] is left as an open problem.

5 The New Signature Scheme

We present a signature scheme based on a secure bilinear map. An instantiation of this construction yields the most efficient provably secure scheme based on the Computational Diffie-Hellman assumption known to date.

The signature scheme is designed as follows.

– **Setup of the scheme.** Groups \mathbb{G}_0 and \mathbb{G}_1 have prime order p . Bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is secure. $\mathcal{H}_k: \mathcal{M} \mapsto \{0,1\}^s$ is a family of collision-resistant hash functions, where \mathcal{M} is the message space (the key for \mathcal{H}_k is fixed at random during key generation and made public). The signature scheme allows signing of ℓ^n messages, where ℓ and n are arbitrary positive integers.

– **Key generation.** The public and private keys are created as follows:

Step 1. Pick $\alpha_1, \dots, \alpha_n \xleftarrow{R} \mathbb{Z}_p^*$ and $H \xleftarrow{R} \mathbb{G}_1$. Choose a random key k for the collision resistant hash function \mathcal{H}_k . Compute $H_1 \leftarrow H^{1/\alpha_1}, \dots, H_n \leftarrow H^{1/\alpha_n} \in \mathbb{G}_1$.

Step 2. Pick $g \xleftarrow{R} \mathbb{G}_0$. Compute $y \leftarrow e(g, H)$.

Step 3. Pick $\beta_0 \xleftarrow{R} \mathbb{Z}_p$. Compute $x_0 \leftarrow e(g, H)^{\beta_0}$.

Step 4. The **public key** is k, H, H_1, \dots, H_n, y and x_0 . The **private key** is $\alpha_1, \dots, \alpha_n, \beta_0$, and g .

– **Signing algorithm.** Each node in the tree is authenticated with respect to its parent; messages to be signed are authenticated with respect to the leaves, which are selected in sequential order and never reused. To sign the i^{th} message $M \in \mathcal{M}$ the signer generates the i^{th} leaf of the authentication tree together with a path from the leaf to the root. We denote the leaf by $x_\ell \in \mathbb{G}_1$ and denote the path from the leaf to the root by $(x_\ell, i_\ell, x_{\ell-1}, i_{\ell-1}, \dots, i_1, x_0)$, where x_j is the i_j^{th} child of x_{j-1} (here $i_j \in \{1, \dots, n\}$). The leaf and the nodes along the path and their authentication values are computed as follows:

Step 1. All nodes of the tree, including the leaf x_ℓ , that have not been visited before are computed as $x_j \leftarrow e(g, H)^{\beta_j}$ for some random $\beta_j \xleftarrow{R} \mathbb{Z}_p$. The secret β_j is stored for as long as node x_j is an ancestor of the current signing leaf (β_ℓ is discarded immediately after use). Note that when using stateless signatures [Gol86] the β_j are derived using a PRF and hence there is no need to remember their values.

Step 2. The authentication value of x_j , the i_j^{th} child of x_{j-1} , is $f_j \leftarrow g^{\alpha_{i_j}(\beta_{j-1} + \mathcal{H}_k(x_j))}$.

Step 3. The authentication value of $\mathcal{H}_k(M)$, the child of x_ℓ , is $f \leftarrow g^{\beta_\ell + \mathcal{H}_k(M)}$.

Step 4. The **signature** on M is $(f, f_\ell, i_\ell, \dots, f_1, i_1)$.

– **Verification algorithm.** To verify a signature $(\hat{f}, \hat{f}_\ell, \hat{i}_\ell, \dots, \hat{f}_1, \hat{i}_1)$ on a message M we reconstruct the nodes $\hat{x}_\ell, \dots, \hat{x}_0$ in a bottom-up order (from leaf \hat{x}_ℓ to root \hat{x}_0). The signature is accepted if and only if \hat{x}_0 matches the root of the tree. More precisely, to verify the signature the verifier performs the following steps:

Step 1. Compute $\hat{x}_\ell \leftarrow e(\hat{f}, H) \cdot y^{-\mathcal{H}_k(M)}$.

Step 2. For $j = \ell \dots 1$ compute $\hat{x}_{j-1} \leftarrow e(\hat{f}_j, H_{i_j}) \cdot y^{-\mathcal{H}_k(\hat{x}_j)}$.

Step 3. The signature is accepted if $\hat{x}_0 = x_0$.

A signature output by the signing algorithm passes the verification test. Indeed, step 1 of the verification algorithm results in

$$e(f, H) \cdot y^{-\mathcal{H}_k(M)} = e(g^{\beta_\ell + \mathcal{H}_k(M)}, H) \cdot e(g, H)^{-\mathcal{H}_k(M)} = e(g^{\beta_\ell}, H) = x_\ell.$$

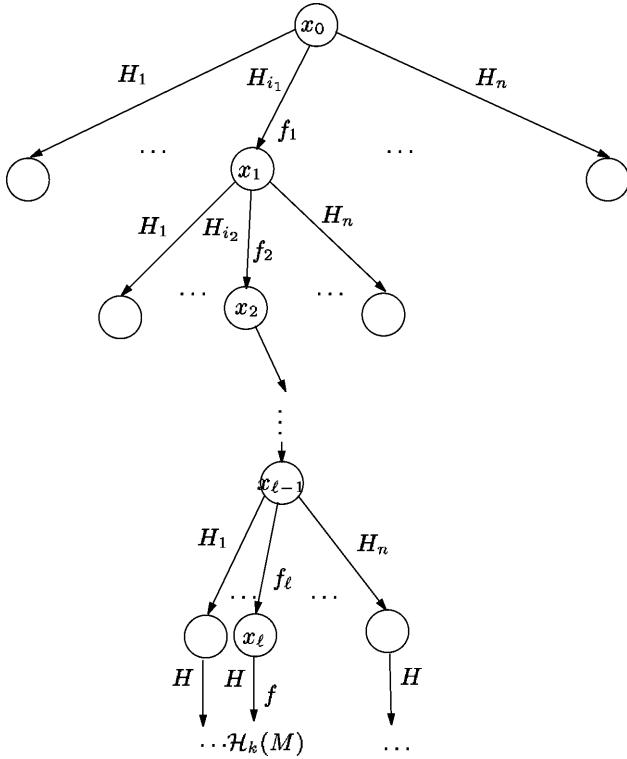


Fig. 1. Authentication tree. The signature on M is $(f, f_\ell, i_\ell, \dots, f_1, i_1)$

For any $j \in 1 \dots \ell$ the result of computation in step 2 of the verification algorithm is

$$\begin{aligned} e(f_j, H_{i_j}) \cdot y^{-\mathcal{H}_k(x_j)} &= e(g^{\alpha_{i_j}(\beta_{j-1} + \mathcal{H}_k(x_j))}, H^{1/\alpha_{i_j}}) \cdot e(g, H)^{-\mathcal{H}_k(x_j)} = \\ &e(g^{\beta_{j-1} + \mathcal{H}_k(x_j)}, H) \cdot e(g^{-\mathcal{H}_k(x_j)}, H) = e(g^{\beta_{j-1}}, H) = x_{j-1}. \end{aligned}$$

Signature length. Suppose the user needs to generate a billion signatures. Then taking $n = 20$ and $\ell = 4$ is sufficient ($4^{20} > 10^{12}$). The public key will contain 20 elements in \mathbb{G}_1 and two elements in \mathbb{G}_2 . The signature contains five elements in \mathbb{G}_0 . To get stateless signatures we would need to double the depth so that each signature will contain nine elements of \mathbb{G}_0 .

Security. We show that the signature scheme is secure against the most general attack. To formalize it as a claim we need to recall the definition of collision-resistance and assume the existence of a collision-finding algorithm for e .

Definition 5 (Collision-resistant function). *We call function $\mathcal{H}: \mathcal{K} \times \mathcal{M} \mapsto \{0, 1\}^s$ a family of (t, ε) -collision-resistant functions, if for any t -time algorithm*

\mathcal{A} its advantage in finding a collision

$$\text{AdvCR}_{\mathcal{A}} = \Pr[\mathcal{H}_k(M_1) = \mathcal{H}_k(M_2), M_1 \neq M_2 \mid k \xleftarrow{R} \mathcal{K}; (M_1, M_2) \leftarrow \mathcal{A}(k)] < \varepsilon.$$

The probability is taken over \mathcal{A} 's coin tosses.

Definition 6 (Collision-finding algorithm). We say that an algorithm is collision-finding for bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ if it outputs a solution $g', g'' \in \mathbb{G}_0$, where $g', g'' \neq 1$, to the equation

$$e(g', G') = e(g'', G''),$$

for given $G', G'' \in \mathbb{G}_1$ whenever such a solution exists.

For the bilinear maps defined on elliptic curves, namely the Weil and Tate pairings, it is easy to build collision finding algorithms. This yields a concrete discrete-log signature scheme as described later in this section.

Theorem 1. The signature scheme is $(t, \varepsilon/(n+1), m)$ -secure against existential forgery against adaptive chosen-message attack under the following assumptions:

- e is a (t, ε) -secure bilinear map;
- \mathcal{H} is (t, ε) -collision-resistant function;
- there is an efficient collision-finding algorithm for e .

Proof. The proof is by contradiction. We show that existence of an efficient forger implies that we can either find a collision in \mathcal{H} or solve

$$e(g^*, G_1) = e(g_2, G_3)$$

for g^* given $G_1, G_3 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_0$.

To this end we set up the public key to help the simulator in answering the adversary's signing queries. In the same time a forgery would let us respond to the challenge or find a collision of the hash function with a non-negligible probability.

First, we set $y \leftarrow e(g_2, G_3)$. Second, we pick random $i \in \{0, \dots, n\}$. Consider two cases.

Case 1: $i = 0$. We assign $H \leftarrow G_1$, pick $\gamma_j \xleftarrow{R} \mathbb{Z}_p$ for all $j \in 1 \dots n$ and assign $H_j \leftarrow G_3^{1/\gamma_i}$. All internal nodes of the authentication tree, including the root x_0 but excluding the leaves, are computed as random powers of $e(g_2, G_3)$. Suppose x' is the j^{th} child of x'' , where $x'' = e(g_2, G_3)^\gamma$. The authentication value for x' is computed as $f' \leftarrow g_2^{\gamma_j(\gamma + \mathcal{H}(x'))}$. It correctly authenticates x' as the j^{th} child of x'' , since

$$e(f', H_j) \cdot y^{-\mathcal{H}(x')} = e(g_2^{\gamma_j(\gamma + \mathcal{H}(x'))}, G_3^{1/\gamma_j}) \cdot e(g_2, G_3)^{-\mathcal{H}(x')} = e(g_2, G_3)^\gamma = x''.$$

When the adversary requests a signature on a message M , the path $(i_\ell, x_{\ell-1}, i_{\ell-1}, \dots, i_1, x_0)$ is generated and the authentication values for all internal

nodes along the path are included in the signature. The leaf x_ℓ is computed as $x_\ell \leftarrow e(g_2, G_1^\gamma \cdot G_3^{-\mathcal{H}(M)})$ for a randomly chosen $\gamma \xleftarrow{R} \mathbb{Z}_p$. The authentication value for $\mathcal{H}(M)$ is set to be $f \leftarrow g_2^\gamma$. The leaf x_ℓ is authenticated as the i_ℓ^{th} child of $x_{\ell-1}$ as above. This results in a valid signature $(f, f_\ell, i_\ell, \dots, f_1, i_1)$.

Case 2: $i \in \{1, \dots, n\}$. Assign $H_i \leftarrow G_1$. We randomly choose $\gamma, \gamma_1, \dots, \gamma_{i-1}, \gamma_{i+1}, \dots, \gamma_n \xleftarrow{R} \mathbb{Z}_p$, assign $H \leftarrow G_3^{1/\gamma}$ and $H_j \leftarrow G_3^{1/\gamma_j}$, for all $j \neq i$. We apply the collision-finding algorithm that returns d_1 and d_2 satisfying

$$e(d_1, G_1) = e(d_2, G_3).$$

The authentication tree is constructed from the bottom up. Suppose we are given a set of siblings z_1, \dots, z_n . The challenge is to define their parent node z such that we may find authentication values for all of them. Let $z \leftarrow e(d_2, G_3^\delta) y^{-\mathcal{H}(z_i)}$ for a randomly chosen $\delta \xleftarrow{R} \mathbb{Z}_q$. For all $j \in 1 \dots n$, such that $j \neq i$, the authentication value of z_j can be computed as $f_j \leftarrow (g_2^{\mathcal{H}(z_j)-\mathcal{H}(z_i)} \cdot d_1^\delta)^{\gamma_j}$. Indeed, for all such j

$$\begin{aligned} e(f_j, H_j) \cdot y^{-\mathcal{H}(z_j)} &= e((g_2^{\mathcal{H}(z_j)-\mathcal{H}(z_i)} \cdot d_1^\delta)^{\gamma_j}, G_3^{1/\gamma_j}) \cdot e(g_2, G_3)^{-\mathcal{H}(z_j)} = \\ &e(g_2^{\mathcal{H}(z_j)-\mathcal{H}(z_i)} \cdot d_1^\delta, G_3) \cdot e(g_2^{-\mathcal{H}(z_j)}, G_3) = e(g_2^{-\mathcal{H}(z_j)} \cdot d_1^\delta, G_3) = \\ &e(d_2, G_3^\delta) \cdot e(g_2, G_3)^{-\mathcal{H}(z_i)} = e(d_2, G_3^\delta) y^{-\mathcal{H}(z_i)} = z_j. \end{aligned}$$

Node z_i is authenticated with $f_i \leftarrow d_1^\delta$. Indeed,

$$e(f_i, H_i) \cdot y^{-\mathcal{H}(z_i)} = e(d_1^\delta, G_1) \cdot y^{-\mathcal{H}(z_i)} = e(d_2, G_3) \cdot y^{-\mathcal{H}(z_i)} = z.$$

It follows that every internal node may be computed given its j^{th} child. In particular, the root of the tree, x_0 , is determined by values on the path $(x_\ell, j, x_{\ell-1}, \dots, x_1, j)$, which can be efficiently computed by the randomized algorithm given above.

When a signature is requested by the adversary, a new leaf is generated as $x_\ell \leftarrow e(g_2, H^\delta)$, where $\delta \xleftarrow{R} \mathbb{Z}_p$. Then the path to the root is computed, which would involve generating new nodes from their j^{th} children. The authentication value for $\mathcal{H}(M)$ is given by $f \leftarrow g_2^{\delta+\gamma\mathcal{H}(M)}$.

We have shown that the simulator may effectively replace the signing oracle and answer the adversary's queries. The simulated answers have the same distribution as signatures output by the real signer.

Solving the challenge. Let us consider an algorithm that interacts with the signing oracle and then forges a signature $(f, f_\ell, i_\ell, \dots, f_1, i_1)$ on a message M . Without loss of generality we may assume that the adversary makes ℓ^n queries. Denote the full authentication tree constructed by the simulator by T . The signature on M has the form of an authenticated path up the tree from a leaf to the root x_0 . Let $(x_\ell, i_\ell, x_{\ell-1}, i_{\ell-1}, \dots, i_1, x_0)$ be the path reconstructed from the signature. Define x_j as the lowest node of the path that also appears in T .

We distinguish between two types of forgeries:

Type I: $j = \ell$. Denote the message that was previously authenticated using x_ℓ by M' and let f' be the authentication value of $\mathcal{H}(M')$. It follows that

$$\begin{aligned} e(f, H) \cdot y^{-\mathcal{H}(M)} &= x_\ell, \\ e(f', H) \cdot y^{-\mathcal{H}(M')} &= x_\ell, \end{aligned}$$

from which we have

$$y^{\mathcal{H}(M)-\mathcal{H}(M')} = e(f/f', H). \quad (1)$$

Type II: $j < \ell$. Let the x'_{j+1} be the i_j^{th} child of x_j in T and f'_j be its authentication value. Similarly, there are two equations

$$\begin{aligned} e(f_j, H_{i_j}) \cdot y^{-\mathcal{H}(x_{j+1})} &= x_j, \\ e(f'_j, H_{i_j}) \cdot y^{-\mathcal{H}(x'_{j+1})} &= x_j, \end{aligned}$$

that imply

$$y^{\mathcal{H}(x_{j+1})-\mathcal{H}(x'_{j+1})} = e(f_j/f'_j, H_{i_j}). \quad (2)$$

Consider two possibilities. If $\mathcal{H}(M) = \mathcal{H}(M')$ in type I or $\mathcal{H}(x_{j+1}) = \mathcal{H}(x'_{j+1})$ in type II forgery, then we find a collision in the hash function. Otherwise, we solved equation (1) or (2) of the type

$$y^d = e(\hat{f}, \hat{H}),$$

for d and \hat{f} , where $d \neq 0$ and \hat{H} is one of H, H_1, \dots, H_n .

Recall that $y = e(g_2, G_3)$. Since the simulation of the adversary's view is perfect, the probability that $\hat{H} = G_1$ is $\frac{1}{n+1}$. Thus $e(\hat{f}^{1/d}, G_1) = e(g_2, G_3)$ and $\hat{f}^{1/d} = g^*$, i.e. a solution to the challenge.

Therefore an efficient algorithm for forging a signature can be used to either find a collision in the hash function or disprove security of e . This completes the proof of the theorem. \square

6 Concrete Signature Scheme

To obtain a concrete secure signature scheme we instantiate the bilinear map $e: \mathbb{G}_0 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with a map for which there is an efficient collision finding algorithm. Let E/\mathbb{F}_p be an elliptic curve. We denote by $E(\mathbb{F}_{p^r})$ the group of points of E over \mathbb{F}_{p^r} . Let q be a prime divisor of $|E(\mathbb{F}_p)|$.

When using a supersingular curve over $\mathbb{F}_p, p > 3$ we can take $\mathbb{G}_0 = \mathbb{G}_1$ as the subgroup containing all points of order q in $E(\mathbb{F}_p)$. The group \mathbb{G}_2 is the subgroup of order q of $\mathbb{F}_{p^2}^*$. The modified Weil pairing [BF01], denoted \hat{e} , is a non-degenerate bilinear map on $\mathbb{G}_0 \times \mathbb{G}_1$. Since the CDH assumption is believed to hold for the group \mathbb{G}_0 , we know by Claim 2 that \hat{e} is secure in the sense of Definition 2. Since the modified Weil pairing is symmetric (i.e. $\hat{e}(G, H) = \hat{e}(H, G)$ for all $H, G \in \mathbb{G}_0$) a collision finding algorithm for \hat{e} is immediate: given $G, H \in \mathbb{G}_1$ we output $H, G \in \mathbb{G}_0$ as a collision. Indeed, $\hat{e}(G, H) = \hat{e}(H, G)$ and hence H, G is a collision for G, H . Since \hat{e} is secure and there is an efficient collision finder we obtain the following corollary to Theorem 1:

Corollary 1. *The signature scheme of Section 5 instantiated with the modified Weil pairing \hat{e} is existentially unforgeable under a chosen message attack if the CDH assumption holds for the group $\mathbb{G}_0 = \mathbb{G}_1$ defined above.*

To obtain shorter signatures one can avoid using supersingular curves and instead use a family of curves due to Miyaji et al. [MNT01]. Curves E/\mathbb{F}_p , $p > 3$ in this family are not supersingular and have the property that if q divides $|E(\mathbb{F}_p)|$ then $E[q]$ is contained in $E(\mathbb{F}_{p^6})$. Let \mathbb{G}_0 be the subgroup of order q in $E(\mathbb{F}_p)$ and let \mathbb{G}_1 be some other subgroup of order q in $E(\mathbb{F}_{p^6})$. The Weil pairing e on $\mathbb{G}_0 \times \mathbb{G}_1$ is not degenerate. Furthermore, since CDH is believed to be hard on $\mathbb{G}_0 \times \mathbb{G}_1$, we know by Claim 2 that e is a secure bilinear map in the sense of Definition 2. To build a collision finder for e we use the trace map $\text{tr}: E(\mathbb{F}_{p^6}) \rightarrow E(\mathbb{F}_p)$. For almost all choices of \mathbb{G}_1 the map tr defines an isomorphism from \mathbb{G}_1 to \mathbb{G}_0 . Then, a collision finder for e works as follows: given $(G, H) \in \mathbb{G}_1$ it outputs as a collision the pair $(\text{tr}(G), \text{tr}(H)) \in \mathbb{G}_0$. Indeed, one can verify that $e(\text{tr}(G), H) = e(\text{tr}(H), G)$. Therefore, by Theorem 1 our signature scheme is secure when using this family of curves.

We note that the RSA function $r(x, d) = x^d \bmod N$ while being bilinear does not satisfy the condition of Theorem 1 since the orders of groups \mathbb{G}_0 and \mathbb{G}_1 are not known to the simulator. Nevertheless, the signature scheme can be instantiated with this function, which would yield a scheme similar to the Dwork-Naor scheme.

7 Conclusion

We presented a new signature scheme secure in the standard model (i.e. without random oracles) using groups in which the Computation Diffie-Hellman assumption holds. Our scheme can be implemented using any secure bilinear map (secure in the sense of Definition 2). Instantiating our signature scheme using the Weil or Tate pairings gives the most efficient known discrete-log type signature secure without random oracles.

References

- [BP97] N. Barić and B. Pfitzmann, “Collision-free accumulators and fail-stop signature schemes without trees,” Proc. of Eurocrypt’97, pp. 480–494, 1997. [100](#)
- [BF01] D. Boneh and M. Franklin, “Identity based encryption from the Weil pairing,” Proc. of CRYPTO’01, pp. 213–229, 2001. Also <http://eprint.iacr.org/2001/090/>. [99](#), [100](#), [108](#)
- [BLS01] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the Weil pairing,” Proc. of Asiacrypt’01, pp. 514–532, 2001. [99](#)
- [BR94] M. Bellare and P. Rogaway, “Optimal asymmetric encryption—how to encrypt with RSA,” Proc. of Eurocrypt’94, pp. 92–111, 1994. [101](#)
- [CD95] R. Cramer and I. Damgård, “Secure signature schemes based on interactive protocols,” Proc. of CRYPTO’95, pp. 297–310, 1995. [99](#), [102](#)

- [CD96] R. Cramer and I. Damgård, “New generation of secure and practical RSA-based signatures,” Proc. of CRYPTO’96, pp. 173–185, 1996. [98](#), [102](#), [103](#)
- [CS99] R. Cramer and V. Shoup, “Signature schemes based on the Strong RSA assumption,” Proc. of ACM CCS’99, pp. 46–51, 1999. Full version appeared in ACM TISSEC, v. 3(3), pp. 161–185, 2000. [98](#), [102](#), [103](#)
- [DN94] C. Dwork and M. Naor, “An efficient existentially unforgeable signature scheme and its applications,” Proc. of CRYPTO’94, pp. 234–246, 1994. Full version appeared in J. of Cryptology, v. 11(2), pp. 187–208, 1998. [98](#), [102](#), [103](#)
- [FFS88] U. Feige, A. Fiat, and A. Shamir, “Zero-knowledge proofs of identity,” J. of Cryptology, v. 1, pp. 77–94, 1988.
- [GHR99] R. Gennaro, S. Halevi, and T. Rabin, “Secure hash-and-sign signatures without the random oracle,” Proc. of Eurocrypt’99, pp. 123–139, 1999. [98](#), [102](#), [103](#)
- [GS02] C. Gentry and A. Silverberg, “Hierarchical ID-based cryptography”, Proc. of Asiacrypt’02, pp. 548–566, 2002. [99](#)
- [Gol86] O. Goldreich, “Two remarks concerning the Goldwasser-Micali-Rivest signature scheme,” Proc. of CRYPTO’86, pp. 104–110, 1986. [103](#), [104](#)
- [GMR84] S. Goldwasser, S. Micali, and R. Rivest, “A ‘paradoxical’ solution to the signature problem (extended abstract),” Proc. of FOCS’84, pp. 441–448, 1984. Journal version in [GMR88]. [101](#), [102](#)
- [GMR88] S. Goldwasser, S. Micali, and R. Rivest, “A digital signature scheme secure against adaptive chosen-message attacks,” SIAM J. on Computing, 17(2), pp. 281–308, 1988. [101](#), [110](#)
- [HL02] J. Horwitz and B. Lynn, “Towards hierarchical identity-based encryption”, Proc. of Eurocrypt’02, pp. 466–481, 2002. [99](#)
- [Jou00] A. Joux, “A one-round protocol for tripartite Diffie-Hellman,” Proc. of ANTS’00, pp. 385–394, 2000. [99](#)
- [Lys02] A. Lysyanskaya, “Unique signatures and verifiable random functions from DH-DDH separation,” Proc. of CRYPTO’02, pp. 597–612, 2002. [99](#), [102](#)
- [MRV99] S. Micali, M. Rabin, and S. Vadhan, “Verifiable random functions,” Proc. of FOCS’99, pp. 120–130, 1999. [99](#)
- [MNT01] A. Miyaji, M. Nakabayashi, and S. Takano, “New explicit condition of elliptic curve trace for FR-reduction,” IEICE Trans. Fundamentals, v. E84 A(5), May 2001. [109](#)
- [NY89] M. Naor and M. Yung, “Universal one-way hash functions and their cryptographic applications,” Proc. of STOC’89, pp. 33–43, 1989. [98](#), [102](#)
- [PS96] D. Pointcheval and J. Stern, “Security proofs for signature schemes,” in Proc. of Eurocrypt’96, pp. 387–398, 1996. [99](#)
- [Rom90] J. Rompel, “One-way functions are necessary and sufficient for secure signatures,” Proc. of STOC’90, pp. 387–394, 1990. [98](#), [102](#)
- [Ver01] E. Verheul, “Self-Blindable Credential Certificates from the Weil Pairing,” Proc. of Asiacrypt’01, pp. 533–551, 2001. [99](#)

Access Control Using Pairing Based Cryptography

Nigel P. Smart

Dept. Computer Science, University of Bristol, Merchant Venturers Building
Woodland Road, Bristol BS8 1UB, United Kingdom
nigel@cs.bris.ac.uk

Abstract. We present a mechanism to encrypt to an arbitrary collection of identities using a variant of the Boneh-Franklin identity based encryption scheme. The decryptor is defined by a logical formulae of conjunctions and disjunctions. This enables a simple mechanism to drive access control to broadcast encrypted data using user identities as the public keys.

1 Introduction

Controlling access to sensitive data or resources is of central importance in an information system. Often access is granted according to a number of different constraints depending on the privileges of the user. Many access control systems have been presented over the years to meet different user requirements. For example there is the simple ACL lists in Unix systems which assign user, group and world privileges to different files, Kerberos [5] which provides both entity authentication services and access control, or the role-based access control found in more modern systems.

Many systems require access control to be built in from the start. But what happens when a service is already available and one wishes to restrict who has access to its output? For example corporate firewalls may allow certain internal users access to the external Internet and other external users access to some of the internal Intranet. Such a control service is often implemented as a proxy, this allows legacy applications to function in their usual way, with access control decisions taken by the proxy. The proxy can then be configured to maintain whatever access control policy is desired.

In this paper we examine such a system, see Figure 1, whereby we have an information source and a set of possible applications (or users). The applications and the information source are assumed to be provided by legacy systems which allow no sophisticated form of access control. Hence, to impose an access control policy on the source we insert proxies. One proxy sits between the information source and the public network. This proxy will encrypt the information depending on who the policy says should have access to the data. Before the application receives the data it will be decrypted by a client side proxy which will decrypt the data only if the client has the required privileges for the given data.

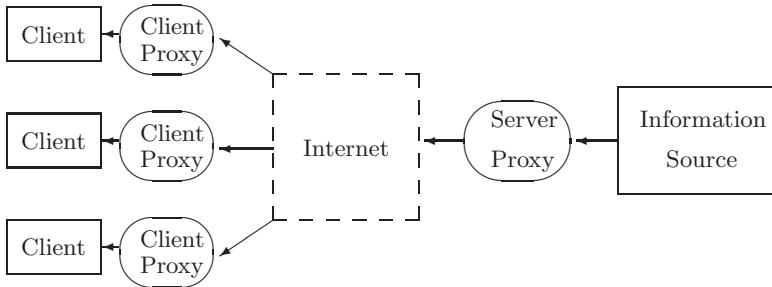


Fig. 1. System configuration

Such a system is particularly simple to envisage yet provides some interesting cryptographic challenges. For example the server side proxy needs to encrypt the data to possibly many subsets of users in a way which allows the required subsets to decrypt. Yet this needs to be done in a relatively cheap manner. The following examples show that standard symmetric or public key techniques will not scale or be usable.

- Suppose there are two users, Alice and Bob, who are both allowed to decrypt the information. Then it would appear either the information needs encrypting twice or Alice and Bob need to share a secret key.
- Now suppose that there is a single user who needs to obtain two pieces of authorization before they can access the data. Standard cryptographic techniques would seem to imply the need for an “onion”-type protocol in which the data is encrypted twice, or the use of a secret sharing scheme.
- If the server proxy is to encrypt to a particular user or group of users it needs to obtain authentic copies of the symmetric or public keys those users will be using. In a Kerberos type system this is achieved using an online trusted third party to distribute symmetric keys, in a public key system one uses digital certificates. However, both systems would provide a considerable strain on such a server side proxy when a large number of users are going to access the information source.
- Setting up secure SSL-like channels between the application and the server proxy is often too expensive, especially when the server proxy wishes to broadcast the same encrypted data to a large user population.

Of course in the above examples when one says “encrypting the data” one clearly means only encrypting the session key which encrypts the actual data. This makes the problem easier but still complicated.

Notice that we assume the server proxy knows precisely who should have access to the data, and the only role of the client proxies should be to obtain the required secret keys. It is this observation which leads us naturally to an identity based system, since the encryptor needs only know who is allowed to decrypt, it is up to the decryptor to obtain the required key or keys.

In this paper we discuss such a system in more detail and explain how the use of pairing based cryptographic systems, such as the identity based encryption scheme of Boneh and Franklin [1] can be used to describe a rich access control structure. We fully describe how all possible access control structures can be described in this system, how the number of expensive pairing operations can be minimised and also how the bandwidth can also be kept to a minimum.

This paper continues the ideas explained in the paper of Chen, Harrison, Smart and Soldera [2], where the use of multiple identities and trust authorities allowed various “virtual” keys to be built up from a disjunction and conjunction of existing keys. In the paper of Chen et. al. only a simple form of disjunction was allowed and the resulting access control structures were very limited, consisting of a restricted disjunctive normal form.

In the current paper we restrict to the case of a single trust authority, in which case the full range of access control structures can be realised, as we shall explain in Section 4. Before this we set up some notation in Section 2 related to pairing based systems. Then in Section 3 we relate the type of policy structures one would want to obtain to a variant of Hilbert’s positive propositional logic in a natural way and to a generalization of the conjunctive and disjunctive normal forms, called a conjunctive-DNF or CDNF for short. In Section 5 we show how the bandwidth of our solution can be reduced by minimizing the number of disjunctions within a CDNF.

2 Notation

Let G_1 and G_2 denote two groups of prime order q in which the discrete logarithm problem is believed to be hard and for which there exists a computable bilinear map

$$t : G_1 \times G_1 \longrightarrow G_2.$$

We shall write G_1 with an additive notation and G_2 with a multiplicative notation, since in real life G_1 will be the group of points on an elliptic curve and G_2 will denote a subgroup of the multiplicative group of a finite field.

We recap on the Boneh and Franklin encryption algorithm. We define the following cryptographic hash functions

$$\begin{aligned} H_1 &: \{0, 1\}^* \longrightarrow G_1, \\ H_2 &: G_2 \longrightarrow \{0, 1\}^*. \end{aligned}$$

We require the following two types of keys:

- A standard public/private key pair is a pair (R, s) where $R \in G_1$ and $s \in \mathbb{F}_q$ with

$$R = sP$$

for some given fixed point $P \in G_1$.

- An identifier based key pair is a pair $(Q_{\text{ID}}, S_{\text{ID}})$ where $Q_{\text{ID}}, S_{\text{ID}} \in G_1$ and there is some trust authority (TA) with a standard public/private key pair given by (R_{TA}, s) , such that the key pair of the trust authority and the key pair of the identifier are linked via

$$S_{\text{ID}} = sQ_{\text{ID}} \text{ and } Q_{\text{ID}} = H_1(\text{ID}),$$

where ID is the identifier string.

The scheme of Boneh and Franklin [1], allows the holder of the private part S_{ID} of an identifier based key pair to decrypt a message sent to her under the public part Q_{ID} . We present only the simple scheme which is only ID-OWE, for an ID-CCA scheme one applies the Fujisaki-Okamoto transformation [3].

Let m denote the message to be encrypted

- **Encryption :**

Compute $U = rP$ where r is a random element of \mathbb{F}_q . Then compute

$$V = m \oplus H_2(t(R_{\text{TA}}, rQ_{\text{ID}}))$$

Output the ciphertext (U, V) .

- **Decryption :**

Decryption is performed by computing

$$\begin{aligned} V \oplus H_2(t(U, S_{\text{ID}})) &= V \oplus H_2(t(rP, sQ_{\text{ID}})) \\ &= V \oplus H_2(t(P, Q_{\text{ID}})^{rs}) \\ &= V \oplus H_2(t(sP, rQ_{\text{ID}})) \\ &= V \oplus H_2(t(R_{\text{TA}}, rQ_{\text{ID}})) \\ &= m. \end{aligned}$$

3 Access Control Based on Knowing a Secret Key

We suppose a given entity, or group of entities, is defined by their knowledge of a secret key x_i . Using naive protocols one could then encrypt a message m to a disjunction of such entities by computing and sending

$$(E_{x_1}(m), E_{x_2}(m), \dots, E_{x_n}(m)).$$

Encrypting to a conjunction of entities can be performed using

$$(E_{x_1}(E_{x_2}(\dots E_{x_n}(m)\dots))).$$

Hence, in both cases encrypting data to a conjunction of n -entities or a disjunction of n -entities requires n applications of the encryption operation and can lead to (in the case of disjunction) a large increase in message size.

The ability to decrypt the above two ciphertexts implies knowledge of the given keys. Hence, one is able to describe access to the message m in terms of

a logical formula involving **ands** and **ors** over atomic elements which describe knowledge of a given key. Clearly, one can never show you do not know a given key hence the logical formulae used can never use logical not. In other words one can only describe positive outcomes, namely that some person has knowledge of a certain subset of keys rather than that they do not know something.

Hence, the logical formula obtained are precisely those used in a variant of Hilbert's positive propositional logic P_+ , [4]. In this logic only the following four connectives are allowed

- \Rightarrow , implication,
- \equiv , equivalence,
- \vee , disjunction,
- \wedge , conjunction.

In addition the allowed rules of deduction are Modus Ponens and substitution. Theorems in P_+ allow us to reason about when access control is allowed under conjunction and disjunction.

Clearly the use of only **and** and **or** means that every access control policy can be expressed in either conjunctive normal form (CNF)

$$\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} s_{i,j} \right),$$

or disjunctive normal form (DNF)

$$\bigvee_{i=1}^n \left(\bigwedge_{j=1}^{m_i} s_{i,j} \right),$$

by simply applying De Morgan's law. However, it turns out that in our system we would like a form which minimizes the number of **ors**. By examining simple expressions such as

$$(a \vee b) \wedge (c \vee d) \text{ and } (a \wedge b) \vee (c \wedge d),$$

one sees that sometimes the CNF minimizes the number of **ors** and sometimes it is the DNF which minimizes the number of **ors**.

Hence, we find it convenient to introduce the following, non-unique, form for expressions containing only **ands** and **ors**. A CDNF (or conjunctive-DNF) is an expression of the form

$$\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} \left(\bigwedge_{k=1}^{m_{i,j}} s_{i,j,k} \right) \right).$$

A CDNF is clearly non-unique since on setting $n = 1$ one obtains the DNF and on setting $m_{i,j} = 1$ for all i, j one obtains the CNF.

4 Describing Access Control Policies

Let $s_{i,j,k}$ denote the statement that “an entity has access to the secret key corresponding to the identity $ID_{i,j,k}$ ”, with respect to the fixed single trust authority with public key

$$R = sP.$$

Hence, the private key corresponding to the identity $\text{ID}_{i,j,k}$ is given by

$$S_{i,j,k} = sQ_{i,j,k} \text{ where } Q_{i,j,k} = H_1(\text{ID}_{i,j,k}).$$

Suppose we wish to grant access to data (i.e. an ability to decrypt encrypted data) to a set of entities which satisfy the policy given by the CDNF

$$P = \bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} \left(\bigwedge_{k=1}^{m_{i,j}} s_{i,j,k} \right) \right).$$

The actual data we shall assume is encrypted under a session key m , hence we need to encrypt this session key so that only the required people may decrypt it.

4.1 Encryption Procedure

Let $A_i = (a_{j,k}^{(i)})$ denote a $m_i \times m_i$ non-singular matrix over \mathbb{F}_q whose first column is equal to one, and $B_i = (b_{j,k}^{(i)}) = A_i^{-1} \pmod{q}$. The encryptor computes

$$Q'_{i,j} = \sum_{k=1}^{m_{i,j}} Q_{i,j,k} \text{ for } 1 \leq j \leq m_i, 1 \leq i \leq n \quad (1)$$

and then

$$\begin{pmatrix} Q_{V_i} \\ T_{i,1} \\ \vdots \\ T_{i,m_i-1} \end{pmatrix} = B_i \cdot \begin{pmatrix} Q'_{i,1} \\ Q'_{i,2} \\ \vdots \\ Q'_{i,m_i} \end{pmatrix} \text{ for } 1 \leq i \leq n. \quad (2)$$

The values Q_{V_i} correspond to ‘‘virtual’’ identities corresponding to the DNF’s

$$\bigvee_{j=1}^{m_i} \left(\bigwedge_{k=1}^{m_{i,j}} s_{i,j,k} \right) \text{ for } 1 \leq i \leq n$$

and the $T_{i,j}$ correspond to ephemeral public keys needed to allow a disjunction. The encryptor then computes

$$Q_V = \sum_{i=1}^n Q_{V_i} \quad (3)$$

and generates a random ephemeral secret $r \in \mathbb{F}_q^*$. Finally the ciphertext is produced by computing and transmitting

$$\begin{aligned} U &= rP, \\ U_{i,j} &= rT_{i,j} \text{ for } 1 \leq j \leq m_i - 1, 1 \leq i \leq n \\ V &= m \oplus H_2(t(R, rQ_V)). \end{aligned}$$

Hence to produce a ciphertext one needs to compute

- One pairing computation.
- $1 - n + \sum_{i=1}^n m_i$ multiplications by r in the group G_1 , from the definition of $U_{i,j}$.
- $\sum_{i=1}^n \sum_{j=1}^{m_i} m_{i,j} + \sum_{i=1}^n m_i^3 + n$ additions in the group G_1 .
 - $\sum_{i=1}^n \sum_{j=1}^{m_i} m_{i,j}$ from equation (1).
 - $\sum_{i=1}^n m_i^3$ from equation (2).
 - n from equation (3).

If b is the bit-length of an encoding of an element of G_1 then the ciphertext is of size

$$|m| + b \left(1 - n + \sum_{i=1}^n m_i \right)$$

which should be compared to a ciphertext of length $|m| + b$ when one encrypts to a single individual. It is for these reasons that we wish to choose a CDNFS representation which minimizes $\sum_{i=1}^n m_i$.

Encryption Example Assume we have four identifiers/users in our system denoted by a, b, c and d . Let the trust authority have private key s and public key

$$R = sP$$

and let the secret keys corresponding to the identifies a, b, c, d be

$$S_a = sQ_a, S_b = sQ_b, S_c = sQ_c, S_d = sQ_d.$$

Suppose we wish to encrypt a message m to a party who either has access to the keys S_a and S_b or has access to the keys S_a, S_c and S_d . This corresponds to the CDNFS, with an obvious abuse of notation,

$$a \wedge (b \vee (c \wedge d)).$$

Following the encryption procedure as above we define the matrices

$$\begin{aligned} A_1 &= (1) & B_1 &= A_1^{-1} = (1), \\ A_2 &= \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} & B_2 &= A_2^{-1} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}. \end{aligned}$$

The values of $Q'_{i,j}$ are given by

$$Q'_{1,1} = Q_a, Q'_{2,1} = Q_b, Q'_{2,2} = Q_c + Q_d.$$

Which means that Q_{V_1}, Q_{V_2} and $T_{1,1}$ are defined by

$$\begin{aligned} Q_{V_1} &= Q'_{1,1} = Q_a, \\ Q_{V_2} &= Q'_{2,1} = Q_b, \\ T_{1,1} &= Q'_{2,2} - Q'_{2,1} = Q_c + Q_d - Q_b. \end{aligned}$$

Finally the virtual public key Q_V is given by

$$Q_V = Q_{V_1} + Q_{V_2} = Q_a + Q_b$$

which means the ciphertext is given by $(U, U_{1,1}, V)$ where

$$\begin{aligned} U &= rP, \\ U_{1,1} &= rT_{1,1} \\ V &= m \oplus H_2(t(R, rQ_V)). \end{aligned}$$

for some random ephemeral secret key r .

4.2 Decryption Procedure

To decrypt the receiver will know a certain subset of the $S_{i,j,k}$ and hence will know a certain subset of

$$S'_{i,j} = \sum_{k=1}^{m_i,j} S_{i,j,k} \text{ for } 1 \leq j \leq m_i, 1 \leq i \leq n. \quad (4)$$

The precise subsets corresponding to which of the formulae

$$\bigwedge_{k=1}^{m_i,j} s_{i,j,k} \text{ for } 1 \leq j \leq m_i, 1 \leq i \leq n$$

evaluate to true. Using the structure of the matrix $A_i = B_i^{-1}$ the decryptor can evaluate the values of the “virtual” identifiers

$$Q'_{i,j} = Q_{V_i} + \sum_{k=1}^{m_i-1} a_{j_i, k+1}^{(i)} T_{i,k} \text{ for } 1 \leq j \leq m_i, 1 \leq i \leq n.$$

Then, supposing that the decryptor knows the secret S'_{i,j_i} , for every value of i , they can then compute

$$U' = r \sum_{i=1}^n \left(\sum_{k=1}^{m_i-1} a_{j_i, k+1}^{(i)} T_{i,k} \right) = \sum_{i=1}^n \left(\sum_{k=1}^{m_i-1} a_{j_i, k+1}^{(i)} U_{i,k} \right) \quad (5)$$

and

$$S' = s \sum_{i=1}^n Q'_{i,j_i} = \sum_{i=1}^n S'_{i,j_i} \quad (6)$$

The decryptor can then compute the value of $t(R, rQ_V)$ via

$$\begin{aligned} t(R, rQ_V) &= t \left(R, r \sum_{i=1}^n Q_{V_i} \right) \\ &= t \left(R, r \sum_{i=1}^n \left(Q'_{i,j_i} - \sum_{k=1}^{m_i-1} a_{j_i, k+1}^{(i)} T_{i,k} \right) \right) \end{aligned}$$

$$\begin{aligned}
&= t \left(R, r \sum_{i=1}^n Q'_{i,j_i} \right) \cdot t \left(R, -r \sum_{i=1}^n \sum_{k=1}^{m_i-1} a_{j_i,k+1}^{(i)} T_{i,k} \right) \\
&= t \left(rP, s \sum_{i=1}^n Q'_{i,j_i} \right) \cdot t(R, -U') \\
&= t(U, S') \cdot t(R, -U').
\end{aligned}$$

Hence to decrypt ciphertext one needs to compute

- Two pairing computations.
- $\sum_{i=1}^n \sum_{j=1}^{m_i} m_{i,j} + \sum_{i=1}^n m_i + n$ additions in the group G_1 .
 - $\sum_{i=1}^n \min(m_{i,j}) \leq \sum_{i=1}^n \sum_{j=1}^{m_i} m_{i,j}$ from equation (4).
 - $(\sum_{i=1}^n m_i) - n$ from equation (5).
 - n from equation (6).

Plus the multiplications by $a_{j_i,k+1}^{(i)}$ from equation (5), which can be considered to be negligible, in terms of the overall computation, if the values of the $a_{j_i,k+1}^{(i)}$ are chosen to be small.

Decryption Example We return to our previous example where we had the four secret keys

$$S_a = sQ_a, S_b = sQ_b, S_c = sQ_c, S_d = sQ_d.$$

The ciphertext transmitted was $(U, U_{1,1}, V)$ and we wanted parties who knew either the keys S_a and S_b or the keys S_a , S_c and S_d to be able to decrypt the message. Clearly there are two cases to consider:

Case 1: Decryptor Knows S_a and S_b .

In our previous notation the decryptor knows

$$S'_{1,1} = S_a \text{ and } S'_{2,1} = S_b.$$

They compute

$$\begin{aligned}
U' &= \mathcal{O}, \\
S' &= S'_{1,1} + S'_{2,1} = S_a + S_b.
\end{aligned}$$

Then the encryptors input to the key derivation function H_2 can be computed by the decryptor via

$$\begin{aligned}
t(U, S') \cdot t(R, -U') &= t(U, S') \\
&= t(rP, s(Q'_{1,1} + Q'_{2,1})), \\
&= t(R, r(Q_a + Q_b)), \\
&= t(R, rQ_V).
\end{aligned}$$

Case 2: Decryptor Knows S_a , S_c and S_d .

Now we have that the decryptor knows

$$S'_{1,1} = S_a \text{ and } S'_{2,2} = S_c + S_d.$$

They can then compute

$$\begin{aligned} U' &= U_{1,1} = rT_{1,1}, \\ S' &= S'_{1,1} + S'_{2,2} = S_a + S_c + S_d. \end{aligned}$$

The encryptors input to the key derivation function H_2 can be computed via

$$\begin{aligned} t(U, S') \cdot t(R, -U') &= t(rP, s(Q_a + Q_c + Q_d)) \cdot t(R, -rT_{1,1}), \\ &= t(sP, r(Q_a + Q_c + Q_d)) \cdot t(R, -rT_{1,1}), \\ &= t(R, r(Q_a + Q_c + Q_d - T_{1,1})), \\ &= t(R, r(Q_a + Q_c + Q_d - (Q_c + Q_d - Q_b))), \\ &= t(R, r(Q_a + Q_b)), \\ &= t(R, rQ_V)). \end{aligned}$$

5 Minimizing the Bandwidth

As we have already explained minimizing the number of disjunctions in a CDNF is important in reducing both the computational cost and bandwidth of the above system. One heuristic method to achieve this goal is given below.

Firstly we express the CDNF in conjunctive normal form:

$$\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} \left(\bigwedge_{k=1}^{m_{i,j}} s_{i,j,k} \right) \right).$$

where the $m_{i,j}$ are initially set to be equal to one. Our heuristic algorithm then proceeds as follows.

We write the current value of the CDNF as

$$C = \bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} t_{i,j} \right)$$

where

$$t_{i,j} = \bigwedge_{k=1}^{m_{i,j}} s_{i,j,k}.$$

Then if

$$\bigvee_{j \neq j_1} t_{i_1,j} = \bigvee_{j \neq j_2} t_{i_2,j} \text{ for } i_1 \neq i_2$$

we write

$$C = \left[\bigwedge_{i \neq i_1, i_2} \left(\bigvee_{j=1}^{m_i} t_{i,j} \right) \right] \wedge \left[\left(\bigvee_{j \neq j_1} t_{i_1,j} \right) \bigvee (t_{i_1,j_1} \wedge t_{i_2,j_2}) \right]$$

To see this heuristic in operation consider the following simplification

$$\begin{aligned} (a \vee c) \wedge (a \vee d) \wedge (b \vee c) \wedge (b \vee d) &= [(a \vee c) \wedge (a \vee d)] \wedge [b \vee (c \wedge d)] \\ &= [a \vee (c \wedge d)] \wedge [b \vee (c \wedge d)] \\ &= (a \wedge b) \vee (c \wedge d) \end{aligned}$$

6 Conclusion

We have shown how pairing based systems can allow a simple way to define access control to an encrypted broadcast information source. A single encrypted stream can be made available to targetted groups of users in an inexpensive manner. It is easy to determine how to encrypt to a certain subset of keys and, since we are using an identity based system, it is easy for the encryptor to determine the public keys of all parties. The decryptor is then left with the cost of determining, and obtaining, the correct secret keys needed to decrypt the stream. Which is precisely where the cost should be, since it should be easy to protect data and the ownus should be on the recipient to decrypt the data.

References

- [1] D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advances in Cryptology – CRYPTO 2001*, Springer-Verlag LNCS 2139, 213–229, 2001. [113](#), [114](#)
- [2] L. Chen, K. Harrison, N. P. Smart and D. Soldera. Applications of multiple trust authorities in pairing based cryptosystems. In *InfraSec 2002*, Springer-Verlag LNCS 2437, 260–275, 2002. [113](#)
- [3] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *Advances in Cryptology – CRYPTO '99*, Springer-Verlag LNCS 1666, 537–554, 1999. [114](#)
- [4] L. H. Hackstaff. *Systems of Formal Logic*. D. Reidel Publishing, 1966. [115](#)
- [5] J. Kohl, B. Neumann and T. Ts'o. The evolution of the Kerberos authentication service. In *Distributed Open Systems*, IEEE Computer Society Press, 1994. [111](#)

NTRUSign: Digital Signatures Using the NTRU Lattice

Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher,
Joseph H. Silverman, and William Whyte

NTRU Cryptosystems
5 Burlington Woods, Burlington, MA 02144
`{jhoffstein, nhowgravegraham, jpipher, jsilverman, wwhyte}@ntru.com`

Abstract. In this paper we introduce NTRUSIGN, a new family of signature schemes based on solving the approximate closest vector problem (APPR-CVP) in NTRU-type lattices. We explore the properties of general APPR-CVP based signature schemes (e.g. GGH) and show that they are not immune to transcript attacks even in the random oracle model. We then introduce the idea of using carefully chosen perturbations to limit the information that is obtainable from an analysis of a large signature transcript. In the case of NTRUSIGN this can be achieved while maintaining attractive efficiency properties.

1 Introduction

Lattices have been studied by cryptographers for quite some time, both in the field of cryptanalysis (see for example [18, 19, 20]) and as a source of hard problems on which to build encryption schemes (see [1, 8, 10]). Interestingly, though, research about building secure and efficient *signature schemes* using the theory of lattices is extremely sparse in the cryptographic literature. An early scheme is due to Goldreich, Goldwasser and Halevi [8], who proposed that one could sign a message by demonstrating the ability to solve the approximate closest vector problem (APPR-CVP) reasonably well for a point in space (hereafter referred to as the *message digest point*) generated from a hash of the message, and verify by checking that the “close lattice point” returned was indeed a lattice point and that it was close enough to the message digest point to make forgeries impractical. However, this idea was not analyzed in detail by its authors.

Another public-key cryptosystem, NTRUENCRYPT [10], was proposed at roughly the same time as [8]. One of the (several) ways that NTRUENCRYPT can be viewed is as a lattice cryptosystem based on a particularly efficient class of convolution modular lattices, which we will refer to as *NTRU lattices*. In this paper we study signature schemes based on solving APPR-CVP in the special class of NTRU lattices. A private key for the NTRUENCRYPT encryption scheme consists of a good basis for an N -dimensional sublattice of a $2N$ -dimensional NTRU lattice, but in order to solve APPR-CVP efficiently for arbitrary message digest points one must know a full good basis for the lattice. A major contribution

of this paper is given in section 4.1 where we detail an efficient technique to generate a full basis from knowledge of the small half-basis, thereby validating this approach.

A second major undertaking of this paper concerns the security of signature algorithms based on APPR-CVP. Such algorithms do not give a zero-knowledge scheme, because a transcript of signatures leaks information about the private key. In section 5.2 we therefore introduce *perturbation techniques* as a general way to reduce the effectiveness of transcript analysis in APPR-CVP-based signature schemes. In the case of NTRUSIGN (section 5.3), the use of perturbations guarantees that the number of signatures required to extract useful information far exceeds any practical requirements.

A third idea proposed in this paper is to consider a slightly different class of convolution modular lattices, which we refer to as *transpose NTRU lattices*. These lattices allow for more efficient signing than the standard NTRU lattices, but have slightly different security considerations. See section 4.3 for details.

We now describe the organization of this paper. In section 2 we explain the basic operations behind NTRUSIGN in practical engineering terms. In sections 3 and 4 we describe the mathematics underlying the NTRU technology. We begin by describing the NTRU lattice¹ as a module of rank 2 over certain algebraic rings. We then explain how we can still perform standard lattice tasks, such as completing bases and solving APPR-CVP, by working over these rings.

Next, we turn to the security of NTRUSIGN. Due to page limitations, we have been forced to considerably condense this analysis. A far more detailed presentation can be found at [11]. We first consider an adversary who does not use any signatures generated by the private key. In section 6 we show that the underlying lattice problems are infeasible to attack (using the best known methods) when N is suitably large.

We then turn our attention to an attacker who has access to a transcript of signatures. In section 7 we show that any signature scheme based on solving APPR-CVP using a (secret) short basis is vulnerable to a transcript attack after some finite number of signatures have been obtained, even in the random oracle model. This has implications for both the GGH signature scheme [8] and NTRUSIGN. In section 7.2 we specifically concentrate on the information leaked by NTRUSIGN signatures, including discussion of the effectiveness of the perturbations of sections 5.2–5.3 as a countermeasure.

Further technical details are described in the appendices. Appendix A completes the proof of a theorem in section 4.1 and a lemma in section 4.2. Appendices B and C extend the analyses of sections 6 and 7 respectively. Appendix D details the requirement on the hash function used with NTRUSIGN. Appendix E gives preliminary performance figures.

It should be noted that there have been several other attempts to create a signature scheme using the mathematics underlying the NTRUENCRYPT cryptosystem [12], but they succumbed to successful cryptanalysis in [6] and most recently in [7]. The downfall of these approaches was that without knowledge

¹ Strictly speaking, the “NTRU module,” but we will continue to call it a lattice.

of a full basis of an NTRU lattice, additional structure needed to be added to the signature scheme. An attacker could exploit this structure, leading to both forgery of individual signatures and key recovery. These attacks on previous signature schemes either do not apply to NTRUSIGN or require an infeasible number of signed messages to be effective; see section 7 for details.

2 NTRUSign: An Engineering Specification

In this section we outline the basic operations of NTRUSIGN. For engineering purposes, NTRUSIGN is defined in terms of operations on the set R of polynomials of degree (strictly) less than N and having integer coefficients. (The parameter N is fixed.) The basic operations on these polynomials are addition and convolution multiplication. Convolution multiplication $*$ of two polynomials f and g is defined by taking the coefficient of X^k in $f * g$ to equal

$$(f * g)_k \equiv \sum_{i+j \equiv k \pmod{N}} f_i \cdot g_j \quad (0 \leq k < N).$$

In more mathematical terms, R is the quotient ring $R = \mathbb{Z}[X]/(X^N - 1)$. If one of the polynomials has all coefficients chosen from the set $\{0, 1\}$ we will refer to the convolution as being *binary*. If coefficients of the polynomials are reduced modulo q for some q , we will refer to the convolution as being *modular*.

We will also need to round numbers to the nearest integer and to take their fractional parts. For any $a \in \mathbb{Q}$, let $\lfloor a \rfloor$ denote the integer closest to a , and define $\{a\} = a - \lfloor a \rfloor$. If A is a polynomial with rational (or real) coefficients, let $\lfloor A \rfloor$ and $\{A\}$ be A with the indicated operation applied to each coefficient.

The basic operations of NTRUSIGN are as follows:

Key Generation — requires a source of (pseudo)random bits.

1. INPUT: Integers $N, q, d_f, d_g, B \geq 0$, and the string $t = \text{"standard"}$ or "transpose" .
2. GENERATE B PRIVATE LATTICE BASES AND ONE PUBLIC LATTICE BASIS:
Set $i = B$. While $i \geq 0$:
 - (a) Randomly choose $f, g \in R$ to be binary with d_f, d_g ones, respectively.
 - (b) Find small $F, G \in R$ such that $f * G - F * g = q$. Sections 3 to 4.2 give more detail on this process.
 - (c) If $t = \text{"standard"}$, set $f_i = f$ and $f'_i = F$. If $t = \text{"transpose"}$, set $f_i = f$ and $f'_i = g$. Set $h_i = f_i^{-1} * f'_i \pmod{q}$. Set $i = i - 1$.
3. PUBLIC OUTPUT: The input parameters and $h = h_0 \equiv f_0^{-1} * f'_0 \pmod{q}$.
4. PRIVATE OUTPUT: The public output and the set $\{f_i, f'_i, h_i\}$ for $i = 0 \dots B$.

Signing — requires a hash function $H : \mathcal{D} \rightarrow R$ on a digital document space \mathcal{D} . The properties required of this hash function are explored in appendix D. Signing also requires a norm function $\|\cdot\| : R^2 \rightarrow \mathbb{R}$ and a “norm bound” $\mathcal{N} \in \mathbb{R}$. For $(s, t) \in R^2$ we define $\|(s \pmod{q}, t \pmod{q})\|$ to be the minimal value of $\|(s + k_1 q, t + k_2 q)\|$ for $k_1, k_2 \in R$.

1. INPUT: A digital document $D \in \mathcal{D}$ and the private key $\{f_i, f'_i, h_i\}$ for $i = 0 \dots B$.
2. Pick a random r .
3. Set $s = 0$. Set $i = B$. Set $m_0 = H(D||r)$, where “ $||$ ” denotes concatenation. Set $m = m_0$.
4. PERTURB THE POINT USING THE PRIVATE LATTICES: While $i \geq 1$:
 - (a) Set $x = \lfloor -(1/q)m * f'_i \rfloor$, $y = \lfloor (1/q)m * f_i \rfloor$, $s_i = x * f_i + y * f'_i$.
 - (b) Set $m = s_i * (h_i - h_{i-1}) \bmod q$.
 - (c) Set $s = s + s_i$. Set $i = i - 1$.
5. SIGN THE PERTURBED POINT USING THE PUBLIC LATTICE:
Set $x = \lfloor -(1/q)m * f'_0 \rfloor$, $y = \lfloor (1/q)m * f_0 \rfloor$, $s_0 = x * f_0 + y * f'_0$, $s = s + s_0$.
6. CHECK THE SIGNATURE:
 - (a) Set $b = \|(s, s * h - m_0 \bmod q)\|$.
 - (b) If $b \geq \mathcal{N}$, pick another random r and go to step 3.
7. OUTPUT: The triplet (D, r, s) .

Verification — requires the same hash function H , norm function $\|\cdot\|$ and “norm bound” $\mathcal{N} \in \mathbb{R}$.

1. INPUT: A signed document (D, r, s) and the public key h .
2. Set $m = H(D||r)$.
3. Set $b = \|(s, s * h - m \bmod q)\|$.
4. OUTPUT: **valid** if $b < \mathcal{N}$, **invalid** otherwise.

Remark 1. The recommended parameters

$$(N, q, d_f, d_g, B, t, \mathcal{N}) = (251, 128, 73, 71, 1, \text{"transpose"}, 310),$$

where $\|\cdot\|$ is the centered Euclidean norm (section 3) appear to give a practical and efficient signature scheme with 2^{80} security. We henceforth use $d = 72$ to denote the case where $d_f \approx d_g \approx 72$.

3 A View of NTRU: Background Mathematics

Underlying NTRU is the ring $R = \mathbb{Z}[X]/(X^N - 1)$. There is a natural lattice of dimension N associated with any element $r = \sum_{i=0}^N r_i X^i \in R$, namely the one generated by the rows of the following matrix:

$$\begin{pmatrix} r_0 & r_1 & \dots & r_{N-1} \\ r_{N-1} & r_0 & \dots & r_{N-2} \\ \vdots & \ddots & \ddots & \vdots \\ r_1 & r_2 & \dots & r_0 \end{pmatrix}.$$

It is easily checked that if M_r and M_s are the matrices corresponding to $r, s \in R$ then the matrix corresponding to $r * s \in R$ is given by $M_r M_s$; i.e. this matrix mapping is a ring isomorphism since it respects both addition and multiplication.

For each $q \in \mathbb{Z}$ and $h \in R$, the set $M_{h,q} = \{(u,v) \in R^2 \mid v \equiv u * h \pmod{q}\}$ is an R -module of rank 2. (Notice $M_{h,q}$ is also a lattice of dimension $2N$.) Every element of R has a unique representation as a polynomial $r = \sum_{i=0}^{N-1} r_i X^i$. Then a natural measure of size in R is the centered Euclidean norm of the vector of coefficients $\|r\|^2 = \sum_{i=0}^{N-1} r_i^2 - (1/N) \left(\sum_{i=0}^{N-1} r_i \right)^2$. The norm imposed on elements of $M_{h,q}$, or more generally on $(u,v) \in R^2$, is the component-wise Euclidean norm: $\|(u,v)\|^2 = \|u\|^2 + \|v\|^2$.

The element $h \in R$ is chosen so that $(f,g) \in M_{h,q}$ for some f and g of a special form. Assuming that f is invertible in R/qR , the lattice $M_{h,q}$ will contain (f,g) if we set $h = f^{-1} * g \pmod{q}$. The specified form of f and g is that they be binary elements of R in the sense that d_f of the coefficients of f (respectively d_g of the coefficients of g) are set to 1 and the rest are 0. Thus the norms of $f, g \in R$ and the norm of $(f,g) \in M_{h,q} \subset R^2$ are given by

$$\|f\| = \sqrt{d_f(1 - d_f/N)}, \quad \|g\| = \sqrt{d_g(1 - d_g/N)}, \quad \|(f,g)\| = \sqrt{\|f\|^2 + \|g\|^2}.$$

There is an obvious R -basis for $M_{h,q}$, namely $\{(1,h), (0,q)\}$. Since we know that $(f,g) \in M_{h,q}$, it is natural to ask if we can find another vector $(F,G) \in M_{h,q}$ so that the pair $\{(f,g), (F,G)\}$ is also an R -basis for $M_{h,q}$. This is possible if (and only if) $\gcd(\text{resultant}(f, X^N - 1), \text{resultant}(g, X^N - 1)) = 1$ – see section 4.1. A supplemental paper to this one will explain how this condition can be relaxed.

4 NTRUSign Key Generation

4.1 Completing the Basis

The general strategy for completing the basis of $M_{h,q}$ is to project f and g down to \mathbb{Z} via the resultant mapping, which respects multiplication. The definition of the resultant of f with $X^N - 1$, R_f , is the product of f evaluated at all the complex roots of $X^N - 1$. For basic properties of resultants, see for example [3, Chapter 3]. We here use the fact that

$$R_f \equiv \prod_{i=1}^{N-1} f(x^i) \pmod{\Phi} \in \mathbb{Z},$$

where $\Phi(X) = \sum_{i=0}^{N-1} X^i \in R$. Therefore we define $\rho_f \equiv \prod_{i=2}^{N-1} f(x^i) \pmod{\Phi}$, and ρ_g similarly. We then know that for some $k_f, k_g \in \mathbb{Z}[X]$,

$$\begin{aligned} \rho_f f + k_f(X^N - 1) &= R_f = \text{resultant}(f, X^N - 1) \pmod{\Phi}, \\ \rho_g g + k_g(X^N - 1) &= R_g = \text{resultant}(g, X^N - 1) \pmod{\Phi}. \end{aligned}$$

Assuming that R_f and R_g are coprime over the integers, we can now use the extended Euclidean algorithm to find $\alpha, \beta \in \mathbb{Z}$ such that $\alpha R_f + \beta R_g = 1$, in which case we have

$$(\alpha \rho_f)f + (\beta \rho_g)g = 1 + k(X^N - 1).$$

Thus if we set $F = -q\beta\rho_g$ and $G = q\alpha\rho_f$, then

$$f * G - g * F = q. \quad (1)$$

Theorem 1. Let $f, g, F, G \in R$ satisfy (1), let $h = f^{-1} * g \pmod{q}$, and let $M_{h,q}$ be the NTRU R -module generated by $\{(1, h), (0, q)\}$.

- (a) $\{(f, g), (F, G)\}$ form a basis for $M_{h,q}$.
- (b) If $F', G' \in R$ also satisfy $f * G' - g * F' = q$, then there is an element $c \in R$ so that $F' = F + c * f$ and $G' = G + c * g$.

Proof. Elementary linear algebra. For details, see Theorem 2 in Appendix A.

If we view $M_{h,q}$ as a matrix of generating rows, the above theorem can be seen to be a unimodular change of basis:

$$\begin{pmatrix} f & g \\ F & G \end{pmatrix} = \begin{pmatrix} f & (g - f * h)/q \\ F & (G - F * h)/q \end{pmatrix} \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix}.$$

With this notation we can define the *discriminant* of the R -module to be the determinant of the matrix, which can be seen to be an invariant modulo multiplication by a unit of R .

Although the F and G generated as in Theorem 1 complete a basis for $M_{h,q}$, they typically have very large coefficients. However, we can clearly remove any R -multiple of the vector (f, g) from (F, G) and still have an R -basis. In the next section we discuss how to find a good R -multiple by which to reduce.

4.2 Finding a Good Second Half for the Basis

We first discuss what we mean by APPR-CVP in the context of an R -module.

Definition 1. Let \mathcal{L} be a R -module of rank r , and let \mathbf{m} be an arbitrary element $\mathbf{m} \in (\mathbb{R}[X]/(X^N - 1))^r$. A vector $\mathbf{v} \in \mathcal{L}$ is said to be a solution of APPR-CVP if $\|\mathbf{v} - \mathbf{x}\| < \mathcal{N}$ for some suitably small $\mathcal{N} \in \mathbb{R}$.

In this paper the most common instance of APPR-CVP that will be considered is the case when the rank is $r = 2$, and the point \mathbf{m} will actually be restricted to lie in $\mathbf{m} \in R^2 \subset (\mathbb{R}[X]/(X^N - 1))^2$. It is important to note that if \mathbf{v} is a close vector to \mathbf{m} then $\mathbf{v} + \mathbf{w}$ is equally close to the vector $\mathbf{m} + \mathbf{w}$ for any $\mathbf{w} \in M_{h,q}$. Thus if one is given an arbitrary point $(m_1, m_2) \in R^2$ then it suffices to find a close vector $(w_1, w_2) \in M_{h,q}$ to the point $(0, m_2 - m_1 * h) \in R^2$. For this reason we will only consider solving APPR-CVP on points of the form $(0, m)$ where $m \in R$.

The approach we use to solve APPR-CVP in R -modules is equivalent to Babai's "inverting and rounding" approach [2] in lattices, i.e. we find the exact rational coordinates to solve the problem, and then round the coefficients to integers to obtain an R -module point.

For example, we can use this technique to reduce the (F, G) vector obtained in section 4.1 from completing the basis of $M_{h,q}$. In Appendix A we show that

reducing the first coordinate F will also usefully reduce the second coordinate G . Therefore it suffices to find a $k \in R$ such that $F_{red} = \|F - k * f\|$ is small. We know $f^{-1} = (1/R_f)\rho_f \in \mathbb{Q}[X]/(X^N - 1)$, and if we could take k to be equal to $l = F * f^{-1} \in \mathbb{Q}[X]/(X^N - 1)$, F_{red} would be 0. To obtain a $k \in R$, we take $k = \lfloor l \rfloor$, the rounding of l to the nearest integer.²

Since we know $F - l * f = F - (\lfloor l \rfloor + \{l\}) * f = 0$, i.e. $F - k * f = \{l\} * f$ we can use the following lemma to estimate the norm of the resulting vector.

Lemma 1. *Let X_1, \dots, X_N be independent continuous random variables that are uniformly distributed in the range $[-B/2, B/2]$, and let $Y = X_1^2 + \dots + X_N^2$. Then the mean and standard deviation of Y are given by the formulas*

$$\mu(Y) = \frac{NB^2}{12} \quad \text{and} \quad \sigma(Y) = \frac{B^2}{6} \sqrt{\frac{N}{5}}.$$

The proof of this lemma is left to appendix A, but we see that with $B = 1$ this lemma implies that the expected Euclidean norm of l is $\sqrt{N/12}$ (the centered norm will be very marginally smaller), and so using the pseudo-multiplicative property $\|F - k * f\| \approx \sqrt{N/12}\|f\| \approx \sqrt{Nd_f/18}$. Once we have reduced F in this fashion we can simply set $G = h * F \bmod q$ in the range $(-q/2, q/2]$, since we know G will also be small. One can further reduce the vector (F, G) by performing weak reduction (i.e. taking dot products) with rotations of the (f, g) vector, but in practice this appears not to be necessary. This therefore completes the description of “good” basis completion in the NTRU lattice.

Finally, we note that the APPR-CVP technique extends naturally to R -modules of higher dimension. For example in the case of $M_{h,q}$ we are trying to find $x, y \in R$ such that $x * (f, g) + y * (F, G) \approx (0, m)$. Over $\mathbb{Q}[X]/(X^N - 1)$ the exact solutions are given by

$$(x', y') = (0, m) \begin{pmatrix} f & g \\ F & G \end{pmatrix}^{-1} = \frac{1}{q}(0, m) \begin{pmatrix} G & -g \\ -F & f \end{pmatrix} = \left(\frac{-m * F}{q}, \frac{m * f}{q} \right),$$

thus we take $x = \lfloor x' \rfloor$ and $y = \lfloor y' \rfloor$ in which case

$$\|(0, m) - x * (f, g) - y * (F, G)\| = \|\{x\} * (f, g) + \{y\} * (F, G)\| \approx \frac{N\sqrt{2d_f}}{18},$$

assuming (F, G) is as above (i.e. after reduction by (f, g)).

4.3 The Transpose NTRU Lattice

The NTRU R -module is characterized by $h = f^{-1} * g \bmod q$, where f and g are binary elements of R . However, once we have obtained a collection of vectors

² By the mapping between elements of R and N -dimensional lattices described in section 3, we see that this is exactly equivalent to solving a simple N -dimensional APPR-CVP lattice problem using Babai’s inverting and rounding approach. The reason that the lattice corresponding to the matrix M_f is well reduced (which is essential for Babai’s technique) is that f is a relatively sparse binary element of R and thus highly orthogonal to its rotations.

f, g, F, G such that $f * G - F * g = q$ we can exchange the roles of F and g , and consider the R -module generated by the rows of the following matrix:

$$\begin{pmatrix} f & F \\ g & G \end{pmatrix} = \begin{pmatrix} f & (F - f * h')/q \\ F & (G - g * h')/q \end{pmatrix} \begin{pmatrix} 1 & h' \\ 0 & q \end{pmatrix},$$

where $h' = f^{-1} * F \bmod q$. We call the R -module generated by these rows the *transpose NTRU lattice*. In the transpose lattice, the shortest known vector is of the form (f, F) (i.e. with very small first coordinate, and reasonably small second coordinate) rather than (f, g) (i.e. both coordinates very small), so the lattice is significantly more efficient for signing operations, which now involve only multiplications of m with *binary* elements of R .

For verification, we could define any norm we like in this R -module, but there appears to be no advantage to changing from the standard centered Euclidean norm. However, the fact that one coordinate of the signature will be much smaller than the other affects transcript analysis; this is analyzed in section 7.

5 Signing and Verification Reviewed

5.1 The Case of No Perturbations ($B = 0$)

With the above mathematical background, we can see that, when $B = 0$, the specification in section 2 is simply a description of solving APPR-CVP within the norm bound \mathcal{N} for the R -module $M_{h,q}$. The only “trick” is that in the NTRU lattice it is sufficient to just give the first module coordinate s as the signature, since all such module points are of the form $(s, h * s + kq)$ and the k which makes this as close to $(0, m)$ as possible can be trivially obtained by a modulo q reduction.

For NTRUSIGN with the parameters $(N, q, d, \mathcal{N}, B) = (251, 128, 72, 310, 0)$, the typical experimental size of a signature is about 210.

5.2 Perturbation Techniques

In section 7 it is shown that general lattice based APPR-CVP signing algorithms are not zero knowledge, and that an attacker can average a finite transcript to obtain the private information. Section 7 shows that the averages involved converge quite fast for a signing algorithm based on APPR-CVP alone. To make these schemes practical, we must increase the length of transcript required. Since these schemes are not zero-knowledge, it is hard to quantify information leakage: our aim at the 2^{80} security level is to require an attacker to assemble at least 10^{10} signatures to mount a meaningful attack.

For general APPR-CVP signing algorithms, we propose that the signer first hashes the digital document D to obtain a point \mathbf{m} , and then “perturbs” this point to $\mathbf{m}' = \mathbf{m} + \epsilon$ by adding a point ϵ . This ϵ is different for each message and not known to the verifier. The signer then signs \mathbf{m}' ; if ϵ is small enough, a signature on \mathbf{m}' will also be a valid signature for \mathbf{m} .

At a minimum, this will slow down the convergence of any transcript averages. However, for full effectiveness, we would like it to be impossible for an attacker to distinguish between those parts of the averages due to the perturbations and those parts due to the “true” signatures.

Therefore, in APPR-CVP signing algorithms, we propose that the perturbations be generated using a signing process in one or more secret lattices of similar geometry to the public lattice. In the case of GGH, each signer would know a good basis \mathbf{b}_1 for an entirely secret lattice. To sign, the document would be hashed to a point \mathbf{m} , then “signed” using \mathbf{b}_1 to obtain \mathbf{m}_1 , a point in the secret lattice. Then \mathbf{m}_1 would be signed with the good basis \mathbf{b} of the public lattice. If the signer knew B secret lattices, the good basis of each would be applied in turn to the result of signing in the previous lattice.

This process will clearly multiply the average norm of signatures by \sqrt{B} , and there is a limit to the number of perturbations that can be applied before a validly-generated signature will in general exceed \mathcal{N} . Designers of APPR-CVP-based systems should ensure that \mathcal{N} is sufficiently large, if possible, to allow the use of at least one secret lattice.

5.3 NTRUSign with Perturbations

In the case of NTRUSIGN the use of perturbations is already specified in section 2. Here, we start from a digest point $(0, m)$ and find a close point (s_B, t_B) to this in the lattice generated by $\{(1, h_B), (0, q)\}$. We only need to find the first coordinate, since $t_B = s_B * h_B \bmod q$.

For subsequent bases B_i we now want to find the closest point (s_i, t_i) to (s_{i+1}, t_{i+1}) in the lattice generated by $\{(1, h_i), (0, q)\}$. Again, we can transform (s_B, t_B) by a lattice vector to a point $(0, m')$, with

$$\begin{aligned} m' &= t_B - s_B * h_{B-1} \bmod q \\ &= s_B(h_B - h_{B-1}) \bmod q, \end{aligned}$$

as in steps 4a and 4b of the signing process in section 2.

If we consider the first coordinate only, then the difference between the given point and the lattice point is s_i for each $i = B, \dots, 0$. Thus we define $s = \sum_{i=0}^B s_i$, and the final signature point is $(s, h_0 * s \bmod q)$.

Perturbations increase signing time in two ways: first, by requiring the solution of APPR-CVP in several lattices; second, because they increase the average norm of signatures and therefore the chance that the norm of a validly-generated signature will exceed \mathcal{N} . The variable r used in signing allows us to recover when a validly generated signature is too large. So long as the average norm of a validly generated signature is less than \mathcal{N} , no more than k attempts are needed to give a chance of $1 - 2^{-k}$ of successfully signing a message. We can therefore bound r above by 256 for practical purposes, and encode it in a single byte.

6 Security against a Transcriptless Adversary

6.1 Security of Private Keys

If the adversary is forced to work without the knowledge of any signed messages, key recovery from the public information is equivalent to finding small vectors in the NTRU lattice. This is the long standing hard problem that NTRUENCRYPT is based on. In the transpose lattice the situation is even harder for the adversary since the target small vectors (even after weighting) are considerably nearer the Gaussian heuristic of the lattice (and hence harder to find in practice by lattice reduction techniques). Of course any well reduced basis could be used for signing, but finding such a basis is a very hard problem for large N .

The measurements of [10, 11, 17] imply that for the parameters $(N, q, d) = (251, 128, 72)$, a lattice attack on the private key requires about $6.6 * 10^{12}$ MIPS-years. This corresponds to the strength of an 80-bit symmetric cipher [15].

6.2 Security against Forgery

We now consider the difficulty of forging a specific signature without knowledge of the private key. The typical way to attack this [6] is to use a combination of setting some coordinates and using lattice techniques to find others.

Consider a forger who picks a small s , with the hope that $m - sh \bmod q$ will have all small coefficients too. On average these coefficients will be more-or-less random modulo q , so using lemma 1 the average norm of an attempted forgery will be $q\sqrt{N}/12$. Since the asymptotic value of q is $O(N)$, the forgery will have norm $O(N^{3/2})$, which is the same order of magnitude as the (good) signature norm given at the end of section 4.2. For NTRUSIGN with no perturbations and the parameters $(N, q, d) = (251, 128, 72)$, experimentally generated signatures have an average norm of about 210. The average forgery norm can be calculated to be 585.40. Thus the security of NTRUSIGN appears to lie in the relative constants involved.

A more refined measure of the difficulty of forging a signature via this method is reflected by the number of standard deviations between an average forgery norm and the average norm of a true signature. Again using lemma 1 we find

$$\frac{\mu(\|\text{Forgery}\|^2) - \mu(\|\text{Good Sig}\|^2)}{\sigma(\|\text{Forgery}\|^2)} \approx \sqrt{\frac{5N}{4}} \left(1 - \frac{Nd}{6q^2}\right) \approx 0.87\sqrt{N}$$

for the parameters in question. So for a fixed ratio of $Nd/(6q^2) < 1$, the number of standard deviations from an average forgery to a true signature grows like $O(\sqrt{N})$.

Further information on the security against forgery, e.g. when the above techniques are used in conjunction with lattice reduction, is given in appendix B.

7 Transcript Leakage

7.1 The Security of appr-CVP Based Signature Schemes

In this section we consider the information available to an attacker who can obtain a large number of signatures generated by the same key.

First, consider general signature schemes based on solving APPR-CVP, such as GGH [8] and NTRUSIGN. Let us denote the private basis $\mathcal{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\}$, and let B be the matrix with rows corresponding to these vectors. If H is the matrix with rows corresponding to the public basis, then we know $B = UH$ for some unimodular matrix $U \in GL_n(\mathbb{Z})$.

Any APPR-CVP-type signature constructed using \mathcal{B} has the form $\mathbf{s} - \mathbf{m} = \epsilon_1 \mathbf{b}_1 + \dots + \epsilon_n \mathbf{b}_n$, where $\epsilon_1, \dots, \epsilon_n$ are (essentially) uniformly distributed in the interval $(-1/2, 1/2)$. (This follows from the fact that \mathbf{s} is obtained from \mathbf{m} by a process of rounding. The distribution of the ϵ s is constrained by the fact that the coordinates of $(\mathbf{s} - \mathbf{m})$ must be integers; however, experimentally, this distribution proves to be indistinguishable from the uniform distribution.) Therefore, a transcript is a random collection of points in a (centered) fundamental domain for the lattice spanned by the basis vectors \mathcal{B} .

Thus we can view \mathbf{s} as a vector valued random variable, and each signature in the transcript is a sample value of that random variable. The question becomes how to extract information about the basis from a transcript of signatures. Take an arbitrary bilinear form $F : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$. We can compute the expectation of $F(\mathbf{s}, \mathbf{s})$ as

$$E(F(\mathbf{s}, \mathbf{s})) = \sum_{i,j=1}^n E(\epsilon_i \epsilon_j) F(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{12} \sum_{k=1}^n F(\mathbf{b}_k, \mathbf{b}_k)$$

since $E(\epsilon_i \epsilon_j) = 1/12$ if $i = j$ and 0 otherwise.

The values $E(F(\mathbf{s}, \mathbf{s}))$ are second moments, since F is a form of degree 2. Thus an attacker can use a sufficiently long transcript to recover the second moment values $E(F(\mathbf{s}, \mathbf{s}))$ for every bilinear form.

However, the space of bilinear forms is itself a vector space of dimension n^2 , i.e. every bilinear form F is a linear combination of the bilinear forms E_{ij} defined by $E_{ij}(\mathbf{x}, \mathbf{y}) = x_i y_j$, so the maximum amount of information that an attacker can obtain from second moments is the set of n^2 values $E(F_{ij}(\mathbf{s}, \mathbf{s}))$. We have

$$E(F_{ij}(\mathbf{s}, \mathbf{s})) = \frac{1}{12} \sum_{k=1}^n F_{ij}(\mathbf{b}_k, \mathbf{b}_k) = \frac{1}{12} \sum_{k=1}^n b_{ki} b_{kj},$$

and these sums give 1/12 times the coordinates of the Gram matrix BB^t .

As shown in [7] if the bases B and H have integral entries, then this leaks the information UU^t where U is the unimodular transformation matrix. This contains essentially all the information about the basis U of \mathbb{Z}^n ; indeed this is effectively the internal information that the LLL lattice reduction algorithm uses when reducing. So if lattice reduction techniques could reduce this basis

fully to the *orthogonal* basis \mathbb{Z}^n then the implied transformation matrix would leak U itself, and hence B . It is a fascinating open problem to find any results about reducing lattices for which there is known to exist a completely orthogonal basis. Such lattices have been conjectured to be easier, but no results are known. However, we do not wish the security of NTRUSIGN to rest on the conjectured hardness of this problem. This is discussed in the next section, as is the generalization of the above computations to moments of higher order.

7.2 Transcript Leakage by NTRUSIGN

For NTRUSIGN without perturbations ($B = 0$), second moment information reveals the entries of an associated $2N$ by $2N$ Gram matrix. As noted, the problem of using this information to an attacker's advantage remains unsolved. Fourth moment information reveals the private key in polynomial time by a combination of simple algebra and a method of Gentry and Szydlo [7].

Experiments have determined that for the usual NTRU lattice second moment information can be obtained with transcripts on the order of 10,000 signatures. At least 100 million signatures are required to obtain the fourth moment, if a certain associated square root problem can be solved. If the square root problem is not solved, then the number of signatures required is increased considerably beyond 100 million. In the transpose lattice, the signatures are smaller and the moments converge faster to the appropriate integer values.

However, the use of correctly chosen perturbations adds two additional unknown basis vectors which must be eliminated by an attacker, postponing private key leakage until the next *two* even moments have converged. For example, with NTRUSIGN parameters $(N, q, d, B) = (251, 128, 72, 1)$ (one private basis), we find that obtaining the Gram matrix requires 6th moment convergence. The transcript length needed for this appears to be at least 10^{18} signatures, well above our aim of 10^{10} .

We would like to thank Craig Gentry and Mike Szydlo for pointing out that the fourth moment analysis needed to be extended and included in the present discussion. See appendix C for more details.

8 Conclusions and Open Problems

A fascinating open problem is to build an (efficient) lattice based signature scheme that leaks no useful transcript information to an adversary (e.g. provably zero knowledge). Note that such a scheme need not necessarily be based on the closest vector problem.

Another fundamental research question to do with NTRU lattices, is to try to utilize the factorization of $N - 1$. At present there is no evidence to suggest that this has an influence on security. The fact that none of the subrings of $\mathbb{Z}[\zeta]$, other than \mathbb{Z} , are Euclidean domains seems to be a serious obstacle.

A third fascinating problem is to devise a lattice reduction algorithm that efficiently finds an orthogonal basis of a lattice when one exists, even in high

dimensions. This too seems a hard problem, since the ordering of the vectors is critical to such an algorithm. Note however that with the use of perturbation techniques the security of NTRUSIGN does not need to rest on the hardness of this problem.

See appendix E for practical timing results of NTRUSIGN.

Acknowledgments

The authors would like to acknowledge much useful input from the cryptographic community while NTRUSIGN was in its draft stages. Useful conversations were had with Craig Gentry, Shai Halevi, Jyrki Lahtonen, Tommi Meskanen, Ari Renvall, Mike Szydlo and Adam Woodbury. Particular thanks go to Craig and Mike who have had a large input to this area of research.

References

- [1] M. Ajtai, C. Dwork, *A public-key cryptosystem with worst case/average case equivalence*. In Proc. 29th ACM Symposium on Theory of Computing, 1997, 284–293. [122](#)
- [2] L. Babai *On Lovász lattice reduction and the nearest lattice point problem*, Combinatorica, vol. 6, 1986, 1–13. [127](#)
- [3] H. Cohen, *A course in computational algebraic number theory*, GTM 138, Springer-Verlag, 1993. [126](#)
- [4] Wei Dai, Crypto++ 4.0 Benchmarks, <http://www.eskimo.com/~weidai/benchmarks.html>. [139](#)
- [5] Consortium for Efficient Embedded Security, *Efficient Embedded Security Standard #1*, available from <http://www.ceesstandards.org>. [139](#)
- [6] Craig Gentry, Jakob Jonsson, Jacques Stern, Michael Szydlo *Cryptanalysis of the NTRU Signature Scheme (NSS) from Eurocrypt ’01*, Advances in Cryptology—Asiacrypt ’01, Lecture Notes in Computer Science, Springer-Verlag, 2001. [123](#), [131](#)
- [7] C. Gentry, M Szydlo, *Cryptanalysis of the Revised NTRU Signature Scheme*, Advances in Cryptology—Eurocrypt ’02, Lecture Notes in Computer Science, Springer-Verlag, 2002. [123](#), [132](#), [133](#), [138](#)
- [8] O. Goldreich, S. Goldwasser, S. Halevi, *Public-key cryptography from lattice reduction problems*. In Proc. CRYPTO ’97, Lect. Notes in Computer Science 1294, Springer-Verlag, 1997, 112–131. [122](#), [123](#), [132](#)
- [9] D. Hankerson, J. L. Hernandez, A. Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Fields*, Cryptographic Hardware and Embedded Systems - CHES 2000, LNCS 1965, C. K. Koc and C. Paar (eds), Springer-Verlag, 2000, 1–19. [139](#)
- [10] J. Hoffstein, J. Pipher, J. H. Silverman, *NTRU: A new high speed public key cryptosystem*, in Algorithmic Number Theory (ANTS III), Portland, OR, June 1998, Lecture Notes in Computer Science 1423 (J. P. Buhler, ed.), Springer-Verlag, Berlin, 1998, 267–288. [122](#), [131](#)

- [11] J. Hoffstein, N. Howgrave-Graham, J. Pipher, J.H. Silverman, W. Whyte *NTRUSign: Digital signatures using the NTRU lattice. Preliminary draft 2* http://www.ntru.com/NTRUFTPDocsFolder/NTRUSign_v2.pdf 123, 131, 138, 139
- [12] J. Hoffstein, J. Pipher, J. H. Silverman, *NSS: An NTRU Lattice-Based Signature Scheme*, Advances in Cryptology—Eurocrypt '01, Lecture Notes in Computer Science, Springer-Verlag, 2001. 123, 137
- [13] J. Hoffstein, D. Lieman, J. H. Silverman, *Polynomial Rings and Efficient Public Key Authentication*, in Proceeding of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99), Hong Kong, (M. Blum and C. H. Lee, eds.), City University of Hong Kong Press. 137, 138
- [14] J. Hoffstein, J. H. Silverman, *Polynomial Rings and Efficient Public Key Authentication II*, in Proceedings of a Conference on Cryptography and Number Theory (CCNT '99), (I. Shparlinski, ed.), Birkhauser. 137
- [15] A. K. Lenstra, E. R. Verheul, *Selecting Cryptographic Key Sizes*, Journal of Cryptology vol. 14, no. 4, 2001, 255-293. 131
- [16] T. Meskanen and A. Renvall, University of Turku, private communication. 139
- [17] A. May, J. H. Silverman, *Dimension reduction methods for convolution modular lattices*, in Cryptography and Lattices Conference (CaLC 2001), J. H. Silverman (ed.), Lecture Notes in Computer Science 2146, Springer-Verlag, 2001 131
- [18] P. Nguyen, *Cryptanalysis of the Goldreich-Goldwasser-Halevi Cryptosystem from Crypto '97*, Advances in Cryptology—Proceedings of CRYPTO '99, (August 15–19, 1999, Santa Barbara, California), M. Wiener (ed.), Lecture Notes in Computer Science, Springer-Verlag. 122
- [19] P. Nguyen and J. Stern, *Lattice Reduction in Cryptology: An Update*, ANTS 2000, pp 85-112. 122
- [20] A. Shamir, *A polynomial-time algorithm for breaking the basic Merkle-Hellman cryptosystem*. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, IEEE, 1982, 145–152. 122

A Further Analysis of NTRU Algebra

In this appendix we cover the pieces of mathematics that were a little too detailed for the main report. We start by giving a more detailed version of Theorem 1.

Theorem 2. Let $f, g \in R$, let $h \equiv f^{-1} * g \pmod{q}$, and let $M_{h,q}$ be the NTRU R -module generated by $\{(1, h), (0, q)\}$.

(a) Suppose that $F, G \in R$ satisfy $f * G - g * F = q$. Then $\{(f, g), (F, G)\}$ also form a basis for $M_{h,q}$.

(b) Suppose that $F', G' \in R$ also satisfy $f * G' - g * F' = q$. Then there is an element $c \in R$ so that $F' = F + c * f$ and $G' = G + c * g$.

Proof. (a) It suffices to check that the following change-of-basis matrix B has coefficients in R and has determinant equal to a unit in R :

$$B = \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} \begin{pmatrix} f & g \\ F & G \end{pmatrix}^{-1} = \frac{1}{q} \begin{pmatrix} 1 & h \\ 0 & q \end{pmatrix} \begin{pmatrix} G & -g \\ -F & f \end{pmatrix} = \begin{pmatrix} (G - F * h)/q & (-g + f * h)/q \\ -F & f \end{pmatrix}$$

The congruence $h \equiv f^{-1} * g \pmod{q}$ ensures that $(-g + f * h)/q \in R$, and the equation $f * G - g * F = q$ implies that $F^{-1} * G \equiv f^{-1} * g \pmod{q}$, so we also

get that $(G - F * h)/q \in R$. This proves that B has coefficients in R , and one easily calculates that $\det(B) = (f * G - F * g)/q = 1$, so B^{-1} also has coefficients in R . Therefore $\{(1, h), (0, q)\}$ and $\{(f, g), (F, G)\}$ generate the same R -module.

(b) First observe that $F' * G \equiv F * G'$ (mod q), since $F^{-1} * G = F'^{-1} * G' \equiv f^{-1} * g \equiv h$ (mod q). Set $c = (F' * G - G' * F)/q$. Then

$$\begin{aligned} F + c * f &= F + \frac{F' * G * f - G' * F * f}{q} \\ &= F + \frac{F' * (q + g * F) - F * (q + g * F')}{q} = F'. \end{aligned}$$

Similarly, $G + c * g = G'$, which completes the proof of the theorem.

We next give the proof of Lemma 1.

Proof. Let X be a random variable uniformly distributed in $[-B/2, B/2]$. Then

$$\mu(Y) = N\mu(X^2) = \frac{N}{B} \int_{-B/2}^{B/2} x^2 dx = \frac{NB^2}{12}.$$

Similarly,

$$\begin{aligned} \sigma^2(Y) &= N\sigma^2(X^2) = N(\mu(X^4) - \mu(X^2)^2) \\ &= \frac{N}{B} \int_{-B/2}^{B/2} x^4 dx - N \left(\frac{1}{B} \int_{-B/2}^{B/2} x^2 dx \right)^2 = \frac{NB^4}{180}. \end{aligned}$$

To see why it is sufficient to just reduce the first coordinate of (F, G) by (f, g) in key generation, one should observe that $f * G - g * F = q$, so

$$\|G * g^{-1} - F * f^{-1}\| = \|q(fg)^{-1}\| \approx \frac{q}{\|f\| \cdot \|g\|}.$$

For the parameter set $(N, q, d) = (251, 128, 72)$ this quantity is equal to 2.49, which means that on average, corresponding coefficients of $G * g^{-1}$ and $F * f^{-1}$ differ by only 0.157. Therefore, one obtains almost the same result whether one rounds $G * g^{-1}$ or $F * f^{-1}$.

In practice however one obtains a smaller result by treating the two components together. This corresponds to the standard lattice paradigm of multiplying a non-square basis by its transpose in order to be able to perform Babai's inverting and rounding technique. In this context the optimal k to reduce $\|(F, G) - k * (f, g)\|$ is

$$k = \left\lfloor \frac{\bar{f} * F + \bar{g} * G}{f * \bar{f} + g * \bar{g}} \right\rfloor,$$

where $\bar{f}(x) = f(1/x) \bmod X^N - 1$ and similarly for \bar{g} .

B Security against Forgeries

As mentioned in section 6 a forger could use lattice reduction to aid a preselection process. Specifically, he could preselect somewhat fewer than N coordinates and use lattice reduction techniques to find the remaining coordinates. Thus suppose that a forger preselects αN coordinates of s and t for some choice of $0 \leq \alpha \leq 1$. He then reduces a lattice of dimension $(2 - \alpha)N$ and determinant $q^{N(1+\alpha)}$ to make the remaining $(1 - \alpha)N$ coordinates as small as possible.³ As α increases, the fundamental ratio

$$\frac{\mathcal{N}}{\sqrt{\frac{(2-\alpha)N}{2\pi e} \cdot q^{(1+\alpha)/(2-\alpha)}}}$$

decreases, and when it passes below 1, the Gaussian heuristic says that it is very unlikely for any solutions to exist.

The case $\mathcal{N} = 310$ gives a cut off point of $\alpha = 0.3835$, which corresponds to a lattice of dimension 407. Thus a lattice reduction attack cannot hope to be reduced below dimension 407 (down from 502). At dimension 502, with $\alpha = 0$, experiments suggest a breaking time of greater than 10^{80} MIPS years. Further, as the dimension is reduced towards 407, the advantage gained from the reduction in dimension is offset by the decrease in the Gaussian ratio, causing predicted breaking time to increase.

C Further Transcript Analysis Experiments

In the particular case of NTRUSIGN without perturbations ($B = 0$), the signature description given in Section 7.1 translates into a collection of pairs of polynomials

$$\epsilon_1 * f + \epsilon_2 * F, \quad \epsilon_1 * g + \epsilon_2 * G$$

where the coefficients of ϵ_1 and ϵ_2 are more-or-less randomly distributed in the interval $[-1/2, 1/2]$.

Taking a simple average of a collection of signatures is not useful, since the average will be zero (or some other simple expression that reveals no useful information). However, there are other ways to take averages that introduce higher moments. The subject of moment averaging attacks was discussed in [12, 13, 14], and briefly mentioned in Section 7.1. Here we give more details.

Our tool here is the *reversal* $\bar{c}(X) := c(X^{-1})$ of a polynomial $c(X)$. The product $\hat{c}(X) = c(X) * c(X^{-1})$ of a polynomial with its reversal will often have a nontrivial average. The coefficients of $\hat{c}(X)$ involve products $c_i c_{i+k}$, so the polynomial $\hat{c}(X)$ is known as a *second moment polynomial*. Similarly, its square $\hat{c}(X)^2$ is a *fourth moment polynomial*.

³ Notice that $\alpha = 0$ corresponds to pure lattice reduction and $\alpha = 1$ corresponds to pure exhaustive search.

First, we look at the average of the second moment polynomials \hat{s} from a transcript of signatures s . These are equal to

$$\begin{aligned}\hat{s} &= (\epsilon_1 * f + \epsilon_2 * F) * (\bar{\epsilon}_1 * \bar{f} + \bar{\epsilon}_2 * \bar{F}) \\ &= \hat{\epsilon}_1 * \hat{f} + \hat{\epsilon}_2 * \hat{F} + \epsilon_1 * \bar{\epsilon}_2 * f * \bar{F} + \epsilon_2 * \bar{\epsilon}_1 * \bar{f} * F.\end{aligned}$$

As the number of signatures in the transcript goes to infinity, $\hat{\epsilon}_1$ and $\hat{\epsilon}_2$ (essentially) approach constants, and the cross terms (essentially) average out to zero. Hence by averaging the second moment polynomials over a sufficiently long transcript, an attacker may be able to recover the quantity

$$\hat{f} + \hat{F} = f * \bar{f} + F * \bar{F},$$

along with the remaining entries in the Gram matrix described in Section 7.1. Experiments indicate that in order to reconstruct this value, it is necessary to average a transcript consisting of on the order of 10,000 signatures.

The security of NTRUSIGN with no perturbations and transcripts of moderate length thus depends on the Gram matrix problem being hard. If an attacker cannot solve the Gram matrix problem efficiently, they need a longer signature transcript – as observed by [7] (and earlier, in a different context, [13]), a transcript long enough to yield accurate limiting averages of the fourth power moments should contain enough information to compromise the private key. In fact, an average of \hat{s}^2 eventually converges to a linear combination of the three quantities \hat{f}^2 , \hat{F}^2 , and $\hat{f} * \hat{F}$. If this limiting value can be determined sufficiently accurately, then it can be combined with the second moment information to recover \hat{f} . Finally, the attacker would apply a method of Gentry and Szydlo [7] to \hat{f} and $\hat{f} * h$ to recover f in polynomial time.

Note that the averages required for this attack to proceed will converge quite slowly. It has previously been noted [13], and extensive experiments in the present case have confirmed, that a practical attack would require more than 100 million signatures – possibly much more, if an attacker cannot solve an N -dimensional square-rooting problem described in [11].

We would like to thank Craig Gentry and Mike Szydlo for emphasising the importance of the fourth moment analysis. They also suggested some possible variations on second moment attacks by restricting to subtranscripts where the norms (or coefficients) of signatures meet certain boundary conditions. We have investigated this approach; it appears to require transcripts of at least similar length to fourth moment based attacks.

D Hash Function Considerations

When signing, we map a digital document D to a message representative $(0, m)$. This mapping is actually a two stage process. First a standard secure hash function H_1 is applied to D to give β -bit output $H_1(D)$. Next a (public) function

$$H_2 : (\mathbb{Z}/2\mathbb{Z})^\beta \longrightarrow (\mathbb{Z}/q\mathbb{Z})^N$$

Table 1. Comparison of NTRUSIGN, ECDSA, RSA

	NTRUSIGN-251	ECDSA-163	RSA-1024
Keygen (μs)	180,000	1424	500,000
Sign (μs)	500	1424	9090
Verify (μs)	303	2183	781

is applied to $H_1(D)$ to yield the message digest $m = H_2(H_1(D))$. We require that the mapping H_2 is “reasonably” uniform into the set of q^{2N} possible m s. For example, one possible instantiation of H_1, H_2 for $N = 251$, $q = 128$ is given in [5]. Here H_1 is SHA-1. H_2 is defined by taking

$$D' \equiv \text{SHA-1}(H_1(D) \| 0) \| \text{SHA-1}(H_1(D) \| 1) \| \dots,$$

where D' is at least N bytes long. Each coefficient of m is then generated by taking the low-order $\log_2(q)$ bits of the corresponding byte of D' .

We identify two potential attacks related to this mapping. First, if two digital documents D and D' map to two message representatives m and m' which are very close together, and a signer can be induced to sign both of them, then there the difference of the signatures might be a small element of the underlying lattice, and could reveal the private key [16]. Such a pair of documents would endanger all NTRUSIGN implementations using a common mapping H .

Alternatively, an attacker can attempt direct forgery by generating lattice points of the form $(u, uh \bmod q)$ for arbitrary (small) u . In this case, the attacker generates a large set \mathcal{L} of lattice points and a large set \mathcal{M} of message representatives, and checks to see if one of the lattice points in \mathcal{L} signs one of the message representatives in \mathcal{M} .

Both of these problems are analyzed in detail in [11]. For the key recovery attack, we find that for $(N, q) = (251, 128)$, an attacker will have to generate 2^{205} distinct messages before there is a 50% chance of finding two message representatives within $B_{\text{coll}} = 10$ of each other.

For the collision attack, we note that each lattice point generated signs all points within a radius \mathcal{N} . An attacker who generates n lattice points can sign at most a fraction $n \cdot C$ of all potential messages, where $C = \frac{\pi^{N/2}}{\Gamma(1+N/2)} \left(\frac{\mathcal{N}}{q}\right)^N$. A standard birthday paradox type argument shows that if the attacker generates n points and k messages, her chance of getting a collision is 50% when $kn \approx 1/C$. Thus, for the parameters $(N, q, \mathcal{N}) = (251, 128, 310)$, an attacker will have to generate 2^{80} lattice points and 2^{80} message representatives to have a 50% chance of forging a signature by this method.

E Performance

Table 1 compares the performance of NTRUSIGN, ECDSA and RSA on an 800 MHz Pentium machine. RSA times are from [4] and were obtained on the same machine as the NTRUSIGN times. ECDSA times are from [9] and were

scaled relative to the clock speed of the machine used in that paper. NTRUSIGN figures are obtained in the standard lattice with no perturbations; the figures for transpose lattice with one perturbation will, however, be comparable.

About the XL Algorithm over $GF(2)$

Nicolas T. Courtois¹ and Jacques Patarin^{1,2}

¹ CP8 Crypto Lab, SchlumbergerSema
36-38 rue de la Princesse, BP 45, 78430 Louveciennes Cedex, France
<http://www.nicolascourtois.net>
courtois@minrank.org
² PRISM, University of Versailles
45 av. des États-Unis, 78035 Versailles Cedex, France
Jacques.Patarin@louveciennes.sema.slb.com

Abstract. Several public key cryptosystems (HFE, Quartz, Sflash, etc.) are based on the problem MQ of solving a system of multivariate quadratic equations over a finite field. At Asiacrypt 2002, Courtois and Pieprzyk show that the MQ problem is also relevant to the security of AES.

At Eurocrypt 2000, Courtois, Klimov, Patarin and Shamir introduced the XL algorithm for solving MQ. They show that if the number of equations m is much larger than the number of variables n , such overdefined MQ systems can be easily solved. From their simplified and heuristic analysis it seemed that even when $m = n$, a variant of XL could still be subexponential. The exact complexity of the XL algorithm remained an open problem. Moreover, all their simulations has been done over $GF(127)$ and with $D < 127$, with D being the parameter of the XL algorithm.

At Asiacrypt 2002, an algorithm XSL, derived from XL, is introduced for the cryptanalysis of block ciphers [5]. Very little is known about the behaviour of XSL and we believe that one should study the XL algorithm itself first. In this paper we study the behaviour of XL for systems of quadratic equations over $GF(2)$. We show that the possibility to use the equations of the field $GF(2)$: $x_i^2 = x_i$ that are also quadratic, makes that the XL algorithm works better. We also introduce two improved versions of XL, called XL' and XL2, with an improved final step of the algorithm (that also can be used in XSL). We present an explanation for the linear dependencies that appear in the XL algorithm, and derive a formula for the number of linearly independent equations in XL or XL2. Then we run various computer simulations and observe that this formula is always verified. Apparently we are able to predict exactly the behaviour of XL, XL' and XL2 for random systems of equations. Due to the entanglement of linear dependencies, the analysis of XL becomes increasingly difficult, and XL may be really exponential for $m = n$.

Keywords: Multivariate quadratic equations, MQ problem, overdefined and exactly defined systems of multivariate equations, XL algorithm, Gröbner bases, XSL attacks on AES.

1 Introduction

In the perpetual search for hard problems on which to base cryptographic security, there is a growing interest in so called "multivariate problems". These problems are usually NP-hard. In terms of scalability of the systems, the best problems are those for which all known attacks are exponential: it is then sufficient to increase slightly the parameter sizes, to keep up with progress in the attacks, or with an increase in the speed of computers. One of such problems is the problem MQ, of solving a system of multivariate quadratic equations over a small finite field.

Several public key cryptosystems based on MQ have been proposed, such as the cryptosystems of HFE family [12]. At Crypto'99, Shamir and Kipnis present a structural attack on HFE [13], in which they reduce the problem of recovering the secret key of HFE to a problem called later MinRank¹, see [4]. Later Courtois, evaluated the complexity of the Shamir-Kipnis attack, and presented two more efficient attacks, see [4]. Moreover, very recently Faugère demonstrated a new, and even more efficient attack on the basic HFE. With his new Gröbner bases algorithm F5, he was able to break the so called "HFE challenge 1" in a 96 hours of workstation time, see [7, 8], instead of 2^{62} for the best previous attack known from [4]. All these attacks use the fact that there is a trapdoor embedded in the set of quadratic equations. In this paper, on the contrary, we study generic attacks that solve the underlying MQ problem independently of the existence of the trapdoor. They apply also to random quadratic equations.

In the same paper at Crypto'99, Shamir and Kipnis present a surprising method called relinearization for solving overdefined systems of multivariate quadratic equations. They point out that, if such a system of equations is overdefined (much more equations than needed), then it can be solved much faster than expected. Subsequently, at Eurocrypt 2000 [14], Courtois, Klimov, Patarin and Shamir, present a new algorithm called XL, that can be seen as an improved version of relinearization. In the same paper, a variant of XL called FXL is introduced. It seems that, for systems of multivariate quadratic equations (MQ) over a small finite field, the FXL might be subexponential (unlike Gröbner bases algorithms), and this not only for overdefined systems, but even when $m = n$. However, in spite of the computer simulations, it was still unclear what is the exact complexity of XL and FXL. Recently a lot of interest in solving MQ systems over $GF(2)$ emerged, due to the Courtois-Pieprzyk attempts to break AES by such means [5].

In this paper we study the XL algorithm and concentrate on MQ over $GF(2)$. First we introduce some notations and define the MQ problem. In Section 3 we recall the original XL algorithm, and in Section 4 we study the necessary working condition for XL. In Sections 5 we present our computer simulations on XL and explain the results in Section 6. Then in Section 8 and 9 we concentrate on the final step of the algorithm, and give two improved versions of XL called XL' and XL2.

¹ As for MQ, MinRank is also a NP-hard problem.

2 Common Conventions and Notations

In this paper we will study solving systems of m multivariate equations with n variables over a small finite field $GF(q)$. We will use very similar notations than in [14]. The variables will be usually denoted by x_i and belong to $GF(q)$ with $q = 2$ unless otherwise stated, and in all cases q is very small, for example $q < 10$. We will **always** assume that the powers of variables are taken in $GF(q)$, i.e. reduced modulo q to the range $1, \dots, q-1$, because of the equation $x_i^q = x_i$ of the finite field $GF(q)$. Thus when $q = 2$ there will be no powers of x_i bigger than 1. We will assume that we want to solve the system for one particular output $b = (b_1, \dots, b_m)$, known in advance, and if $f_i(x_1, \dots, x_n)$ are the equations, we will systematically put $l_i(x_1, \dots, x_n) \stackrel{\text{def}}{=} f_i(x_1, \dots, x_n) - b_i$ so that the system to solve is:

$$\mathcal{A} : \begin{cases} l_1(x_1, \dots, x_n) &= 0 \\ &\vdots \\ l_m(x_1, \dots, x_n) &= 0 \end{cases}$$

The MQ Problem and Its Generalizations

The l_i will always be quadratic, and obviously they can also include terms of lower degree, i.e. linear and constant terms. In this case we call MQ the problem of finding one (not necessarily all) solutions to the above system. The MQ problem is NP-hard, see [9]. In this paper we assume that $m \geq n$. If $m > n$ the system is said to be overdefined. We will see that for a fixed n , the bigger is m , the more easy becomes the MQ problem. If $m < n$ the system is said to be underdefined, and efficient algorithms for the underdefined MQ has been studied in [3]. Thus we expect that the hardest case of MQ is when $m \approx n$.

In XL we will always assume that **the system has one and unique solution**, later, in Section 4.2 we explain why it is important. It happens frequently in cryptographic applications of MQ, and it is also the average number of solutions of a random MQ with $m = n$. Moreover, in practice, for systems that have several solutions, we can always reduce to a system in which there is one solution, by guessing a few variables.

Typically in cryptographic applications n is between 64 and 1600.

Manipulating the Equations

We will frequently refer to the equation $l_i(x_1, \dots, x_n) = 0$ as simply the equation l_i . Because the right hand of all our equations is always 0, it is very useful to identify a multivariate polynomial and an equation that says it is equal to 0. Thus the equation $x_1 \cdot l_2(x_1, \dots, x_n) = 0$ will sometimes be referred to as simply the equation $x_1 l_2$.

We say that the equations of the form $\prod_{j=1}^k x_{i_j} \cdot l_i = 0$, with all the i_j being pairwise different, are of type $x^k l$, and we call $x^k l$ the set of all these equations. For example the initial equations \mathcal{A} are of type l . We observe that

each solution x that satisfies all the equations l_i , also does satisfy all the equations of type $x^k l$, for any $k \geq 0$. Similarly we denote by x^k the set of all terms of degree exactly K , $\prod_{j=1}^K x_{i_j}$. It is a slightly modified extension of the usual convention $x = (x_1, \dots, x_n)$. By extension we define $x^0 = \{1\}$, the constant monomial.

Let $D \in \mathbb{N}$. We consider all the polynomials² $\prod_j x_{i_j} \cdot l_i$ of total degree $\leq D$. Let \mathcal{I}_D be the set of equations they span. \mathcal{I}_D is the linear space generated by all the $x^k l$, $0 \leq k \leq D - 2$. We have $\mathcal{I}_D \subset \mathcal{I}$, \mathcal{I} being the ideal spanned by the l_i (\mathcal{I} could be called \mathcal{I}_∞). We call \mathcal{T} the set of monomials, including the constant monomial, that appear in all the equations of \mathcal{I}_D , $\mathcal{T} = \bigcup_{i=0}^D x^i$.

Note: We do not study the equations of type $x^k l^{k'}$ because they are already included in $\mathcal{I}_{k+2k'}$.

3 The Basic Principle of XL

The XL algorithm is introduced in [14], in order to improve, and simplify, the very promising but less efficient relinearization method proposed by Shamir and Kipnis at Crypto'99 [13].

Let D be the parameter of XL algorithm.

Definition 1 (The XL algorithm). Execute the following steps:

1. **Multiply:** Generate all the products $\prod_{j=1}^k x_{i_j} \cdot l_i \in \mathcal{I}_D$ with $k \leq D - 2$.
2. **Linearize:** Consider each monomial in the x_i of degree $\leq D$ as a new variable and perform Gaussian elimination on the equations obtained in 1. The ordering on the monomials must be such that all the terms containing one variable (say x_1) are eliminated last.
3. **Solve:** Assume that step 2 yields at least one univariate equation in the powers of x_1 . Solve this equation over the finite fields (e.g., with Berlekamp's algorithm).
4. **Repeat:** Simplify the equations and repeat the process to find the values of the other variables.

4 The Necessary Condition for XL to Work

The XL algorithm consists of multiplying the initial m equations l_i by all possible monomials of degree up to $D - 2$, so that the total degree of resulting equations is D . With the notations introduced above, this set of equations is called \mathcal{I}_D . Let R be the number of equations generated in \mathcal{I}_D and T be the number of all monomials. We have, (the first term is dominant):

$$R = m \cdot \left(\sum_{i=0}^{D-2} \binom{n}{i} \right) \approx m \cdot \binom{n}{D-2}$$

² We recall that we systematically use the equations $x_i^2 = x_i$, which amounts to doing the computation in $GF(q)(x_1, \dots, x_n)/\{x_i^2 = x_i, i = 1 \dots n\}$, or equivalently making computations with powers reduced systematically modulo $q - 1 = 1$.

It is likely that not all of these equations are linearly independent, and we denote by $Free$ the exact dimension of \mathcal{I}_D . We have $Free \leq R$. We also have necessarily $Free \leq T$.

The basic principle of XL is the following: for some D we will have $R \geq T$. Then we expect that $Free \approx T$, as obviously it cannot be bigger than T . In [14], when $Free \geq T - D$, it is possible to obtain one equation with only one variable x_1 , and XL will succeed. In this paper, due to the equations of the field $x_i^2 = x_i$, only powers $1 \dots q - 1$ are possible, and the algorithm will work when $Free \geq T - \min(D, q - 1)$. In particular, over $GF(2)$, we get the following necessary condition: $Free \geq T - 1$.

Simplified Analysis of XL from [14]

In Section 6 of [14], R is evaluated as $R = m \cdot \frac{n^{D-2}}{(D-2)!}$ and T is evaluated as $\frac{n^D}{D!}$. The authors state that "if most of the equations are linearly independent" then XL will succeed as long as $R \geq T$, which gives that:

$$m \geq \frac{n^2}{D(D-1)}, \text{ and thus they obtain the (approximative) bound } D \geq \frac{n}{\sqrt{m}}.$$

A More Precise Evaluation for $D < q$

This part is in Appendix A, in this paper we are mainly interested in the case $q = 2$ and thus when $D \geq q$.

4.1 An Evaluation for $GF(2)$

In XL algorithm over $GF(2)$ we will always use the fact that $x_i^2 = x_i$. When we multiply the equations, we will use the equation $x_i^2 = x_i$ to eliminate all powers of x_i . Therefore the number of monomials of degree k is exactly $\binom{n}{k}$. We have:

$$T = |\mathcal{T}| = \sum_{i=0}^D |\mathcal{x}^i| = \sum_{\lambda=0}^D \binom{n}{\lambda} \text{ and } R = |\mathcal{I}_D| = m \left(\sum_{\lambda=0}^{D-2} \binom{n}{\lambda} \right).$$

Again if μ is the proportion of the equations that are linearly independent, $\mu = Free/R$, then XL algorithm will succeed if $R \times \mu \geq T$, i.e. when

$$m \left(\sum_{\lambda=0}^{D-2} \binom{n}{\lambda} \right) \times \mu \geq \sum_{\lambda=0}^D \binom{n}{\lambda}.$$

If we consider only the two main terms of the summation, this gives:

$$\begin{aligned} m \binom{n+1}{D-2} \times \mu &\geq \binom{n+1}{D} \\ m \frac{(n+1)!}{(D-2)!(n-D+3)!} \times \mu &\geq \frac{(n+1)!}{D!(n-D+1)!} \end{aligned}$$

Therefore we get:

$$m \geq \frac{(n-D+3)(n-D+2)}{D(D-1)\mu}$$

Thus when $D \ll n$ we get the same evaluation as before:

$$D \geq \text{ about } \frac{n}{\sqrt{m}\sqrt{\mu}}.$$

Then the total complexity of the XL attack is about:

$$T^\omega \approx \left(\frac{n}{n/\sqrt{m\mu}} \right)^\omega$$

with $\omega \leq 3$ being the exponent of the Gaussian reduction. Again one of the problems in XL is to see if μ is (or is not) close to 1.

4.2 General Theory and Moh's Comments on XL

In [11], Moh states that "From the theory of Hilbert-Serre, we may deduce that the XL program will work for many interesting cases for D large enough". According to [10], for big D we have $Free = T - \alpha$, with α being the number of solutions, and we always have $Free \leq T - \alpha$. Thus, on the one side, under our condition that the system has one and unique solution, we know that for D large enough we have $Free = T - 1$ as expected. On the other side, this condition is necessary, and when the system has several solutions, $Free = T - 1$ is never achieved and the basic XL cannot work.

In Section 4 of [11], Moh shows an interesting example on which the "general computation idea" of the basic XL always fails, for any D . For the precise XL algorithm that we defined here, using the equations $x_i^2 = x_i$, this kind of counter-example cannot occur, cf. [10]. This comes from the fact that in his example the projective variety associated to the set of solutions has a component "of strictly positive dimension contained in the hyperplane at infinity", and it is this property makes the process fail. In the case of XL, it cannot happen: the projective variety of the solutions has **no points** at infinity. This is because we included all the equations $x_i^2 = x_i$, and if we homogenize the system, we obtain $x_i^2 = x_i x_0$, then if we put $x_0 = 0$ we obtain $x_i = 0$ for all i , which is impossible in the projective space, cf. [10].

In our computer simulations we will see that indeed, for some D , XL always works for random systems over GF(2). Moreover, we will present improved versions of XL (called XL' and XL2) that succeed for some values D , for which the original XL fails.

5 Our Computer Simulations

Let $Free$ be the dimension of \mathcal{I}_D , i.e. the maximum number of equations that are linearly independent. Very little is known about the value of $Free$ for $D \geq 3$. In the paper that describes XL, the authors demonstrate that XL works with a series of computer simulations over GF(127) (and some more are given in the extended version of the paper [14]). In this section we will present some

computer simulations on the XL algorithm over $GF(2)$. No such simulations have been published so far.

Apparently we will be able to predict the exact value $Free$ obtained in these simulations. We will propose a formula for $Free$, that though not rigourously proven to work, is confirmed with good precision in all our experiments.

In all the simulations that follow we will pick a random system of linearly independent quadratic non-homogenous equations $y_i = f_i(x_1, \dots, x_n)$ and pick a random input $x = (x_1, \dots, x_n)$. Then we modify the constants in the system in order to have a system that gives 0 in x , i.e. we write a system to solve as $l_i(x_0, \dots, x_{n-1}) = 0$, for $i = 1, \dots, m$. If n is not too big, we also verify that the system has a unique solution, which is the case with good probability.

5.1 The Behaviour of XL for $D = 3$

In general it is not possible that $Free = R$. One reason is that $Free$ cannot exceed T . We have therefore always

$$Free \leq \min(T, R)$$

We have done various computer simulations with $D = 3$ and in our simulations, for $D = 3$, we have always³ $Free = \min(T, R)$ or $Free = \min(T, R) - 1$ or $Free = \min(T, R) - 2$.

In the following table we fix n and try XL on a random system of m linearly independent equations with growing m and with a fixed D .

n	10	10	10	10	10	20	20	20	20	64	64
m	10	14	16	17	18	20	40	60	65	512	1024
D	3	3	3	3	3	3	3	3	3	3	3
R	110	154	176	187	198	420	840	1260	1365	33280	66560
T	176	176	176	176	176	1351	1351	1351	1351	43745	43745
T'	92	92	92	92	92	382	382	382	382	4034	4034
$Free$	110	154	174	175	175	420	840	1260	1350	33280	43744
$\frac{Free}{R}$	1.000	1.000	.9886	.9358	.8838	1.000	1.000	1.000	.9890	1.000	.6572
$\frac{Free}{T}$.6250	.8750	.9886	.9943	.9943	.3109	.6218	.9326	.9993	0.7608	1.000
$\frac{Free}{T-T'}$	1.310	1.833	2.071	2.083	2.083	.4334	.8669	1.300	1.393	0.8381	1.102

Fig. 1. XL simulations for $D = 3$

n number of variables.

m number of equations.

D we generate equations of total degree $\leq D$ in the x_i .

³ Actually $Free$ is always the minimum of the two functions, except around the point where the two graphics meet, where we observe a "smooth" transition. Here the smooth transition is visible for $n = 10, m = 16, D = 3$.

R number of equations generated (independent or not).

T number of monomials of degree $\leq D$.

T' number of special monomials in the later XL2 algorithm defined in Section 8.3.

Free number of linearly independent equations among the R equations.

◊ Not that XL will work when $Free \geq T - D$.

◊ Similarly XL2 will work when $\frac{Free}{T-T'} > 1$ (see §8.3).

5.2 The Behaviour of XL for $D = 4$

When $D = 4$ we do not have $Free = \min(T, R)$ anymore.

We see that for $D = 4$ most of the equations are linearly independent. We observed that we have always:

$$\text{For } D = 4, \quad Free = \min\left(T, R - \binom{m}{2} - m\right)$$

The fact that $Free = R - \binom{m}{2} - m$ when $R - \binom{m}{2} - m \leq T$, means that, in all cases, there are $\binom{m}{2} + m$ linear dependencies between the equations in R . We are able to explain the origin (and the exact number) of these linear dependencies. Let l_i be the equations names (not expanded, just written as " l_i "), and let $[l_i]$ denote the expanded expression of these equations as quadratic polynomials. Then we have:

$$l_i[l_j] = [l_i]l_j$$

For each $i \neq j$, the above equation defines a linear dependency between the equations of XL. This explains the $\binom{m}{2}$ dependencies.

Example: For example if $l_1 = x_1x_3 + x_4$ and $l_5 = x_2x_1 + x_4x_7$ then the notation $l_1[l_5] = [l_1]l_5$ denotes the following linear dependency between the $l_ix_jx_k$:

$$l_1x_2x_1 + l_1x_4x_7 = l_5x_1x_3 + l_5x_4.$$

There are also other dependencies. They come from the fact that:

$$l_i[l_i] = l_i$$

n	10	10	10	20	20	20	20	20	20	40
m	5	10	11	20	24	28	30	32	36	128
D	4	4	4	4	4	4	4	4	4	4
R	280	560	616	4220	5064	5908	6330	6752	7596	105088
T	386	386	386	6196	6196	6196	6196	6196	6196	102091
T'	260	260	260	2320	2320	2320	2320	2320	2320	19840
$Free$	265	385	385	4010	4764	5502	5865	6195	6195	96832
$\frac{Free}{R}$.9464	.6875	.6250	.9502	.9408	.9313	.9265	.9175	.8156	.9214
$\frac{Free}{T}$.6865	.9974	.9974	.6472	.7689	.8880	.9466	.9998	.9998	.9485
$\frac{Free}{T-T'}$	2.103	3.056	3.056	1.035	1.229	1.420	1.513	1.598	1.598	1.177

Fig. 2. XL simulations for $D = 4$ (same notations as for Figure 1)

This explains the remaining m dependencies. For example if $l_1 = x_1x_3 + x_4$ we obtain that: $l_1 = l_1x_1x_3 + l_1x_4$.

5.3 The Behaviour of XL for $D = 5$

Following the method given in the previous chapter, we will try to predict the exact number of linearly independent equations that will be obtained for $D = 5$. First of all, we have the $\binom{m}{2} + m$ linear dependencies that are the same that existed for $D = 4$. Then we have also:

$$\begin{cases} l_i[l_j] = [l_i]l_j \\ l_i[l_i] = l_i \end{cases}$$

In addition we have:

$$l_i[l_j]x_k = [l_i]l_jx_k$$

It gives $n \cdot \binom{m}{2}$ dependencies. We also have:

$$l_i[l_i]x_j = l_ix_j$$

This gives nm additional dependencies. By inspection we check that for $D = 5$ we are unable to generate any more dependencies. From the above, we expect that

$$\text{For } D = 5, \quad \text{Free} = \min \left(T, R - (n+1) \binom{m}{2} - (n+1)m \right)$$

Is that all the linear dependencies ? We did some computer simulations to verify this, less than for $D = 3$ and 4 , because systems of equations become easily much bigger. All our simulations confirm the above formula.

n	20	20	22	22	22	24	24	24	27	27
m	10	20	10	22	27	16	27	32	27	37
D	5	5	5	5	5	5	5	5	5	5
R	13510	29722	17940	39468	48438	37200	62775	74400	89208	122248
T	21700	21700	35443	35443	35443	55455	55455	55455	101584	101584
T'	10072	10072	15094	15094	15094	21806	21806	21806	35804	35804
Free	12355	21699	16675	33649	35442	33800	53325	55454	78624	101583
$\frac{\text{Free}}{R}$.9145	.7301	.9295	.8526	.7317	.9806	.8495	.7543	.8814	.8310
$\frac{\text{Free}}{T}$.5694	1.000	.4705	.9494	1.000	.6095	.9616	1.000	.7740	1.000
$\frac{\text{Free}}{T-T'}$	1.062	1.866	.8195	1.654	1.742	1.005	1.585	1.648	1.196	1.544

Fig. 3. XL simulations for $D = 5$ (same notations as for Figure 1)

5.4 The Behaviour of XL for $D = 6$

In the extended version of this paper we show that $D = 6$, we have $\text{Free} = \min(T, R - [\binom{n}{2} + \binom{n}{1} + \binom{n}{0}] \cdot [\binom{m}{2} + \binom{m}{1}])$.

6 Our Results on XL over $GF(2)$

Our first observation is that unlike what happened in [14] for $GF(127)$, XL over $GF(2)$ works even if $m = n$, and with $D \ll 2^n$. For example for $m = n = 10$, XL works for $D = 4$ and for $m = n = 20$, XL works for $D = 5$. This comes from the fact that in [14] all the simulations were done over $GF(127)$, i.e. when $D \ll q = 127$, and thus the equations of the field $x_i^q = x_i$ were never used, while here the equations of the field $x_i^2 = x_i$ are systematically used. Due to these n additional quadratic equations, our results are similar to those of [14] with $m+n$ quadratic equations and n variables, instead of m equations. Therefore, unlike as in [14], where a dramatic change in the behaviour of XL was observed when $m = n$, $m = n + 1$ and $m = n + 2$, we do not observe such a behaviour here.

Another important observation is that very often, many general equations are not independent, even when the number R of generated equations is smaller than the total number of monomials T . For example when $D = 4$, $n = 20$ and $m = 28$, only 93.13% of the equations generated are independent, and when $D = 4$, $n = 20$ and $m = 30$, only 92.65% of the equations generated are independent. Therefore it is not possible to assume that "almost all" the generated equations are independent when $R \leq T$.

We have observed that the proportion of the linearly independent equations $Free/R$ slowly decreases when m increases, but the absolute value $Free$ increases. This proportion is also lower for increasing values of D .

We gave an algebraic explanation for the dependent equations in XL, and when $R \leq T$, for all our simulations we could predict their exact number, except in some rare cases, when R becomes very close to T .

7 The Asymptotic Behaviour of XL

As in [14] we expect that XL will be polynomial when $m = \mathcal{O}(n^2)$ and not polynomial when $\frac{m}{n^2} \rightarrow 0$. We do not yet have a proof of it. Despite some progress, it seems that it is difficult to evaluate the asymptotic complexity of XL when $m \approx n$. In this case it is not clear if XL is sub-exponential or fully exponential.

Can XL be Subexponential (for $m = n$) ?

The possibility of XL being sub-exponential even in the most difficult case, which is when $m = n$, was suggested in the original paper [14] and treated as a (not-very likely) possibility. At present we believe that when $m = n$, XL may be indeed truly exponential and it is possible that XL over $GF(2)$ is never better than the exhaustive search (when $m = n$).

It is possible to see that if XL were subexponential, it might give a subexponential attack on many cryptographically hard problems, such as cryptanalysis of block ciphers, see [5]. In general it is possible to see that XL is closely related to the elimination methods known in algebraic geometry as Gröbner bases. The

classical algorithm to compute a Gröbner basis is the Buchberger algorithm, see for example [6]. In general this algorithm has double exponential worst case complexity. However in the setting of multivariate cryptography we are only interested in solutions over the basis field $GF(q)$ and not in the algebraic closure. In this case the complexity can be cut down to single exponential worst case complexity just by adding the field equations $x_i^q = x_i$, exactly as we did for XL. It can be shown that applying Buchberger Algorithm to ideals of this form has single exponential worst case complexity (for this see [6] or [1]). However, the XL algorithm (as the algorithms F5 and F5/2 proposed by Faugère [7, 8]), does not need to compute a complete Gröbner basis, and for this reason, it could prove to be faster.

8 Improved Versions of XL: FXL, XL' and XL2

Sometimes XL fails because just a few equations are missing. In such cases, it is generally possible to still solve the system without increasing the degree D of the generated equations.

8.1 The FXL Algorithm

In [14] an improvement of XL called FXL is proposed. It consists of guessing the values of a few variables and applying XL. It was motivated by the fact that for a large q , the behaviour of XL changes dramatically when the number of equations exceeds very slightly the number of variables, i.e. when $m = n - \varepsilon$ with $\varepsilon = 1, 2, 3, \dots$. In this paper we work over $GF(2)$ and integrated n additional equations $x_i^2 = x_i$ into the algorithm, which makes, in a way, every quadratic system overdefined, and therefore we do not observe any such behaviour. Therefore the improvement that can be achieved by FXL is less interesting over $GF(2)$. In FXL, in order to guess the values of r variables, we multiply by 2^r the complexity of XL applied to a system of m equations with $n - r$ variables and therefore r has to be small. In the XL' method described below, the complexity will only be increased by 2^r , not multiplied by 2^r , for some small value of r .

8.2 The XL' Algorithm

Instead of repeating the whole XL, we propose to modify the last step of the XL algorithm. In XL we need to obtain at least one equation in only one variable x_1 . In XL' the goal is to obtain at least r equations with only r variables. Such a system will have on average one solution, and will be solved by the exhaustive search in about q^r . If the variables are in $GF(2)$, we can allow r up to about 40. After the exhaustive search step, and substitution of the r variables, all the other variables are computed by Gaussian reduction in the original system. We call XL' this modification of XL.

For example, for $D = 3$, $n = 20$ and $m = 60$, our implementation of XL algorithm generated 1260 linearly independent equations, and we have only 1351

monomials. Therefore XL fails for $D = 3$, we lack some $1351 - 1260 = 91$ equations. Still XL' will succeed for $D = 3$. There are 127 monomials of degree ≤ 3 in only 9 variables. Thus we can obtain, by Gaussian elimination, a system of about $127 - 91 = 28$ equations in only 9 variables. Then we find these variables by the exhaustive search, and then the values of all the other variables will be obtained by Gaussian reduction in the original system.

XL' will work with r variables at the end, if we have $Free > T - \phi + r$ with ϕ being the number of monomials of degree $\leq D$ in the given r variables x_1, \dots, x_r . For variables over $GF(2)$ we have $\phi = \sum_{i=0}^D \binom{r}{D}$.

Thus XL will work over with r variables and over $GF(2)$ when:

$$Free > T - \sum_{i=0}^D \binom{r}{D} + r \text{ instead of } Free > T - 2 \text{ for the original XL.}$$

8.3 The XL2 Algorithm

With XL it is possible to solve the system when $T - Free$ is very small, and with XL' for substantially bigger values of $T - Free$, but still $Free$ is very close to T . We present a new method, called XL2, in which the number of missing equations can be much bigger. Again, it consists of modifying the last step of XL. The idea of XL2 is to obtain at least one equation with a restricted set of monomials, and from this equations, to obtain a new equation that was not in the original system. This new equations will be combined with the old equations, produce another new equations of a special form, and this will be iterated as many times as possible. This general idea is realized as follows:

Let x_1 be one variable. We define T' as the set of all monomials t_i that are in T and such that we also have $x_1 \cdot t_i \in T$. We have $T' \approx 2 \binom{n}{D-1}$, as we may take all the monomials of degree $\leq D - 1$ and also all the monomials of degree $\leq D - 1$ multiplied by x_1 .

Now we assume that we have $Free \geq T - T' + C$ with $C > 1$, and we apply the following algorithm:

1. By one single Gaussian elimination we bring the system to a form in which each term is a known linear combination of the terms in T' .
2. We do the same pre-computation two times, for example with T' defined for x_1 and separately for x_2 .
3. In each of the two systems, we have a subsystem of C equations that contain only terms of T' . These new equations are probably **not** of the same kind that the initial equations generated in XL method: only combining all the equations one can obtain some information about the solution, parts of the system usually have many solutions.
4. In each of the two subsystems of exceeding C equations, we multiply each equation by x_1 and respectively x_2 . Then we substitute the expressions from point 1 in these to get some **other** equations that contain only terms of T' , but for the other variable.

5. These equations are expected to be new and different⁴. This is mainly because the equations from point 2 are believed to contain "some information" about the solution that is not in any small subset of R equations, it is necessary to combine about all the initial equations to eliminate so many terms.
6. Thus, if at the beginning $Free \geq C + T - T'$ we can "grow" the number of equations. At this moment we expect to have $2C$ additional equations, probably a little less in practice.
7. We expect that the number of new equations grows exponentially⁵.
8. If the initial system has a unique solution we expect that we will end up with $Free = T$ or very close to it.
9. For each equation containing only terms in T' , the cost to compute a derived additional equation will be about T'^2 . Since there are T' equations missing, we expect to do about T'^3 additional operations in the attack, which can probably be reduced to T'^ω and thus will be smaller than T^ω , the cost of the XL attack itself.
10. If the whole attack fails one should try with another couple of variables instead of x_1 and x_2 , or use three variables from the start (and three systems). We conjecture that three variables should always be sufficient. The number of possibilities grows very fast with the number of variables, a new equation obtained with one variable can be immediately transformed and expanded with **all** the other variables.

Examples: In Appendix B we give an explicit example to see how this method works.

Experimental Results on XL2 With this new method, T' can be almost as big as T . Our simulations show clearly XL2 is often more efficient than XL. For example on our Figure 1 we see that for $D = 3$ and $n = 10$, XL works for $m \geq 16$ and XL2 works already for $m \geq 10$. For $D = 3$ and $n = 20$, XL works for $m \geq 65$ and XL2 works already for $m \geq 20$. Finally, for $D = 4$ and $n = 20$, XL works for $m \geq 32$ and XL2 works already for $m \geq 20$. In all these examples we see that in practice XL2 gives much better results than XL.

8.4 Asymptotic Complexity of FXL, XL' and XL2

In this section, we saw on several examples, that XL' or XL2 may, in practice, give much better results than XL, because they will work for a value of D for which XL fails. We expect however that asymptotically all the version of XL are similar: as in [14] they are expected to be polynomial when $m = \mathcal{O}(n^2)$ and not polynomial when $\frac{m}{n^2} \rightarrow 0$.

⁴ Our computer simulations show that this heuristic works very well. New linearly independent equations are obtained. See the explicit example given in Appendix B.

⁵ Even if it grows only by 1 each time, the attack will work as predicted.

9 Conclusion

The problem MQ of solving a set of multivariate quadratic equations over a finite field arises in cryptography (allowing to propose new cryptographic primitives), but also in cryptanalysis (for example for AES). An interesting algorithm called XL is introduced in [14] for solving MQ. The original result was heuristic, and very little was known about the actual behaviour of XL. Moreover only computer simulations over $GF(127)$ were done in [14], while in most of the cryptographic (or cryptanalytic) applications, the equations are over $GF(2)$.

In this paper we have studied the XL algorithm over $GF(2)$. Two things make the algorithms more efficient than initially expected. First, we have shown that the existence of the field equations $x_i^2 = x_i$ increases the efficiency of XL, for example when $m = n$, we obtain that D is not 2^n anymore, and is much smaller. We have also shown how to improve the last step of the algorithm, and introduced two improved algorithms XL' and XL2. On concrete examples XL2 gives significantly better results than XL' and the original XL.

We were always able to explain the origin of the linear dependencies that appear in XL and to predict the number of non-redundant equations in XL. It seems that we can always predict the behaviour of XL over $GF(2)$, yet we do not have a proof that this prediction is always correct. Unfortunately it is still unclear if asymptotically XL is (or not) subexponential. Therefore this paper has no direct consequences on the security of block ciphers. It is however an important preliminary step towards understanding more complex cases such as XSL-type algorithms. It seems from the present work, that though not everything can be proven, it is possible to build a theory that predicts the behaviour of such algorithms with 100 % accuracy.

References

- [1] B. Barkee, D. C. Can, J. Ecks, T. Moriarty, R. F. Ree: *Why You Cannot Even Hope to use Gröbner Bases in Public Key Cryptography: An Open Letter to a Scientist Who Failed and a Challenge to Those Who Have Not Yet Failed*, in Journal of Symbolic Computation 18, 1994, S. 497-501. [151](#)
- [2] Don Coppersmith, Shmuel Winograd: "Matrix multiplication via arithmetic progressions"; J. Symbolic Computation (1990), 9, pp. 251-280.
- [3] Nicolas Courtois, Louis Goubin, Willi Meier, Jean-Daniel Tacier: *Solving Underdefined Systems of Multivariate Quadratic Equations*; PKC 2002, LNCS 2274, Springer, pp. 211-227. [143](#)
- [4] Nicolas Courtois: *The security of Hidden Field Equations (HFE)*; Cryptographers' Track Rsa Conference 2001, San Francisco 8-12 April 2001, LNCS2020, Springer-Verlag, pp. 266-281. [142](#)
- [5] Nicolas Courtois and Josef Pieprzyk, *Cryptanalysis of Block Ciphers with Overdefined Systems of Equations*; to be presented at Asiacrypt 2002, a preprint with a different version of the attack is available at <http://eprint.iacr.org/2002/044/>. [141](#), [142](#), [150](#)
- [6] Magnus Daum: *Das Kryptosystem HFE und quadratische Gleichungssysteme über endlichen Körpern*, Diplomarbeit, Universität Dortmund, 2001. Available from daum@itsc.ruhr-uni-bochum.de. [151](#)

- [7] Jean-Charles Faugère: *Computing Gröbner basis without reduction to 0*, , technical report LIP6, in preparation, source: private communication. Also presented at the Workshop on Applications of Commutative Algebra, Catania, Italy, 3-6 April 2002. [142](#), [151](#)
- [8] Jean-Charles Faugère: Report on a successful attack of HFE Challege 1 with Gröbner bases algorithm F5/2, announcement that appeared in `sci.crypt` newsgroup on the internet in April 19th 2002. [142](#), [151](#)
- [9] Michael Garey, David Johnson: *Computers and Intractability, a guide to the theory of NP-completeness*, Freeman, p. 251. [143](#)
- [10] Mireille Martin-Deshamps, private communication. [146](#)
- [11] T. T. Moh: *On The Method of XL and Its Inefficiency Against TTM*, available at <http://eprint.iacr.org/2001/047/>. [146](#)
- [12] Jacques Patarin: *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): two new families of Asymmetric Algorithms*; in Eurocrypt'96, Springer Verlag, pp. 33-48. [142](#)
- [13] Adi Shamir, Aviad Kipnis: *Cryptanalysis of the HFE Public Key Cryptosystem*; In Advances in Cryptology, Proceedings of Crypto'99, Springer-Verlag, LNCS. [142](#), [144](#)
- [14] Adi Shamir, Jacques Patarin, Nicolas Courtois, Alexander Klimov, *Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations*, Eurocrypt'2000, LNCS 1807, Springer, pp. 392-407. [142](#), [143](#), [144](#), [145](#), [146](#), [150](#), [151](#), [153](#), [154](#), [155](#), [156](#)

A An Evaluation of the Complexity of XL for $D < q$

In [14] the authors always had $D < q$, since all their simulations were done with $q = 127$. When $D < q$ there is no interaction with the equation of the field $GF(q)$, i.e. at no moment the algorithm uses the fact that $x_i^q = x_i$ and all monomials of degree up to D are allowed. A classical combinatorial result says that we have exactly $\binom{n+k-1}{k}$ monomials of degree k with n variables. Therefore the exact number of monomials of degree $\leq D$ is:

$$T = |\mathcal{T}| = \sum_{i=0}^D |x^i| = \sum_{\lambda=0}^D \binom{n + \lambda - 1}{\lambda} = \binom{n + D}{D}$$

In the same time, the number of generated equations is exactly

$$R = |\mathcal{I}_D| = m \left(\sum_{\lambda=0}^{D-2} \binom{n + \lambda - 1}{\lambda} \right) = m \binom{n + D - 2}{D - 2}$$

Now let μ be the proportion of the equations that are linearly independent, i.e. $\mu = \text{Free}/R$ with Free defined as the dimension of \mathcal{I}_D . The XL algorithm will succeed if $R \times \mu \geq T - D$, i.e. when

$$m \frac{(n + D - 2)!}{(D - 2)! n!} \mu - D \geq \frac{(n + D)!}{D! n!}$$

Thus when $D \ll n$ we get the following evaluation:

$$m \geq \frac{(n+D)(n+D-1)}{D(D-1)\mu} \quad \Rightarrow \quad D \geq \text{about } \frac{n}{\sqrt{m}\sqrt{\mu}}.$$

This evaluation is more general than the previous evaluation of [14], we do not have to assume that $\mu \approx 1$. One of the problems in XL is to see if μ is (or is not) close to 0. Indeed in our later simulation we will see that μ is generally not ≈ 1 . It is also more precise, since we used exact formulas for the number of monomials. The result is also very similar, we have $D \geq \mathcal{O}(\frac{n}{\sqrt{m}})$ and the total complexity of the XL attack is:

$$T^\omega \approx \left(\frac{n}{n/\sqrt{m\mu}} \right)^\omega$$

with $\omega \leq 3$ being the exponent of the Gaussian reduction.

B A Toy Example of the XL2 Trick

This is a concrete working example for the final step of the XL2 algorithm.

We have $n = 5$ variables, and thus $T = 16$ and $T' = 10$. We start with a random system that has exactly one solution, and with $Free > T - T'$ and with 2 exceeding equations, i.e. $Free = T - T' + 2$.

Here is a system in which T' is defined with respect to x_1 :

$$\begin{cases} x_3x_2 = x_1x_3 + x_2 \\ x_3x_4 = x_1x_4 + x_1x_5 + x_5 \\ x_3x_5 = x_1x_5 + x_4 + 1 \\ x_2x_4 = x_1x_3 + x_1x_5 + 1 \\ x_2x_5 = x_1x_3 + x_1x_2 + x_3 + x_4 \\ x_4x_5 = x_1x_2 + x_1x_5 + x_2 + 1 \\ 0 = x_1x_3 + x_1x_4 + x_1 + x_5 \\ 1 = x_1x_4 + x_1x_5 + x_1 + x_5 \end{cases}$$

Here is the same system in which T' is defined with respect to x_2 :

$$\begin{cases} x_1x_3 = x_3x_2 + x_2 \\ x_1x_4 = x_3x_2 + x_2 + x_1 + x_5 \\ x_1x_5 = x_2x_4 + x_3x_2 + x_2 + 1 \\ x_3x_5 = x_2x_4 + x_3x_2 + x_2 + 1 + x_4 + 1 \\ x_3x_4 = x_2x_4 + x_1 + 1 \\ x_4x_5 = x_1x_2 + x_2x_4 + x_3x_2 \\ 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \end{cases}$$

We have $rank = 8$. Now multiply the two exceeding equations of the first version of the system by x_1 .

$$\begin{cases} 0 = x_1x_3 + x_1x_4 + x_1 + x_1x_5 \\ 0 = x_1x_4 \end{cases}$$

We have $rank = 10$. We get two new linearly independent equations.

We rewrite these equations, using the second system, only with terms that can be multiplied by x_2 . Now we have 4 exceeding equations for the second system (two old and two new):

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5 + x_2 + 1 \\ 0 = x_3x_2 + x_2 + x_1 + x_5 \end{cases}$$

We multiply these four equations by x_2 .

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_2x_4 + x_2 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5x_2 \\ 0 = x_3x_2 + x_2 + x_1x_2 + x_2x_5 \end{cases}$$

We are not lucky, the second equation is invariant by this transformation. Still we get three new linearly independent equations. We have $rank = 13$.

We rewrite, using the first system, the three new equations with terms that can be multiplied by x_1 .

$$\begin{cases} 1 = x_1x_5 + x_2 + x_3 + x_4 \\ 1 = x_1x_2 + x_1x_3 + x_1x_5 + x_2 + x_3 + x_4 \\ 0 = x_3 + x_4 \end{cases}$$

Still $rank = 13$. Then we multiply the three new equations by x_1 :

$$\begin{cases} 1 = x_1x_5 + x_1x_2 + x_1x_3 + x_1x_4 \\ 1 = x_1x_5 + x_1x_4 \\ 0 = x_3 + x_4 \end{cases}$$

We have $rank = 14$. We get one more linearly independent equation. The two other are redundant. Now we rewrite the first equation with terms that can be multiplied by x_2 :

$$0 = x_1x_2 + x_2x_4 + x_3x_2 + x_1 + x_2 + x_5$$

We have still $rank = 14$. Then we multiply the new equation by x_2 :

$$0 = x_2x_4 + x_3x_2 + x_2x_5 + x_2$$

We get another new, linearly independent equation. We have $rank = 15$. The rank is the maximum that can be achieved, there are 15 non-zero monomials here, and $rank = 16$ can only be achieved for a system that is contradictory.

We note that in all these computations we have obtained additional equations without increasing the degree of the equations ($D = 2$ here).

Efficient $GF(p^m)$ Arithmetic Architectures for Cryptographic Applications

Guido Bertoni¹, Jorge Guajardo², Sandeep Kumar², Gerardo Orlando³,
Christof Paar², and Thomas Wollinger²

¹ Politecnico di Milano, P.zza L. Da Vinci 32
20133 Milano, Italy

bertoni@elet.polimi.it

² Communication Security Group, Ruhr-Universität Bochum
44780 Bochum, Germany

{guajardo,sandeep,cpaar,wollinger}@crypto.rub.de

³ General Dynamics Communication Systems
77 A St. Needham MA 02494, USA
Gerardo.Orlando@GDC4S.Com

Abstract. Recently, there has been a lot of interest on cryptographic applications based on fields $GF(p^m)$, for $p > 2$. This contribution presents $GF(p^m)$ multipliers architectures, where p is odd. We present designs which trade area for performance based on the number of coefficients that the multiplier processes at one time. Families of irreducible polynomials are introduced to reduce the complexity of the modulo reduction operation and, thus, improved the efficiency of the multiplier. We, then, specialize to fields $GF(3^m)$ and provide the first cubing architecture presented in the literature. We synthesize our architectures for the special case of $GF(3^{97})$ on the XCV1000-8-FG1156 and XC2VP20-7-FF1156 FPGAs and provide area/performance numbers and comparisons to previous $GF(3^m)$ and $GF(2^m)$ implementations. Finally, we provide tables of irreducible polynomials over $GF(3)$ of degree m with $2 \leq m \leq 255$.

1 Introduction

Galois field arithmetic has received considerable attention in recent years due to their application in public-key cryptography schemes and error correcting codes. In particular, two public-key cryptosystems based on finite fields stand out: elliptic curve (EC) cryptosystems, introduced by Miller and Koblitz [24, 19], and hyperelliptic cryptosystems, a generalization of elliptic curves introduced by Koblitz in [20]. Both, prime fields and extension fields, have been proposed for use in such cryptographic systems. However, until a few years ago the focus was mainly on fields of characteristic 2 due to the straight forward manner in which elements of $GF(2)$ can be represented, i.e., they can be represented by the logical values “0” and “1”. For these types of fields, both software implementations and hardware architectures have been studied extensively. In recent years, $GF(p^m)$ fields, where p is odd, have gained interest in the research community.

Mihăescu [28] and independently Bailey and Paar [3, 4] introduced the concept of Optimal Extension Fields (OEFs) in the context of elliptic curve cryptography. OEFs are fields $GF(p^m)$ where p is odd and both p and m are chosen to match the particular hardware used to perform the arithmetic, thus allowing for efficient field arithmetic. The treatment in [4, 28] and that of other works based on OEFs has only been concerned with efficient software implementations.

In [21, 32], $GF(p^m)$ fields are proposed for cryptographic purposes where p is relatively small. [21] describes an implementation of ECDSA over fields of characteristic 3 and 7. The author in [32] describes a method to implement elliptic curve cryptosystems over fields of *small* odd characteristic, only considering $p < 24$ in the results section. More recently, Boneh and Franklin [8] introduced an identity-based encryption scheme based on the use of the Weil and Tate pairings. Similarly, [7] described a short signature scheme based on the Weil and Tate pairings. Other applications include [17, 34]. All of these applications consider elliptic curves defined over fields of characteristic 2 and 3. Because characteristic 2 field arithmetic has been extensively studied in the literature, authors have concentrated the efforts to improve the performance of systems based on characteristic 3 arithmetic¹. For example, [5, 10] describe algorithms to improve the efficiency of the pairing computations. [10] also introduces some clever tricks to improve the efficiency of the underlying arithmetic in *software* based solutions. [29] treats the hardware implementation of fields of characteristic 3. Their design is only geared towards fields of characteristic 3. Moreover, it is acknowledged that the element representation makes their architectures unsuitable to take advantage of operations such as cubing (i.e. a cubing will only be as fast as a general multiplication, while in other implementations cubing could be more efficient) which are important in Tate pairing computations.

1.1 Our Contributions

Given the research community's interest on cryptographic systems based on fields of odd characteristic and the lack of hardware architectures for general odd characteristic fields, we try to close this gap. Our approach is different from previous ones, in that, we propose *general* architectures which are suitable for fields $GF(p^m)$ with p odd. In particular, we generalize the work in [33] to fields $GF(p^m)$, p odd. We, then, study carefully the case of $GF(3^m)$ due to its cryptographic significance. In addition, we focused on finding irreducible polynomials over $GF(3)$ to improve the performance of the multiplier. For the problem of efficient $GF(p)$ arithmetic, we refer the reader to [9, 30, 13].

The remaining of this contribution is organized as follows. In Section 2 we survey previous $GF(p^m)$ architectures and discuss certain multiplier architectures for $GF(2^k)$ type fields, which we generalize for the $GF(p^m)$ case in Section 4. In

¹ The use of characteristic 3 fields is preferred in some applications due to the improved bandwidth requirements implied by the security parameters. For example, signatures resulting from Pairing cryptography will be smaller in characteristic 3 than in characteristic 2.

Section 4, we also study both the time and area complexities of the multipliers and give two theorems which help us in choosing irreducible polynomials that will minimize the area and delay of the resulting multipliers. Finally, Section 5 specializes the results of the previous section to the case of $GF(3^m)$ fields which recently have become of great interest in the research community. We also describe a prototype implementation of our architectures for three different fields on two FPGAs. We also provide tables of irreducible polynomials over $GF(3)$ for degrees between 2 and 255. We end this contribution with some conclusions.

2 Related Work

In contrast to the $GF(p)$ case, there has not been a lot of work done on $GF(p^m)$ architectures. Our literature search yielded [31] as the only reference that explicitly treated the general case of $GF(p^m)$ multipliers, p odd². In [31], $GF(p^m)$ multiplication is computed in two stages. First the polynomial product is computed modulo a highly factorizable degree S polynomial, $M(x)$, with $S \geq 2m - 1$. The product is, thus, computed using a polynomial residue number system. The second step involves reducing modulo the irreducible polynomial $p(x)$ over which $GF(p^m)$ is defined. The method, however, does not seem to apply to field sizes common in cryptographic applications due to certain constraints on the size of m . In particular, for $p = 3$, there does not exist a suitable $M(x)$ polynomial.

The authors in [25] consider multiplier architectures for composite fields of the form $GF((3^n)^3)$ using Multi-Value Logic (MVL) and a modified version of the Karatsuba algorithm [18] for polynomial multiplication over $GF((3^n)^3)$. Elements of $GF((3^n)^3)$ are represented as polynomials of maximum degree 2 with coefficients in $GF(3^n)$. Multiplication in $GF(3^n)$ is achieved in the obvious way. Karatsuba multiplication is combined with modular reduction over $GF((3^n)^m)$ to reduce the complexity of their design. Because of the use of MVL no discussion of modulo 3 arithmetic is given. The authors estimate the complexity of their design for arithmetic over $GF((3^2)^3)$ as 56 mod-3 adders and 67 mod-3 multipliers. To our knowledge, [29] is the first work that describes $GF(3^m)$ architectures for applications of cryptographic significance, thus we describe it in some detail. The authors describe a representation similar to the one used by [10] to represent their polynomials. They combine all the least significant bits of the coefficients of an element, say A , into one value and all the most significant bits of the coefficients of A into a second value (notice the coefficients of A are elements of $GF(3)$ and thus 2 bits are needed to represent each of them). Thus, $A = (a_1, a_0)$ where a_1 and a_0 are m -bit long each. Addition of two polynomials $A = (a_1, a_0)$, $B = (b_1, b_0)$ with $C = (c_1, c_0) \equiv A + B$ is achieved as:

$$t = (a_1 \vee b_0) \oplus (a_0 \vee b_1), \quad c_1 = (a_0 \vee b_0) \oplus t, \quad c_2 = (a_1 \vee b_1) \oplus t \quad (1)$$

where \vee and \oplus mean the logical OR and exclusive OR operations, respectively. The authors of [29] notice that subtraction and multiplication by 2 are equiva-

² There has been a lot work done, however, on finite field architectures for characteristic two fields.

lent in characteristic 3 and that they can be achieved as $2 \cdot A = 2 \cdot (a_1, a_0) = -A = -(a_1, a_0) = (a_0, a_1)$. Multiplication is achieved in the bit-serial manner, by repeatedly shifting the multiplier down by one bit position and shifting the multiplicand up by one bit position. The multiplicand is then added or subtracted depending on whether the least significant bit of the first or second word of the multiplier is equal to one. They notice that with this representation a cubing operation is only as fast as a general multiply, whereas, using other implementation methods the cubing operation is much faster. Finally, [29] also discuss the implementation of multiplication in $GF((3^m)^6)$ using the irreducible polynomial $Q(y) = y^6 + y + 2$. They use the school book method to multiply polynomials of degree 5 with coefficients in $GF(3^m)$ and then reduce modulo $Q(y)$ using 10 additions and 4 doublings in $GF(3^m)$. They provide timings which we further discuss in Section 6.2.

In [33] a new approach for the design of digit-serial/parallel $GF(2^k)$ multipliers is introduced. Their approach combines both array-type and parallel multiplication algorithms, where the digit-type algorithms minimize the latency for one multiplication at the expense of extra hardware inside each digit cell. In addition, the authors consider special types of polynomials which allow for efficiency in the modulo $p(x)$ reduction operation. These architectures are generalized in Section 4 to the $GF(p^m)$ case, where p is odd.

3 Mathematical Background

For a thorough introduction to finite fields, we refer the reader to [22]. Here, we briefly review the theory that we will need to develop the architectures of this paper. In the following, we will consider the field $GF(p^m)$ generated by an irreducible polynomial $p(x) = x^m + P(x) = x^m + \sum_{i=0}^{m-1} p_i x^i$ over $GF(p)$ of degree m . Let α be a root of $p(x)$, then we can represent $A \in GF(p^m)$ in polynomial basis as $A(\alpha) = \sum_{i=0}^{m-1} a_i \alpha^i$, $a_i \in GF(p)$. Notice that by assumption $p(\alpha) = 0$ since α is a root of $p(x)$. Therefore,

$$\alpha^m = -P(\alpha) = \sum_{i=0}^{m-1} (-p_i) \alpha^i \quad (2)$$

gives an easy way to perform modulo reduction whenever we encounter powers of α greater than $m - 1$.

In what follows, it is assumed that $A, B, C \in GF(p^m)$, with $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, $C = \sum_{i=0}^{m-1} c_i \alpha^i$, and $a_i, b_i, c_i \in GF(p)$. Then, addition in $GF(p^m)$ can be achieved as shown in (3)

$$C(\alpha) \equiv A(\alpha) + B(\alpha) = \sum_{i=0}^{m-1} (a_i + b_i) \alpha^i \quad (3)$$

where the addition $a_i + b_i$ is done in $GF(p)$. We write the multiplication of two elements $A, B \in GF(p^m)$ as $C(\alpha) = \sum_{i=0}^{m-1} c_i \alpha^i \equiv A(\alpha) \cdot B(\alpha)$, where the

multiplication is understood to happen in the finite field $GF(p^m)$ and all α^t , with $t \geq m$ can be reduced with (2). Much of the remaining of this paper is devoted to efficient ways to perform multiplication in hardware. We abuse our notation and throughout the text we will write $A \bmod p(\alpha)$ to mean *explicitly* the reduction step described previously. Finally, we refer to A as the multiplicand and to B as the multiplier.

4 General Architectures for $GF(p^m)$ Arithmetic

This section is concerned with hardware architectures for addition and multiplication in $GF(p^m)$. Inversion can be performed through the Euclidean algorithm or by exponentiation based techniques (see for example [12]) and we do not treat it any further in this paper.

4.1 Adders

Addition in $GF(p^m)$ is performed according to (3). A parallel adder requires m $GF(p)$ adders and its critical path delay is one $GF(p)$ adder.

4.2 Multipliers

There are three different types of architectures used to build $GF(p^m)$ multipliers: array-, digit-, and parallel-multipliers [33]. Array-type (or serial) multipliers process all the coefficients of the multiplicand in parallel in the first step, while the coefficients of the multiplier are processed serially. Array-type multiplication can be performed in two different ways, depending on the order in which the coefficients of the multiplier are processed: Least Significant Element (LSE) first multiplier and Most Significant Element (MSE) first multiplier not described here because of lack of space. Digit-multipliers are also divided in Most Significant and Least Significant Digit-Element first multipliers, depending on the order in which the coefficients of the polynomial are processed. Parallel-multipliers have a high critical path delay but only require one clock cycle to complete a whole multiplication. Thus, parallel-multipliers exhibit high throughput and they are best suited for applications requiring high-speed and relatively small finite fields. However, they are expensive in terms of area when compared to serial multipliers and thus most of the time prohibitive for cryptographic applications. We don't discuss parallel-multipliers any further in this paper.

Least Significant Element (LSE) First Multiplier. The LSE scheme processes first coefficient b_0 of the multiplier and continues with the remaining coefficients one at the time in ascending order. Hence, multiplication according to this scheme can be performed in the following way:

$$C \equiv AB \bmod p(\alpha) \equiv b_0A + b_1(A\alpha \bmod p(\alpha)) + \dots + b_{m-1}(A\alpha^{m-1} \bmod p(\alpha))$$

Algorithm 1 LSE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, $B = \sum_{i=0}^{m-1} b_i \alpha^i$, where $a_i, b_i \in GF(p)$
Ensure: $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in GF(p)$

$$\begin{aligned} C &\leftarrow 0 \\ \text{for } i &= 0 \text{ to } m - 1 \text{ do} \\ &\quad C \leftarrow b_i A + C \\ &\quad A \leftarrow A \alpha \bmod p(\alpha) \\ \text{end for} \\ \text{Return } (C) \end{aligned}$$

The accumulation of the partial product has to be done with a polynomial adder. This multiplier computes the operation according to Algorithm 1. The adapted substructure sharing technique [16] can be used to compute the LSE multiplication in a more efficient way, if the LSE multiplier is used as building block for a larger system, like a system with broadcast structure.

Reduction mod $p(\alpha)$. In LSE multipliers the quantity $W\alpha$, where $W(\alpha) = \sum_{i=0}^{m-1} w_i \alpha^i \in GF(p^m)$, has to be reduced mod $p(\alpha)$. Multiplying W by α ,

$$W\alpha = \sum_{i=0}^{m-1} w_i \alpha^{i+1} = w_{m-1} \alpha^m + \sum_{i=0}^{m-2} w_i \alpha^{i+1}$$

Using (2) and re-writing the index of the second summation, $W\alpha \bmod p(\alpha)$ can then be calculated as follows:

$$W\alpha \bmod p(\alpha) \equiv (-p_0 w_{m-1}) + \sum_{i=1}^{m-1} (w_{i-1} - p_i w_{m-1}) \alpha^i \quad (4)$$

where all coefficient arithmetic is done modulo p . Using (4) we can write expressions for A and C in Algorithm 1 at iteration i as follows:

$$\begin{aligned} C^{(i)} &= \sum_{j=0}^{m-1} c_j^{(i)} \alpha^j \equiv b_i A^{(i)} + C^{(i-1)} = \sum_{j=0}^{m-1} (b_i a_j^{(i)} + c_j^{(i-1)}) \alpha^j, \\ A^{(i)} &= \sum_{j=0}^{m-1} a_j^{(i)} \alpha^j \equiv A^{(i-1)} \alpha \equiv (-p_0 a_{m-1}^{(i-1)}) + \sum_{j=1}^{m-1} (a_{j-1}^{(i-1)} - p_j a_{m-1}^{(i-1)}) \alpha^j \end{aligned}$$

with $C^{(-1)} = 0$ and $A^{(-1)} = A$. As a final remark, notice that if you initialize C to a value different from 0, say I , then Algorithm 1 computes $C \equiv A \cdot B + I \bmod p(\alpha)$. This multiply-accumulate operation turns out to be very useful in elliptic curve systems and it is obtained at no extra cost.

Table 1. Area complexity and critical path delay of LSE multiplier

Irreducible polynomial	Area Complexity	Critical Path Delay	Latency (# clocks)
r -nomial	$(m + r - 2)$ ADD + $(m + r - 1)$ MUL	1 ADD + 1 MUL	m
General	$(2m - 1)$ ADD + $2m$ MUL	1 ADD + 1 MUL	m

Area/Time Complexity of LSE Multipliers. LSE multipliers take m iterations to output the product $C \equiv A \cdot B \bmod p(\alpha)$. In each iteration, the following operations are performed: 1 multiplication of a $GF(p)$ element by a $GF(p^m)$ element (requires m $GF(p)$ multipliers), 1 $GF(p^m)$ addition (requires m $GF(p)$ adders), 1 multiplication by α (implemented as a $GF(p)$ coefficient shift), and 1 modulo $p(\alpha)$ reduction. This last operation could be implemented according to (4), and thus, it would require $(r - 1)$ $GF(p)$ multipliers and $(r - 2)$ adders for a fixed r -nomial (where r is the number of non-zero coefficients in the irreducible polynomial $p(x)$). However, it could be desired to load the modulus on demand, in which case one would need m $GF(p)$ multipliers and $(m - 1)$ adders.

Both the area and time complexities of Algorithm 1 are summarized in Table 1 in terms of $GF(p)$ adders and multipliers, for two types of irreducible polynomials. This unusual measure is independent of technology and thus most general. One immediate advantage of estimating area in terms of $GF(p)$ adders (multipliers) is that we don't need to care about the way these are implemented. In particular, there are many implementation choices depending on your application and design criteria [9, 13, 30]. Section 6.1 gives specific complexity numbers for an FPGA implementation of $GF(3^m)$ arithmetic in terms of both Look-Up Tables³ (LUTs), also known as Configurable Logic Blocks (CLBs), and flip-flops (FF), thus taking into account the way $GF(p)$ arithmetic is implemented on the target technology. In Table 1, ADD and MUL refer to the area and delay of a $GF(p)$ adder and multiplier, respectively. We have not taken into account the delays or area requirements of storage elements (such as those needed to implement a shift register) or routing elements (such as those used for interconnections in FPGAs). In addition, we do not make any distinction between general and constant $GF(p)$ multipliers, i.e., we assume their complexities are the same. Finally, general irreducible polynomials refer to the case in which you want to be able to change the irreducible polynomial on demand.

Digit-Srial/Parallel Multipliers. LSE multipliers process the coefficients of A in parallel, while the coefficients of B are processed serially. Hence, these multipliers are area-efficient and suitable for low-speed applications. Digit multipliers, introduced in [33] for fields $GF(2^k)$, are a trade-off between speed, area, and power consumption. This is achieved by processing several of B 's coefficients at the same time. The number of coefficients that are processed in parallel is defined to be the digit-size and we denote it by D . For a digit-size D , we can denote

³ Look-Up Tables are the basic building blocks of most common FPGAs [1, 2, 35].

by $d = \lceil m/D \rceil$ the total number of digits in a polynomial of degree $m - 1$. Then, we can re-write the multiplier as $B = \sum_{i=0}^{d-1} B_i \alpha^{D_i}$, where

$$B_i = \sum_{j=0}^{D-1} b_{Di+j} \alpha^j \quad 0 \leq i \leq d-1 \quad (5)$$

and we assume that B has been padded with zero coefficients such that $b_i = 0$ for $m - 1 < i < d \cdot D$ (i.e. B 's size is $d \cdot D$ coefficients but $\deg(B) < m$). Hence,

$$C \equiv AB \bmod p(\alpha) = A \sum_{i=0}^{d-1} B_i \alpha^{D_i} \bmod p(\alpha) \quad (6)$$

In the following, a generalized digit-serial/parallel multiplication algorithm is introduced. We named this algorithm Least Significant Digit-Element first multiplier (LSDE), where the word *element* was introduced to clarify that the digits correspond to groups of $GF(p)$ coefficients in contrast to [33] where the digits were groups of bits. The LSDE is an extension of the LSE multiplier. Using (6), the product $C \equiv AB \bmod p(\alpha)$ in this scheme can be calculated as follows

$$C \equiv [B_0 A + B_1 (A \alpha^D \bmod p(\alpha)) + \dots + B_{d-1} (A \alpha^{D(d-2)} \alpha^D \bmod p(\alpha))] \bmod p(\alpha)$$

This is summarized in Algorithm 2. The adapted substructure sharing technique

Algorithm 2 LSDE Multiplier

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, where $a_i \in GF(p)$, $B = \sum_{i=0}^{\lceil \frac{m}{D} \rceil - 1} B_i \alpha^{D_i}$, where B_i is as defined in (5)
Ensure: : $C \equiv A \cdot B = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in GF(p)$

```

 $C \leftarrow 0$ 
 $\text{for } i = 0 \text{ to } \lceil \frac{m}{D} \rceil - 1 \text{ do}$ 
     $C \leftarrow B_i A + C$ 
     $A \leftarrow A \alpha^D \bmod p(\alpha)$ 
 $\text{end for}$ 
Return  $(C \bmod p(\alpha))$ 
```

can also be used for the LSDE, like in the case of the LSE multiplier. We end this section by noticing that, as in Algorithm 1, if C is initialized to I in Algorithm 2, we can obtain as an output $A \cdot B + I \bmod p(\alpha)$ at no additional (hardware or delay) cost. This operation, known as a multiply/accumulate operation, is very useful in elliptic curve based systems.

Reduction mod $p(\alpha)$ for Digit Multipliers. In LSDE multipliers the product $W \alpha^D \bmod p(\alpha)$ occurs. As in the LSE multiplier case, one can derive equations for the modular reduction for *particular* irreducible $p(\alpha)$ polynomials. However, it is more interesting to search for polynomials that minimize the complexity of the reduction operation. In coming up with these optimum irreducible

polynomials we use two theorems from [33], adapted to the case of $GF(p^m)$ fields with p odd.

Theorem 1. Assume that $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{j=0}^{k-1} p_j\alpha^j$, with $k < m$. For $t \leq m - 1 - k$, the degree of α^{m+t} can be reduced to be less than m in one step with the following equation:

$$\alpha^{m+t} \bmod p(\alpha) = -p_k\alpha^{k+t} - \left(\sum_{j=0}^{k-1} p_j\alpha^{j+t} \right) \quad (7)$$

Theorem 2. For Digit multipliers with digit-element size D , when $D \leq m - k$ the degree of the intermediate results in Algorithm 2 can be reduced to be less than m in one step.

Theorems 1 and 2, implicitly say that for a given irreducible polynomial $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{j=0}^{k-1} p_j\alpha^j$, the digit-element size will depend on the value of k .

Area/Time Complexity of LSDE Multipliers. Before estimating the complexity of the LSDE multiplier, it is helpful to obtain equations to describe the values of A and C at iteration i in Algorithm 2. Thus, assume that B_i is as in (5), $p(\alpha)$ as in Theorem 1, $A = \sum_{i=0}^{m-1} a_i\alpha^i$, and $D \leq m - k$ (Theorem 2). Then,

$$C^{(i)} = D^{(i)} + C^{(i-1)} = \sum_{j=0}^{m+D-2} \left(d_j^{(i)} + c_j^{(i-1)} \right) \alpha^j \quad (8)$$

$$A^{(i)} = \sum_{j=D}^{m-1} a_{j-D}^{(i-1)} \alpha^j + \sum_{s=0}^k \sum_{j=0}^{D-1} \left(-p_s \cdot a_{j+m-D}^{(i-1)} \right) \alpha^{j+s} \quad (9)$$

where $C^{(-1)} = 0$, $A^{(-1)} = A$, and

$$D^{(i)} = \sum_{j=0}^{m+D-2} d_j^{(i)} \alpha^j = B_i \cdot A^{(i-1)} = \sum_{j=0}^{m-1} \sum_{k=0}^{D-1} \left(a_j^{(i-1)} \cdot b_{D(i+k)} \right) \alpha^{j+k} \quad (10)$$

Now it is easy to see that in each iteration one requires mD multipliers in parallel and $\sum_{j=0}^{D-2} j + \sum_{j=D-1}^{m-1} (D-1) + \sum_{j=m}^{m+D-2} (m+D-2-j) = (D-1)(m-1)$ adders. Therefore, according to (8), we only need $(D-1)(m-1) + m + D - 1 = mD$ adders to compute $C^{(i)}$. Using a ripple adder architecture, the critical path is given by $D-1$ adder delays from the computation of $D^{(i)}$, one adder delay from the computation $d_j^{(i)} + c_j^{(i-1)}$ in (8), and one multiplier. We notice, however, that it is possible to improve the critical path delay of the LSDE multiplier by using a binary tree of adders⁴. Using this technique one would reduce the length of the critical path from D $GF(p)$ adders and one $GF(p)$ multiplier to $\lceil \log_2(D+1) \rceil$ adders and one multiplier. We use this, as our complexity for the critical path.

⁴ Binary trees have been used both in [33] and [26] in the context of $GF(2^n)$ arithmetic to reduce delay and power consumption.

Table 2. Area complexity and critical path delay of LSDE multiplier

Irreducible polynomial	Area Complexity	Critical Path Delay	Latency (# clocks)
r -nomial	$(m+r-2)D$ ADD + $(m+r-1)D$ MUL	$\lceil \log_2(D+1) \rceil$ ADD + 1 MUL	$\lceil \frac{m}{D} \rceil$
General	$(m+k)D$ ADD + $(m+k+1)D$ MUL	$\lceil \log_2(D+1) \rceil$ ADD + 1 MUL	$\lceil \frac{m}{D} \rceil$

The computation of (9) requires only $D(k+1)$ multipliers (notice that the second term of (9) looks exactly the same as $D^{(i)}$ in (10), except that the limits in the summation are changed) and at most Dk adders. In the case in which $p(\alpha)$ is an r -nomial, the complexities reduce to $(r-1)D$ multipliers and at most $(r-2)D$ adders (notice that the first summation in (9) starts at $j = D$, thus, the first D coefficients resulting from the second summation do not need to be added to anything). We say *at most* because depending on the values of D and k , some adders may be saved. These results are summarized in Table 2

Table 2 makes the same assumptions as for the LSE case, i.e., ADD and MUL refer to the area and delay of a $GF(p)$ adder and multiplier, respectively, delay or area of storage elements are not taken into account, and no distinction is made between general and constant $GF(p)$ multipliers. We end by noticing that an LSDE multiplier with $D = 1$ is equivalent to an LSE multiplier. Our complexity estimates verify this if you let $D = 1$ and $k = m - 1$ in Table 2.

4.3 Comments on Irreducible Polynomials for $GF(p^m)$

From Theorems 1 and 2, it is obvious that choosing an irreducible polynomial should be carefully done. For fields $GF(p^m)$ with odd prime characteristic it is often possible to choose irreducible binomials $p(\alpha) = x^m - \omega$, $\omega \in GF(p)$. This is particularly interesting since binomials are never irreducible in characteristic 2 fields. Another interesting property of binomials is that they are optimum from the point of view of Theorem 1. In particular for any irreducible binomial $p(\alpha) = x^m - \omega$, $k = 0$ and $D \leq m$ in Theorem 2, which means that even in the degenerate case where $D = m$ (i.e. a parallel multiplier) one is able to perform the reduction in one step. In addition, reduction is virtually for free, corresponding to just a few $GF(p)$ multiplications (this follows from the fact that $\alpha^m = \omega$). A specific sub-class of these fields where q is a prime of the form $q = p = 2^n - c$, c “small”, has recently been proposed for cryptographic applications in [4]. We notice that the existence of irreducible binomials has been completely established as Theorem 3 shows⁵.

Theorem 3 ([22]). Let $m \geq 2$ be an integer and $\omega \in F_q^*$. Then the binomial $x^m - \omega$ is irreducible in $F_q[x]$ if and only if the following two conditions

⁵ Reference [22] is used here as a convenient reference for well established results.

are satisfied: (i) each prime factor of m divides the order e of ω in F_q^* , but not $(q - 1)/e$; (ii) $q \equiv 1 \pmod{4}$ if $m \equiv 0 \pmod{4}$.

When irreducible binomials can not be found, one searches in incremental order for irreducible trinomials, quadrinomials, etc. In [15] von zur Gathen and Nöcker conjecture that the minimal number of terms $\sigma_q(m)$ in irreducible polynomials of degree m in $GF(q)$, q a power of a prime, is for all $m \geq 1$, $\sigma_2(m) \leq 5$ and $\sigma_q(m) \leq 4$ for $q \geq 3$. This conjecture has been verified for $q = 2$ and $m \leq 10000$ [6, 11, 15, 36, 37, 38] and for $q = 3$ and $m \leq 539$ [14].

By choosing irreducible polynomials with the least number of non-zero coefficients, one can reduce the area complexity of the LSDE multiplier (this follows directly from Table 2). We point out that by choosing irreducible polynomials such that their non-zero coefficients are all equal to $p - 1$ one can further reduce the complexity since all the multiplications by $-p_s$ in (9) reduce to multiplication by 1. Notice that there is no existence criteria for irreducibility of trinomials over any field $GF(p^m)$. The most recent advances in this area are the results of Loidreau [23], where a table that characterizes the parity of the number of factors in the factorization of a trinomial over $GF(3)$ is given, and the necessary (but not sufficient) irreducibility criteria for trinomials introduced by von zur Garten in [14]. Neither reference provides tables of irreducible polynomials.

5 Case Study: $GF(3^m)$ Arithmetic

FPGAs are reconfigurable hardware devices whose basic logic elements are Look-Up Tables (LUTs), also called Configurable Logic Blocks (CLBs), flip-flops (FFs), and, for modern devices, memory elements [1, 2, 35]. The LUTs are used to implement Boolean functions of their inputs, that is, functions traditionally implemented with logic gates. In the particular case of the XCV1000E-8-FG1156 and the XC2VP20-7-FF1156, their basic building blocks are 4-input bits/1-output bit LUTs. This means that all basic arithmetic operations in $GF(3)$ (add, subtract, and multiply) can be done with 2 LUTs, where each LUT generates one bit of the output.

5.1 Cubing in $GF(3^m)$

It is well known that for $A \in GF(p^m)$ the computation of A^p is linear. In the particular case of $p = 3$, we can write the frobenius map as:

$$A^3 \equiv \left(\sum_{i=0}^{m-1} a_i \alpha^i \right)^3 \pmod{p(\alpha)} = \sum_{i=0}^{m-1} a_i \alpha^{3i} \pmod{p(\alpha)} \quad (11)$$

Equation (11) can in turn be written as the sum of three terms (where we have re-written the indices in the summation):

$$A^3 \equiv \sum_{\substack{i=0 \\ i \equiv 0 \pmod{3}}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \pmod{p(\alpha)} \equiv T + U + V \pmod{p(\alpha)} \quad (12)$$

$$\equiv \left(\sum_{\substack{i=0 \\ i \equiv 0 \pmod{3}}}^{m-1} a_{\frac{i}{3}} \alpha^i \right) + \left(\sum_{\substack{i=m \\ i \equiv 0 \pmod{3}}}^{2m-1} a_{\frac{i}{3}} \alpha^i \right) + \left(\sum_{\substack{i=2m \\ i \equiv 0 \pmod{3}}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \right) \bmod p(\alpha)$$

Notice that only U and V need to be reduced mod $p(\alpha)$. We further assume that $p(\alpha) = x^m + p_t x^t + p_0$ with $t < m/3$. This assumption is valid in terms of the existence of such irreducible trinomials as shown in Section 5.2. Thus,

$$U = \sum_{\substack{i=m \\ i \equiv 0 \pmod{3}}}^{2m-1} a_{\frac{i}{3}} \alpha^i \bmod p(\alpha) = \sum_{\substack{i=m \\ i \equiv 0 \pmod{3}}}^{2m-1} a_{\frac{i}{3}} \alpha^{i-m} (-p_t \alpha^t - p_0) \bmod p(\alpha)$$

$$V = \sum_{\substack{i=2m \\ i \equiv 0 \pmod{3}}}^{3(m-1)} a_{\frac{i}{3}} \alpha^i \bmod p(\alpha) = \sum_{\substack{i=2m \\ i \equiv 0 \pmod{3}}}^{3(m-1)} a_{\frac{i}{3}} \alpha^{i-2m} (\alpha^{2t} - p_t p_0 \alpha^t + 1) \bmod p(\alpha)$$

where we have made use of the fact that $(-p_t \alpha^t - p_0)^2 = (\alpha^{2t} - p_t p_0 \alpha^t + 1)$ in $GF(3)$. It can be shown that U and V can be reduced to be of degree less than m in one extra reduction step. To estimate the complexity of this cubing circuit, we assume that $p(\alpha)$ is a fixed irreducible trinomial with $t < m/3$ i.e., that multiplications in $GF(3)$ (for example multiplying by $-p_t p_0$) can be handled by adders and subtracters. Then, it can be shown that one needs in the order of $2m$ adders/subtracters to perform a cubic operation in $GF(3^m)$.

5.2 Irreducible Polynomials over $GF(3)$

Following the criteria of Section 4.3 for choosing irreducible polynomials, we tried to find irreducible binomials first. Unfortunately, the only irreducible binomial over $GF(3)$ is $x^2 + 1$, thus we considered irreducible trinomials. Notice that $x^m + x^t + 1$ is never irreducible over $GF(3)$ since 1 is always a root of it. Thus, we only searched for irreducible trinomials of the following forms: $x^m - x^t - 1$ or $x^m \pm x^t \mp 1$. For $2 \leq m \leq 255$, we exhaustively searched for these trinomials (see Tables 6, 7, and 8 in Appendix A). There are only 23 degrees m in the range above for which we were unable to find trinomials (which agrees with the findings in [14]) and thus quadrinomials can be found. Of these quadrinomials only 4 correspond to m prime (149, 197, 223, 233). Prime m is the most commonly used degree in cryptographic applications. We notice that of the 50 primes in the above range which had trinomials, we were not able to find trinomials with $t < m/3$ for 9 of them (18 %).

6 $GF(3^m)$ Prototype Implementation and Comparisons

Figure 1 shows a block diagram of the prototyped arithmetic unit (AU). Notice that in Figure 1, all bus-widths correspond to how many $GF(3)$ elements can be carried by the bus, i.e., if we write m , then it is understood that the bus is $2m$ bits wide. The AU consists of an LSDE multiplier and a cubing circuit.

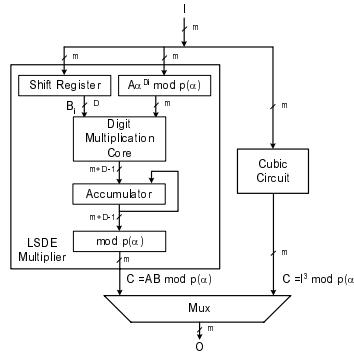


Fig. 1. $GF(3^m)$ arithmetic unit architecture

The multiplier and the cubing circuit support the computation of field additions, squares, multiplications, and inversions. For addition and subtraction we take advantage of the multiply/accumulate capabilities of the LSDE multiplier and cubing circuit. In other words, the addition $C = A + B$ is done by first computing $A \cdot 1$ and then adding to it the product $B \cdot 1$. This takes two clock cycles. However, if operand A is already in the accumulator of the multiplier one can compute $C = B \cdot 1 + A$ in one clock. This eliminates the need for an adder. Subtractions are computed in a similar manner, i.e., $C = A - B$ is done by first computing $A \cdot 1$ and then adding to it the product $(-1) \cdot B$ or alternatively as $C = (-1) \cdot B + A$.

AU prototypes were developed to verify the suitability of the architecture shown in Figure 1 for reconfigurable FPGA logic and compare the efficiency of $GF(3^m)$ and $GF(2^m)$ AUs. The prototypes were coded in VHDL at a very low level. The VHDL code was synthesized using Synopsis FPGA Compiler 3.7.1 and the component placement and routing was done using Xilinx Design Manager 4.2.03i. The prototypes were synthesized and routed for the Xilinx XCV1000-8-FG1156 and XC2VP20-7-FF1156 FPGAs. The XCV1000E-8-FG1156 prototype allowed us to compare our AU implementations against the AU for $GF(2^m)$ used in the EC processor (ECP) from [27], which is one of the fastest ECP implemented in FPGA logic for EC defined over fields $GF(2^{167})$. The XC2VP20-7-FF1156 prototype allowed us to verify the speed of our AU for one of the newest families of Xilinx FPGAs. Three implementation were developed which support the fields $GF(3^{97})$, $GF(2^{151})$, and $GF(2^{241})$. The fields $GF(3^{97})$ and $GF(2^{241})$ are used in Weil and Tate pairing schemes for systems with comparable degrees of security (see [10, 5, 29]). The field $GF(2^{151})$ offers security comparable to that of $GF(3^{97})$ for cryptosystems based on the EC discrete logarithm problem.

6.1 $GF(3^m)$ Complexity Estimates

Table 3 shows the complexity estimates for the AU shown in Figure 1. The estimates assume the use of optimum irreducible polynomials and give the register complexity in terms of the number of flip-flops. Note that the register estimates

Table 3. $GF(3^m)$ AU complexity estimates

Circuit	Area Complexity
LSE Mult.	$(4mD + 2m + 6D)$ LUT + $6m + 2D + 4$ FF
Cubic	$4m$ LUT
Mux	$2m$ LUT
AU (total)	$(4mD + 8m + 6D)$ LUT + $6m + 2D + 4$ FF

do not account for registers used to reduce the critical path delay of a multiplier, a technique known as pipelining. This technique was used to reduce the critical path delay of the prototype implementations. The complexity estimates are based on the following assumptions:

1. $GF(3)$ adders, subtracters, or multipliers require two LUTs, including adders that add weighted inputs, for example, adders that compute $(a_i * c) + (b_i * d)$ where c and d are fixed constants. Also a 2:1 multiplexer requires one LUT.
2. From Table 2 the digit multiplication core and accumulator circuits require mD $GF(3)$ multipliers and mD $GF(3)$ adders. This circuit stores the result in two $(m+D-1)$ -bit registers. An m -bit register requires m flip-flops (FFs).
3. The estimates for the $A\alpha^{D_i} \bmod p(\alpha)$ circuit assume that the circuit contains two m -bit multiplexers that select between the element A and the element $A\alpha^{D_i} \bmod p(\alpha)$. An m -bit multiplexer requires m LUTs. For programmable optimum irreducible trinomials, the circuitry that generates $A\alpha^{D_i} \bmod p(\alpha)$ requires $2D$ $GF(3)$ multipliers and D adders (see Table 2). This circuit stores the result in two m -bit registers and the coefficients of $p(x)$ in two $2r$ -bit registers ($r = 3$ for trinomials).
4. The coefficients of B are fed in by two m -bit parallel in/serial out shift registers. Each shift registers contains m 2:1 multiplexers and m registers.
5. The cubic circuit requires $2m$ $GF(3)$ adders.
6. The complexity for the $GF(2^m)$ AU is done according to [26]. It is also assumed that the $GF(2^m)$ AU contains an LSD multiplier and a squarer.

Table 4. AU estimated vs. measured complexity for prototypes ($D = 16$)

Circuit	Estimated complexity	Measured complexity (incl. pipelining registers & I/O)	LUT Estimate Error (measured/est.)
$GF(2^{151})$	2366 LUT + 453 FF ($15.7m$ LUT + $3m$ FF)	2239 LUT + 1092 FF ($14.3m$ LUT + $7.2m$ FF)	-5.4 %
$GF(2^{241})$	3705 LUT + 723 FF ($14.9m$ LUT + $3m$ FF)	3591 LUT + 1722 FF ($15.4m$ LUT + $7.1m$ FF)	-3.1 %
$GF(3^{97})$	7080 LUT + 618 FF ($73.0m$ LUT + $6.4m$ FF)	7122 LUT + 2790 FF ($73.4m$ LUT + $7.2m$ FF)	1.0 %

The estimates that we obtain from our models are very accurate when compared to the actual measured complexities. This validates our models and assumptions.

6.2 Results

Table 5 presents the timings obtained for our three prototypes. We have tried to implement our designs in such a way that we can make a meaningful comparison. Thus, although, the clock rates are not exactly the same between the different designs (this is due to the fact that the clock rate depends on the critical path of the AU which is different for each circuit), they are not more than 10 % different. The platforms are the same and we chose same digit sizes for both $GF(2^m)$ and $GF(3^m)$ architectures. The results make sense, for the same digit size ($D = 16$) we obtain that the $GF(3^{97})$ design is about twice as big as the $GF(2^{241})$ design and more than 3 times the size of the $GF(2^{151})$ AU. This is offset by the gain in performance. At similar clock rates the $GF(3^{97})$ design is 2.7 times faster than the corresponding $GF(2^{241})$ AU and 1.4 times faster than the $GF(2^{151})$ one. It is clear that by using more hardware resources for $GF(3^{97})$ we achieve better performance than characteristic two fields. In particular, by choosing the same digit size for both fields, we implicitly process twice as many bits of the multiplier in $GF(3^{97})$ as in the $GF(2^m)$ case (remember that the E in LSDE refers to elements of $GF(p)$ and not bits as in the $GF(2^m)$ case). Table 5 also includes the results from [29]. We don't think it is possible to make a meaningful comparison, other than point out that by coding directly in VHDL, one can achieve huge improvements in performance of FPGA based implementations.

7 Conclusions

In this paper, we have generalized the finite field multipliers of [33] to the odd characteristic case. We have also presented multiplication algorithms for both serial and digit-based architectures. Finally, we have presented a general framework to choose irreducible polynomials that reduce the computation of the modulo reduction operation. More importantly, we have shown that it is possible to achieve considerable performance from FPGA implementations of non-binary finite fields. However, from our discussion in the previous section, we conclude that fields of characteristic 2 can not be surpassed by other fields if one considers both area and time performance measures.

Table 5. Comparison of multiplication time for $GF(2^{151})$, $GF(2^{241})$, and $GF(3^{97})$ prototypes ($D = 16$) and the AU from [29]

Circuit	Time for optimized mult. [29] [29](in usecs)	Mult. time for prototypes	
		XCV1000-8-FG1156 (in usecs)	XC2VP20-7-FF1156 (in usecs)
$GF(2^{151})$	N/A	0.139 (@ 71.7 MHz)	0.100 (@ 100.2 MHz)
$GF(2^{241})$	37.32 (@ 20 MHz)	0.261 (@ 61.3 MHz)	0.150 (@ 107 MHz)
$GF(3^{97})$	50.68 (@ 20 MHz)	0.097 (@ 72 MHz)	0.074 (@ 94.4 MHz)

References

- [1] Actel Corporation. *Actel's ProASIC Family, The Only ASIC Design Flow FPGA*, 2001. [164](#), [168](#)
- [2] Altera Corporation. *APEX 20KC Programmable Logic Device Data Sheet*, 2001. [164](#), [168](#)
- [3] D. V. Bailey and C. Paar. Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, volume LNCS 1462, pages 472–485, Berlin, Germany, 1998. Springer-Verlag. [159](#)
- [4] D. V. Bailey and C. Paar. Efficient arithmetic in finite field extensions with application in elliptic curve cryptography. *Journal of Cryptology*, 14(3):153–176, 2001. [159](#), [167](#)
- [5] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In M. Yung, editor, *Advances in Cryptology — CRYPTO 2002*, volume LNCS 2442, pages 354–368. Springer-Verlag, 2002. [159](#), [170](#)
- [6] Blake, Gao, and Lambert. Constructive problems for irreducible polynomials over finite fields. In *Information Theory and Applications*, pages 1–23. Springer-Verlag, 1993. [168](#)
- [7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In J. Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume LNCS 2148, pages 514–532. Springer-Verlag, 2001. [159](#)
- [8] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In J. Kilian, editor, *Advances in Cryptology — CRYPTO 2001*, volume LNCS 2139, pages 213–229. Springer-Verlag, 2001. [159](#)
- [9] E. D. Di Claudio, F. Piazza, and G. Orlandi. Fast Combinatorial RNS Processors for DSP Applications. *IEEE Transactions on Computers*, 44(5):624–633, May 1995. [159](#), [164](#)
- [10] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate Pairing. In C. Fieker and D. Kohel, editors, *Algorithmic Number Theory — ANTS-V*, volume LNCS 2369, pages 324–337. Springer-Verlag, 2002. [159](#), [160](#), [170](#)
- [11] S. W. Golomb. *Shift Register Sequences*. Holden-Day, San Francisco, USA, 1967. [168](#)
- [12] J. Guajardo and C. Paar. Itoh-Tsuji Inversion in Standard Basis and Its Application in Cryptography and Codes. *Design, Codes, and Cryptography*, 25(2):207–216, February 2002. [162](#)
- [13] J. Guajardo, T. Wollinger, and C. Paar. Area Efficient $GF(p)$ Architectures for $GF(p^m)$ Multipliers. In *Proceedings of the 45th IEEE International Midwest Symposium on Circuits and Systems — MWSCAS 2002*, August 2002. [159](#), [164](#)
- [14] J. von zur Gathen. Irreducible Trinomials over Finite Fields. In B. Mourrain, editor, *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation — ISSAC2001*, pages 332–336. ACM Press, 2001. [168](#), [169](#)
- [15] J. von zur Gathen and M. Nöcker. Exponentiation in Finite Fields: Theory and Practice. In T. Mora and H. Mattson, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes — AAECC-12*, volume LNCS 1255, pages 88–113. Springer-Verlag, 2000. [168](#)
- [16] S. K. Jain and K. K. Parhi. Efficient standard basis reed-solomon encoder. In *1996 IEEE International Conference of Acoustics, Speech, and Signal Processing*, Atlanta, May 1996. [163](#)

- [17] A. Joux. A one-round protocol for tripartite Diffie-Hellman. In W. Bosma, editor, *Algorithmic Number Theory — ANTS-IV*, volume LNCS 1838, pages 385–394. Springer-Verlag, 2000. [159](#)
- [18] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Sov. Phys. Dokl. (English translation)*, 7(7):595–596, 1963. [160](#)
- [19] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987. [158](#)
- [20] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):129–150, 1989. [158](#)
- [21] N. Koblitz. An elliptic curve implementation of the finite field digital signature algorithm. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO 98*, volume LNCS 1462, pages 327–337. Springer-Verlag, 1998. [159](#)
- [22] R. Lidl and H. Niederreiter. *Finite Fields*, volume 20 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, Reading, Massachusetts, USA, 1983. [161](#), [167](#)
- [23] P. Loidreau. On the Factorization of Trinomials over F_3 . Rapport de recherche no. 3918, INRIA, April 2000. [168](#)
- [24] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag. [158](#)
- [25] Jin Young Oo, Young-Gern Kim, Dong-Young Park, and Heung-Su Kim. Efficient Multiplier Architecture Using Optimized Irreducible Polynomial over $GF((3^n)^3)$. In *Proceedings of the IEEE Region 10 Conference – TENCON 99. "Multimedia Technology for Asia-Pacific Information Infrastructure"*, volume 1, pages 383–386, 1999. [160](#)
- [26] G. Orlando. *Efficient Elliptic Curve Processor Architectures for Field Programmable Logic*. PhD thesis, Dept. of ECE, Worcester Polytechnic Institute, March 2002. [166](#), [171](#)
- [27] G. Orlando and C. Paar. A High-Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume LNCS 1965, pages 41–56. Springer-Verlag, 2000. [170](#)
- [28] P. Mihăilescu. Optimal Galois Field Bases which are not Normal. Recent Results Session — FSE '97, 1997. [159](#)
- [29] D. Page and N. P. Smart. Hardware implementation of finite fields of characteristic three. In B. S. Kaliski, Jr., Ç. K. Koç, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2002*, volume LNCS. Springer-Verlag, 2002. [159](#), [160](#), [161](#), [170](#), [172](#)
- [30] V. Palioras, K. Karagianni, and T. Stouraitis. A Low-Complexity Combinatorial RNS Multiplier. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(7):675–683, July 2001. [159](#), [164](#)
- [31] M. G. Parker and M. Benissa. $GF(p^m)$ Multiplication Using Polynomial Residue Number Systems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 42(11):718–721, November 1995. [160](#)
- [32] N. Smart. Elliptic Curve Cryptosystems over Small Fields of Odd Characteristic. *Journal of Cryptology*, 12(2):141–151, Spring 1999. [159](#)
- [33] L. Song and K. K. Parhi. Low energy digit-serial/parallel finite field multipliers. *Journal of VLSI Signal Processing*, 19(2):149–166, June 1998. [159](#), [161](#), [162](#), [164](#), [165](#), [166](#), [172](#)
- [34] E. Verheul. Self-blindable Credential Certificates from the Weil Pairing. In C. Boyd, editor, *Advances in Cryptology — Asiacrypt 2001*, volume LNCS 2248, pages 533–551. Springer-Verlag, 2001. [159](#)

- [35] Xilinx, Inc. *The Programmable Logic Data Book*, 2000. 164, 168
 [36] N. Zierler. On $x^n + x + 1$ over $GF(2)$. *Information and Control*, 16:67–69, 1970. 168
 [37] N. Zierler and J. Brillhart. On Primitive Trinomials (mod2). *Information and Control*, 13:541–554, 1968. 168
 [38] N. Zierler and J. Brillhart. On Primitive Trinomials (mod 2), II. *Information and Control*, 14:566–569, 1969. 168

A Lists of Irreducible Polynomials over $GF(3)$

Table 6. Irreducible trinomials of the form $x^m + x^t + 2$ over $GF(3)$, $2 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t		
2	1	22	5	43	26	71	20	93	70	117	52	143	108	167	92	187	8	211	122	238	5
3	2	23	8	44	3	72	28	95	48	119	2	144	56	168	28	188	11	214	65	239	5
4	1	24	4	45	28	73	30	96	16	120	4	145	24	169	24	191	116	215	36	240	8
4	2	25	6	46	5	76	9	97	12	121	40	147	8	170	43	192	32	216	4	241	88
6	1	26	7	47	32	77	16	99	74	124	25	148	3	171	20	193	12	217	132	242	115
7	2	27	20	48	8	78	13	100	25	125	52	150	73	172	19	194	55	219	26	243	122
8	2	28	2	51	50	79	26	101	70	126	49	151	2	173	166	195	26	220	15	244	31
9	4	29	4	52	7	80	2	102	25	127	8	152	18	174	73	196	79	222	89	245	148
11	2	30	1	53	22	81	40	103	50	128	6	153	94	176	12	198	29	224	12	246	13
12	2	31	20	54	1	83	32	104	5	131	48	155	12	177	52	199	164	225	16	247	122
13	4	32	5	55	26	84	14	107	32	133	88	156	26	178	11	200	3	227	68	248	50
14	1	33	28	56	3	85	16	108	2	134	61	157	22	179	104	201	88	228	14	249	76
15	2	35	2	59	20	86	13	109	88	135	44	158	61	180	38	203	8	229	72	251	26
16	4	36	14	60	2	87	26	111	2	136	57	159	32	181	40	204	50	230	73	252	98
17	16	37	6	61	30	88	6	112	6	137	136	160	4	182	25	205	78	232	30	253	12
18	7	39	26	63	26	89	64	113	70	139	80	162	19	183	2	206	61	234	91	254	73
19	2	40	1	64	3	90	19	114	7	140	59	163	80	184	20	208	10	235	26	255	26
20	5	41	40	67	2	91	74	115	32	141	64	164	15	185	64	209	40	236	9		
21	16	42	7	69	52	92	10	116	15	142	65	165	22	186	47	210	7	237	70		

Table 7. Irreducible trinomials of the form $x^m + 2x^t + 1$ over $GF(3)$, $3 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t		
3	1	25	3	46	6	71	20	94	30	119	2	145	24	170	32	194	24	222	4	249	59
5	1	26	2	47	15	73	1	95	47	121	1	146	2	171	20	195	26	225	16	250	104
6	2	27	7	50	6	74	12	97	12	122	14	147	8	173	7	198	38	226	38	251	9
7	2	29	4	51	1	77	16	99	19	125	52	151	2	174	52	199	35	227	11	253	7
9	4	30	4	53	13	78	14	101	31	126	52	153	59	177	52	201	88	229	72	254	16
10	2	31	5	54	14	79	26	102	2	127	8	154	32	178	26	202	62	230	64	255	26
11	2	33	5	55	11	81	40	103	47	131	27	155	12	179	59	203	3	234	104		
13	1	34	2	58	8	82	2	106	26	133	15	157	22	181	37	205	9	235	26		
14	4	35	2	59	17	83	27	107	3	134	4	158	52	182	34	206	94	237	70		
15	2	37	6	61	7	85	16	109	9	135	44	159	32	183	2	209	40	238	4		
17	1	38	4	62	10	86	34	110	22	137	1	162	80	185	64	211	89	239	5		
18	8	39	7	63	26	87	26	111	2	138	34	163	59	186	46	214	6	241	88		
19	2	41	1	66	10	89	13	113	19	139	59	165	22	187	8	215	36	242	2		
21	5	42	10	67	2	90	34	115	32	141	5	166	54	190	94	217	85	243	121		
22	4	43	17	69	17	91	17	117	52	142	40	167	71	191	71	218	18	245	97		
23	3	45	17	70	4	93	23	118	34	143	35	169	24	193	12	219	25	247	122		

Table 8. Irreducible trinomials of the form $x^m + 2x^t + 2$ over $GF(3)$, $2 \leq m \leq 255$

m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t	m	t		
2	1	22	5	43	17	71	51	93	23	117	65	143	35	167	71	187	65	211	89	238	5
3	1	23	3	44	3	72	28	95	47	119	3	144	56	168	28	188	11	214	65	239	5
4	1	24	4	45	17	73	1	96	16	120	4	145	73	169	37	191	71	215	59	240	8
5	1	25	3	46	5	76	9	97	81	121	1	147	43	170	43	192	32	216	4	241	117
6	1	26	7	47	15	77	25	99	19	124	25	148	3	171	151	193	81	217	85	242	115
7	5	27	7	48	8	78	13	100	25	125	73	150	73	172	19	194	55	219	25	243	121
8	2	28	2	51	1	79	53	101	31	126	49	151	125	173	7	195	49	220	15	244	31
9	5	29	25	52	7	80	2	102	25	127	119	152	18	174	73	196	79	222	89	245	97
11	3	30	1	53	13	81	41	103	47	128	6	153	59	176	12	198	29	224	12	246	13
12	2	31	5	54	1	83	27	104	5	131	27	155	129	177	83	199	35	225	209	247	125
13	1	32	5	55	11	84	14	107	3	133	15	156	26	178	11	200	3	227	11	248	50
14	1	33	5	56	3	85	31	108	2	134	61	157	69	179	59	201	113	228	14	249	59
15	7	35	17	59	17	86	13	109	9	135	91	158	61	180	38	203	3	229	79	251	9
16	4	36	14	60	2	87	37	111	13	136	57	159	127	181	37	204	50	230	73	252	98
17	1	37	13	61	7	88	6	112	6	137	1	160	4	182	25	205	9	232	30	253	7
18	7	39	7	63	37	89	13	113	19	139	59	162	19	183	181	206	61	234	91	254	73
19	11	40	1	64	3	90	19	114	7	140	59	163	59	184	20	208	10	235	83	255	229
20	5	41	1	67	11	91	17	115	83	141	5	164	15	185	121	209	49	236	9		
21	5	42	7	69	17	92	10	116	15	142	65	165	77	186	47	210	7	237	167		

Hardware Performance Characterization of Block Cipher Structures

Lu Xiao and Howard M. Heys

Electrical and Computer Engineering

Faculty of Engineering and Applied Science, Memorial University of Newfoundland
St. John's, NF, Canada A1B 3X5
`{xiao,howard}@engr.mun.ca`

Abstract. In this paper, we present a general framework for evaluating the performance characteristics of block cipher structures composed of S-boxes and Maximum Distance Separable (MDS) mappings. In particular, we examine nested Substitution-Permutation Networks (SPNs) and Feistel networks with round functions composed of S-boxes and MDS mappings. Within each cipher structure, many cases are considered based on two types of S-boxes (i.e., 4×4 and 8×8) and parameterized MDS mappings. In our study of each case, the hardware complexity and performance are analyzed. Cipher security, in the form of resistance to differential, linear, and Square attacks, is used to determine the minimum number of rounds required for a particular parameterized structure. Because the discussed structures are similar to many existing ciphers (e.g., Rijndael, Camellia, Hierocrypt, and Anubis), the analysis provides a meaningful mechanism for seeking efficient ciphers through a wide comparison of performance, complexity, and security.

1 Introduction

In product ciphers like DES [1] and Rijndael [2], the concepts of confusion and diffusion are vital to security. The Feistel network and the Substitution-Permutation Network (SPN) are two typical architectures to achieve this. In both architectures, Substitution-boxes (S-boxes) are typically used to perform substitution on small sub-blocks. An S-box is a nonlinear mapping from input bits to output bits, which meets many security requirements. In many recently proposed block ciphers (e.g., Rijndael, Hierocrypt [3], Anubis [4], and Khazad [5]), the outputs of a layer of parallel S-boxes are passed through a linear transformation based on a Maximum Distance Separable (MDS) code.

In this paper, the performance of several cipher structures is considered in terms of hardware time and space complexity. A performance comparison is made between different parameterized cases of 128-bit block ciphers in relation to security requirements. In the analysis, the hardware complexities of S-boxes and MDS mappings are based on the upper bounds of the minimum hardware complexity deduced in [6]. For a general invertible S-box, the upper bounds of

the gate count and delay are obtained from the logic minimization of a hardware-efficient S-box model; for an MDS mapping, the upper bounds of the gate count and delay are obtained by searching MDS candidates for an optimal one when implemented by bit-parallel multipliers. Hence, the structures discussed in this paper are constructed with optimized components to produce high efficiencies in their categories. A conventional evaluation approach is taken in [6] with the space complexity evaluated by the number of bit-wise invertors and 2-input gates and the time complexity evaluated by the number of traversed layers in the gate network. In this paper, a weight is associated with different types of gates to distinguish their discrepancies in hardware cost. Performance metrics are defined for hardware with consideration of complexity and security.

Many ciphers are derived from appropriate configurations of S-boxes and linear transformations (typically MDS mappings). Rijndael, Hierocrypt, and Anubis can be regarded as specific cases of nested SPNs [3]. On the other hand, the round function of a Feistel network may contain one or several layers of S-boxes followed by a linear transformation such as an MDS mapping. For example, Camellia [7] is such a cipher with one layer of S-boxes in the round function (although the linear transformations are not MDS). In this paper, many cases of these cipher structures will be analyzed for their hardware complexities and performances.

2 Background

2.1 Properties of S-boxes

The properties of the S-boxes in a cipher are important in the consideration of a cipher's security against differential cryptanalysis [8] and linear cryptanalysis [9]. An $m \times n$ S-box, S , performs a mapping from an m -bit input X to an n -bit output Y . Considering all S-boxes, $\{S_i\}$, in a cipher, the maximum differential probability p_s is defined as:

$$p_s = \max_i \max_{\Delta X \neq 0, \Delta Y} \text{prob}\{S_i(X) \oplus S_i(X \oplus \Delta X) = \Delta Y\}$$

where “ \oplus ” denotes a bitwise XOR and “ Δ ” denotes a bitwise XOR difference. The maximum linear probability is defined as:

$$q_s = \max_i \max_{\Gamma Y \neq 0, \Gamma X} (2 \times \text{prob}\{X \cdot \Gamma X = S_i(X) \cdot \Gamma Y\} - 1)^2$$

where “ \cdot ” denotes a bitwise inner product and ΓX and ΓY denote masking variables. In this paper, all 4×4 S-boxes are assumed to satisfy $p_s, q_s \leq 2^{-2}$ and all 8×8 S-boxes are assumed to satisfy $p_s, q_s \leq 2^{-6}$. Many proposed ciphers such as Serpent [10], Rijndael, Hierocrypt-3 [11], and Camellia have S-boxes with these features; others such as Anubis and Khazad have slightly higher p_s and q_s .

2.2 MDS Mappings

A linear code over Galois field $\text{GF}(2^n)$ is denoted as a (k, m, d) -code [12], where k is the symbol length of the encoded message, m is the symbol length of the original message, and d is the minimal symbol distance between any two encoded messages. An (k, m, d) -code is MDS if $d = k - m + 1$. In particular, a $(2m, m, m + 1)$ -code with generation matrix $\mathcal{G} = [\mathcal{I}|\mathcal{C}]$ where \mathcal{C} is an $m \times m$ matrix and \mathcal{I} is an identity matrix, determines an MDS mapping from the input \mathcal{X} to the output \mathcal{Y} through matrix multiplication over a Galois field as follows:

$$f_M : \mathcal{X} \mapsto \mathcal{Y} = \mathcal{C} \cdot \mathcal{X} \quad (1)$$

where

$$\mathcal{X} = \begin{pmatrix} X_0 \\ \vdots \\ X_{m-1} \end{pmatrix}, \quad \mathcal{Y} = \begin{pmatrix} Y_0 \\ \vdots \\ Y_{m-1} \end{pmatrix}, \quad \mathcal{C} = \begin{pmatrix} C_{0,0} & \dots & C_{0,m-1} \\ \vdots & \ddots & \vdots \\ C_{m-1,0} & \dots & C_{m-1,m-1} \end{pmatrix}.$$

Every entry in \mathcal{X} , \mathcal{Y} , and \mathcal{C} is an element in $\text{GF}(2^n)$.

When an invertible linear transformation $f : \mathcal{X} \mapsto \mathcal{Y}$ is used in a cipher, the avalanche effect which creates resistance to differential and linear attacks may be measured with its branch number B , which is defined as [13]:

$$B = \min_{\mathcal{X} \neq 0} \{H(\mathcal{X}) + H(\mathcal{Y})\}$$

where $H(\mathcal{X})$ and $H(\mathcal{Y})$ denotes the numbers of nonzero elements in \mathcal{X} and \mathcal{Y} . It is proved that an MDS mapping as defined in (1) has an optimal branch number B equal to $m + 1$.

2.3 Nested SPNs

The concept of a nested SPN was first introduced in [3]. In a nested SPN, S-boxes may be viewed at different levels: each S-box at a higher level is actually a small SPN at the lower level. In this paper, we examine nested SPNs which have the following properties:

- The structure contains just two levels of SPNs. A higher level S-box consists of a lower level SPN; a lower level S-box is a real 4×4 or 8×8 S-box.
- The linear transformation layers in both levels are based on MDS codes, denoted as MDS_H for the higher level and MDS_L for the lower level.
- The round key mixture occurs directly before each layer of actual (i.e., lower-level) S-boxes. One additional subkey mixture is appended at the end of the cipher structure. The subkey bits are mixed with data bits by XOR operations.
- A “round” refers to the combination of the subkey mixture, lower-level S-box layer, and subsequent MDS_L or MDS_H linear transformation.

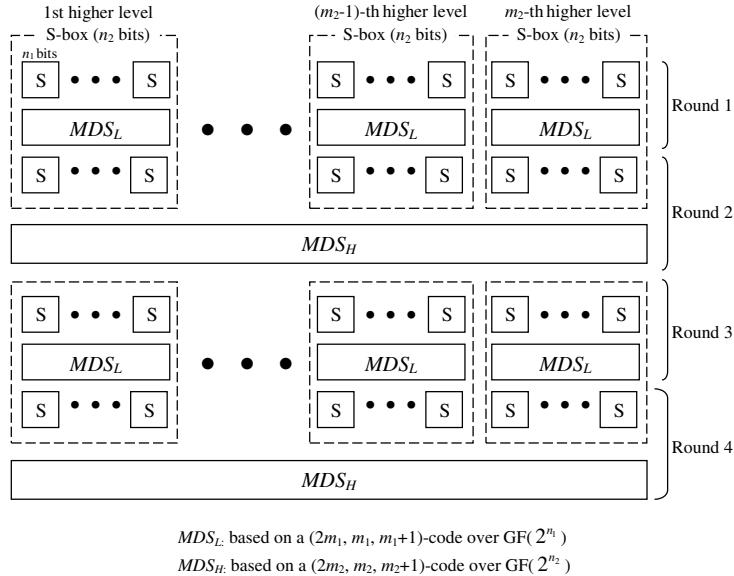


Fig. 1. Basic 2-level nested SPN (4 rounds)

As Figure 1 shows, MDS_L is an MDS mapping from a $(2m_1, m_1, m_1 + 1)$ -code over $\text{GF}(2^{n_1})$, while MDS_H is an MDS mapping from a $(2m_2, m_2, m_2 + 1)$ -code over $\text{GF}(2^{n_2})$. The variables m_1 , m_2 , n_1 , and n_2 represent parameter choices for a nested SPN.

In the most straightforward case, the output of each S-box forms one source symbol for the MDS mapping, and each encoded symbol forms the input of a subsequent S-box at the same level. So the size of an S-box is n_1 bits at the lower level and n_2 bits at the higher level. This leads to $n_2 = n_1 m_1$. Thus, the block size of the SPN is $n_1 m_1 m_2$. For example, Hierocrypt (Type I) is described as the iteration of such a 4-round structure where $n_1 = 8$, $n_2 = 32$, and $m_1 = m_2 = 4$.

At each level of a nested SPN, the branch number of the MDS layer determines the minimum number of active S-boxes in differential or linear cryptanalysis. For 4 rounds of a nested SPN, an active S-box at the higher level contains at least $m_1 + 1$ active S-boxes at the lower level. Since there are at least $m_2 + 1$ active S-boxes at the higher level, the minimum number of active lower-level S-boxes is $(m_1 + 1)(m_2 + 1)$. Therefore, the security against differential and linear attacks is evaluated as the following:

Theorem 1 (deduced from [2, 3, 13, 14]). *With the assumption that all S-box approximations involved in linear and differential cryptanalysis are independent, for 4 rounds of a nested SPN the maximum differential characteristic probability is upper bounded by $p_s^{(m_1+1)(m_2+1)}$ and the maximum linear characteristic probability is upper bounded by $q_s^{(m_1+1)(m_2+1)}$.*

The basic operations in MDS codes are multiplications and additions in finite fields. When n_2 is large, operations over $\text{GF}(2^{n_2})$ are inefficient and MDS_H can be costly in computation. An alternative method to obtain the same branch number is to concatenate several parallel MDS codes over a smaller finite field. The concatenated codes may be designed to ease a bitslice implementation.

Theorem 2 ([3]). *An MDS mapping defined by a $(2m, m, m + 1)$ -code over the nl -bit symbol set can be constructed by concatenating l mappings defined by a $(2m, m, m + 1)$ -code over the n -bit symbol set, where l can be any positive integer.*

For the example illustrated by Figure 1, since $n_2 = m_1 n_1$, the mapping MDS_H over $\text{GF}(2^{n_2})$ can be implemented with m_1 parallel MDS mappings over $\text{GF}(2^{n_1})$. In this case, the basic MDS_H layer is denoted as $1 \times (2m_2, m_2, m_2 + 1)$ over $\text{GF}(2^{n_2})$, and its simplified MDS_H layer is denoted as $m_1 \times (2m_2, m_2, m_2 + 1)$ over $\text{GF}(2^{n_2})$ where n_2 is now the size of a smaller field and for this case $n_2 = n_1$. Since $m_1 n_1$ may be factored in other ways, other simplifications are also possible. Hence we can consider that the general relation $n_2 l = m_1 n_1$ can be used to determine different cases of MDS_H defined by the values of the symbol size, n_2 , or the number of parallel MDS mappings, l . A similar approach can also be applied to the MDS_L layer. However, restrictions on values of n and m must be considered for designing a $(2m, m, m + 1)$ -code over $\text{GF}(2^n)$ such that $2m \leq 2^n + 1$ [12].

The 128-bit ciphers Square, Rijndael, and Anubis can be regarded as the iterations of 4-round nested SPNs where $n_1 = n_2 = 8$, $m_1 = m_2 = 4$. The parameters of Hierocrypt (Type II) are selected as $n_1 = 8$, $n_2 = 4$, $m_1 = m_2 = 4$.

2.4 One Type of Feistel Networks

As a typical form of block ciphers, the Feistel network has been widely used and studied. In each round i of a Feistel network as shown in Figure 2(a), the right half of the round input (denoted as X_i) goes through an F -function parameterized by round key K_i . The output of the F -function (denoted as Y_i) is XORed with the left half of the round input. The round output is the swapped result of X_i and $X_{i-1} \oplus Y_i$. The F -function is also called the round function.

Figure 2(b) illustrates a subset of Feistel networks, which has a one round SPN inside F -function. The F -function includes one layer of key addition with K_i , one layer of invertible¹ S-boxes for substitution, and an MDS mapping layer as a linear transformation. If the MDS mapping layer is constructed through concatenation of several small MDS mappings, it is necessary to include a permutation of MDS symbols in the linear transformation in order to ensure the avalanche effect.

¹ Invertible S-boxes are used so that a bijective round function can be constructed, which achieves the given upper bounds of maximal differential and linear probabilities faster in rounds than a general round function [15].

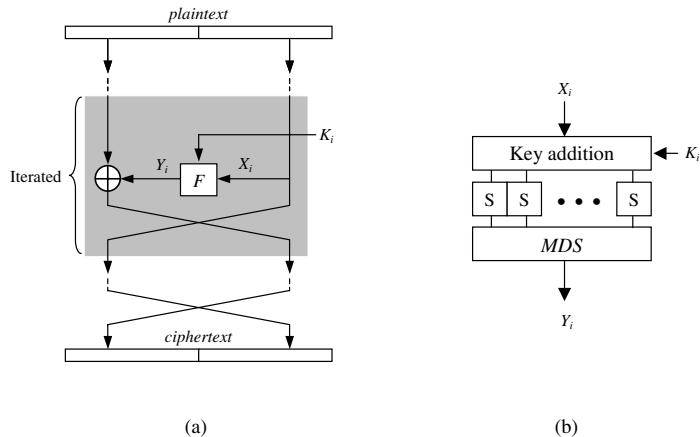


Fig. 2. (a) General encryption dataflow of a Feistel network (b) One type of F -function

In a Feistel network whose round function has an invertible linear transformation appended to parallel S-boxes, it is proved in [16] that the number of active S-boxes in any differential or linear characteristic of $4r$ rounds is lower bounded by $r \times B + \lfloor r/2 \rfloor$, where B is the branch number of the linear transformation. Therefore, we get:

Theorem 3 (deduced from [16]). *In an $4r$ Feistel cipher with a round function as Figure 2(b) shows, the maximum differential and linear characteristic probabilities are upper bounded by $p_s^{r \times B + \lfloor r/2 \rfloor}$ and $q_s^{r \times B + \lfloor r/2 \rfloor}$, respectively.*

3 Comparison of Hardware Performance

It is normally hard to compare hardware performance among different block ciphers. The main problems are: 1) each implementation represents a tradeoff between area and delay; 2) the specific hardware cost of a gate network is dependent on the target technology; 3) ciphers may contain different security margins.

For the first problem, the classical delay-area product is used to evaluate the hardware complexity universally. The typical methods used in the hardware implementation of a block cipher include a round iterated design, a pipelined design, a loop-unrolled design, and a block parallel design [18]. For a given cipher, the delay-area product is kept roughly unchanged across the different design methods (except for a loop-unrolled design), assuming the control overhead for parallelism can be ignored. If a round iterated design is regarded as a reference, a k -block parallel design using several round iterated implementations will cost about k times the number of gates and result in about $1/k$ of the average time to produce an encrypted block. The same situation occurs in a pipelined design

when each stage performs one or several rounds of the cipher. For loop unrolling, when k rounds are unrolled, the gate count will increase over an iterative design, but the average encryption time can be reduced. Loop unrolling usually results in low performance in the sense of the delay-area product.

For the second problem, a universal way is to assume that all gates have the same hardware cost [17]. Thus, the gate count and delay of all components are deduced from the upper bound of typical implementations. Such an approach leads to a measure of complexity which is technology-independent. However, in a certain target VLSI technology, the hardware costs of different gates may not be similar. In this case, it is possible to estimate the overall area (respectively, delay) by summing weighted gate counts (respectively, weighted gate layers traversed). The weights are proportional to the size of a gate (respectively, delay) and can be calculated by statistical comparison of hardware among gates based on a target technology. The hardware complexity is then evaluated by weighted area A_W and weighted delay D_W :

$$A_W = \sum_{\text{gate type } u} G(u) \times W_G(u) \quad (2)$$

$$D_W = \sum_{\text{gate type } u} D(u) \times W_D(u). \quad (3)$$

Associated with gate type u , $G(u)$ and $W_G(u)$ return the gate count and weight of each gate. In the critical path of the circuit, $D(u)$ and $W_D(u)$ return the number of traversed gate layers and weight of each layer associated with gate type u .

For the problem caused by different security margins, we use a rule-of-thumb to determine resistance to differential and linear cryptanalysis. For differential cryptanalysis, the number of chosen plaintext pairs to attack a cipher is expected to be in the order of $1/P_d$, where P_d is the maximum differential characteristic probability determined by Theorems 1 and 3. Similarly, to attack a cipher using linear cryptanalysis, the number of known plaintexts is expected to be in the order of $1/P_l$, where P_l is the maximum linear characteristic probability.

Based on above considerations, we define three hardware performance metrics η_s , η_t , and η to measure the space, time, and overall performance, respectively. The three metrics integrate security and complexity and are defined as follows:

$$\eta_s = \frac{\log_2 1/P}{\# \text{ of rounds} \times A_W \text{ per round}} \quad (4)$$

$$\eta_t = \frac{\log_2 1/P}{\# \text{ of rounds} \times D_W \text{ per round}} \quad (5)$$

$$\eta = \frac{\log_2 1/P}{\# \text{ of rounds} \times (A_W \times D_W \text{ per round})} \quad (6)$$

where $P = P_d$ for hardware performance in relation to differential attacks and $P = P_l$ in relation to linear attacks. In each expression, the numerator is essentially a security measure in bits and the denominator is a complexity measure.

Since we assume that the S-boxes in the three discussed cipher structures satisfy $p_s = q_s$, the values of $\log_2 1/P_d$ and $\log_2 1/P_l$ are the same. For the nested SPNs and Feistel networks discussed in Section 2, $\log_2 1/P$ is a linear function of the number of rounds. Therefore, the values of η_s , η_t , and η indicate how much security is expected to be obtained for a specific hardware cost, regardless of the number of rounds in a cipher.

Targeted to the same design method, η_s shows the security contribution provided by each area unit; η_t shows the security contribution provided by each delay unit. For a fast implementation such as a pipelined or parallel design, a high η_s means that many independent blocks can be processed simultaneously. For a round iterated design, a high η_t means that the encryption time for a block is small. More generally, using the classical delay-area product as its denominator, η indicates the performance integrating both the delay and area complexities.

The cases that we compare in the following sections are generated as 128-bit block ciphers defined by the nested SPN and Feistel networks. To calculate the gate count and number of gate layers per round, we consider the construction of the combinational circuits of the round structure with S-box and MDS mapping components which can produce high efficiencies in hardware. The hardware design and optimization of these components are described in [6]. The detailed data used in the complexity estimation is presented in the Appendix.

3.1 Hardware Performance of Nested SPNs

A set of nested SPNs can be generated with appropriate configurations of parameterized MDS_L , MDS_H , and S-boxes. As Theorem 2 illustrates, the MDS mapping defined over a large Galois field can be simplified using several mappings in a smaller Galois field. Table 1 lists the cases of nested SPNs in 12 categories (labelled as N1 to N12) defined by the S-boxes and MDS_L . Thus, the cases within a category only differ in the simplification of MDS_H . Each case can be regarded as a 128-bit cipher, after a particular key schedule is defined. Due to the difficulty of finding optimized MDS mappings, the cases with a Galois field larger than $GF(2^8)$ are not considered.

In relation to real ciphers, Case N4-a includes Square, Rijndael, and Anubis. Type II of Hierocrypt belongs to Case N4-b with a simplified MDS_H over $GF(2^4)$. Similar to SHARK [13] and Khazad, Case N8 is a one-level SPN. However, SHARK and Khazad are 64-bit ciphers because their MDS mappings are based on a (16, 8, 9)-code over $GF(2^8)$.

From the viewpoint of implementation, a nested SPN follows the iterative dataflow of key addition, one S-box layer, and an MDS mapping layer (either MDS_L or MDS_H). Since S-boxes cost the most hardware complexity, a 128-bit multiplexor selects MDS_L and MDS_H dynamically such that only one layer of S-boxes is needed in a round iterated design. So assuming a round iterated implementation, the round circuit used for each case in Table 1 includes a 128-bit key addition, one layer of S-boxes, MDS_L , MDS_H , and a 128-bit multiplexor².

² MDS Multiplexing is not necessary for N8.

Table 1. 128-bit nested SPNs of $4r$ rounds

Case	S-box size	$MDS_L :$ $l_1 \times (2m_1, m_1, m_1+1)$ over $GF(2^{n_1})$	$MDS_H :$ $l_2 \times (2m_2, m_2, m_2+1)$ over $GF(2^{n_2})$	P_d, P_t
N1-a	8×8	8×(4, 2, 3) over $GF(2^8)$	2×(16, 8, 9) over $GF(2^8)$	2^{-162r}
N1-b			4×(16, 8, 9) over $GF(2^4)$	
N2-a	8×8	16×(4, 2, 3) over $GF(2^4)$	2×(16, 8, 9) over $GF(2^8)$	2^{-162r}
N2-b			4×(16, 8, 9) over $GF(2^4)$	
N3-a	8×8	32×(4, 2, 3) over $GF(2^2)$	2×(16, 8, 9) over $GF(2^8)$	2^{-162r}
N3-b			4×(16, 8, 9) over $GF(2^4)$	
N4-a	8×8	4×(8, 4, 5) over $GF(2^8)$	4×(8, 4, 5) over $GF(2^8)$	2^{-150r}
N4-b			8×(8, 4, 5) over $GF(2^4)$	
N5-a	8×8	8×(8, 4, 5) over $GF(2^4)$	4×(8, 4, 5) over $GF(2^8)$	2^{-150r}
N5-b			8×(8, 4, 5) over $GF(2^4)$	
N6-a	8×8	2×(16, 8, 9) over $GF(2^8)$	8×(4, 2, 3) over $GF(2^8)$	2^{-162r}
N6-b			16×(4, 2, 3) over $GF(2^4)$	
N7-a	8×8	4×(16, 8, 9) over $GF(2^4)$	8×(4, 2, 3) over $GF(2^8)$	2^{-162r}
N7-b			16×(4, 2, 3) over $GF(2^4)$	
N7-c			32×(4, 2, 3) over $GF(2^2)$	
N8	8×8	1×(32, 16, 17) over $GF(2^8)$	same as MDS_L	2^{-204r}
N9	4×4	16×(4, 2, 3) over $GF(2^4)$	1×(32, 16, 17) over $GF(2^8)$	2^{-102r}
N10	4×4	32×(4, 2, 3) over $GF(2^2)$	1×(32, 16, 17) over $GF(2^8)$	2^{-102r}
N11-a	4×4	8×(8, 4, 5) over $GF(2^4)$	2×(16, 8, 9) over $GF(2^8)$	2^{-90r}
N11-b			4×(16, 8, 9) over $GF(2^4)$	
N12-a	4×4	4×(16, 8, 9) over $GF(2^4)$	4×(8, 4, 5) over $GF(2^8)$	2^{-90r}
N12-b			8×(8, 4, 5) over $GF(2^4)$	

The 128-bit multiplexor can be implemented by 385 NAND gates (i.e., $y = x_1 \cdot c + x_2 \cdot \bar{c}$ where c is the select signal and “+” denotes OR).

For the main components and the iterative round structures of each SPN, Table A-1 in the Appendix lists their gate counts and delays of layers. Although each individual value in Table A-1 cannot be perfectly accurate, the comparison of these measures does enable us to distinguish the cases which are more efficient in hardware.

Figure 3 shows the tendency of the universal performance comparison when $W_G(u) = W_D(u) = 1$ for any gate type u (i.e., all gates are assumed to have the same hardware cost). In an ASIC design, XOR gates are more expensive than other gates such as NOT, AND, and OR gates. Figure 4 shows a weighted performance comparison when $W_G(\text{XOR}) = W_D(\text{XOR}) = 2$ and weight for others is one. The two figures follow the similar tendency in performance comparison:

- The size of the S-box largely determines space and time performances. Using small S-boxes tends to cost less hardware area, but more delay than using

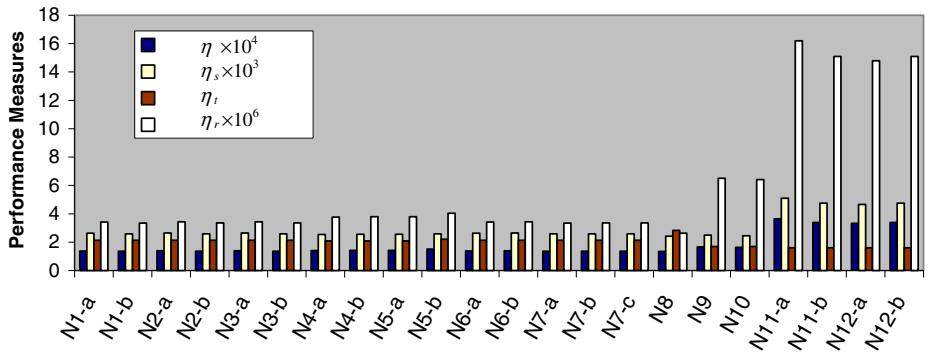


Fig. 3. Universal performance comparison of nested SPNs

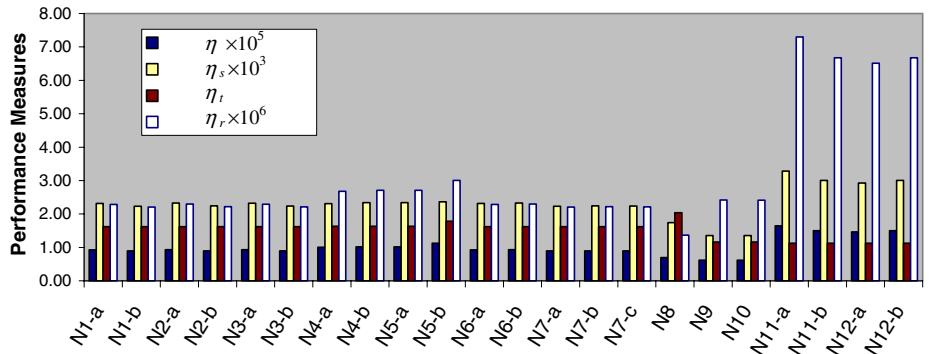


Fig. 4. Weighted performance comparison of nested SPNs

large S-boxes. Given fixed chip area, the cipher cases using small S-boxes are more advantageous for parallelism as their higher η_s values show.

- Many SPN structures (N1-N10, N11-N12) are essentially equivalent with respect to their hardware performance. Hence, it is wise for a cipher designer to consider those structures which can facilitate software implementations.
- When the symbol size is 8 bits or less, the simplification of MDS mappings through concatenation does not significantly improve the performance when the MDS mappings have been selected to be optimized for hardware. For example, Case N4-b in Table 1 does not gain a much higher improvement in hardware than Case N4-a.
- When m_1 or m_2 is very high, the MDS mapping determined by m_1 or m_2 (e.g., MDS_H in cases of N9 and N10) will cost much more hardware and overwhelm S-box costs, which degrades the cipher performance.

- As a cipher of Case N4-a, Rijndael is very suitable for a round iterated design. However, its suitability for pipelined or parallel implementations is not as high as cipher cases using 4×4 S-boxes such as cases of N11 and N12.

The above conclusions are based on hardware complexity and security against differential and linear attacks. For some other attacks such as Square attack, the effectiveness significantly decreases after a certain number. In this circumstance, a performance metric of the round structure is defined as:

$$\eta_r = \frac{1}{A_W \times D_W \text{ per round}} .$$

Since the security in bits to resist these attacks increases very rapidly in the number of rounds, with a trend much steeper than differential and linear attacks as more rounds are appended, we take a fixed number of rounds (e.g., about 8 for the Square attack to Rijndael) as enough for the security. The comparison of round performance is also included in Figures 3 and 4. It is obvious that the nested SPNs with small S-boxes and modest sized MDS_L and MDS_H have significantly better performance in relation to the Square attack than other cases.

3.2 Hardware Performance of Feistel Networks

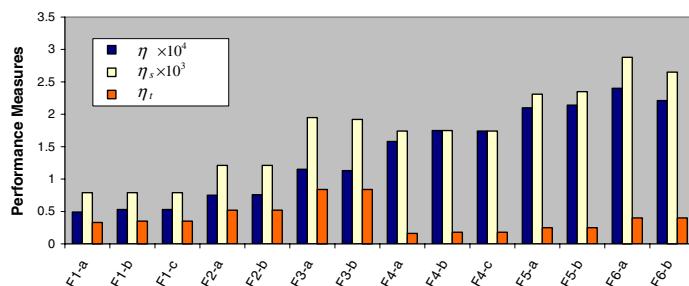
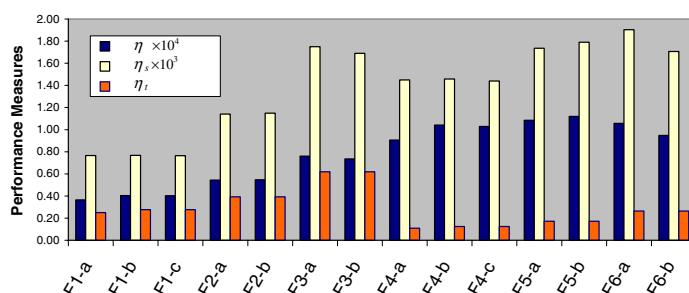
The Feistel network discussed in this section is limited to the subset described in Section 2.4, which has an SPN round function. To construct a typical 128-bit cipher, such a Feistel network has a 64-bit F -function which contains sixteen 4×4 or eight 8×8 parallel S-boxes followed by an MDS mapping layer. As listed in Table 2, six categories (labelled as F1 to F6) of these 128-bit Feistel networks can be generated. To ensure a good avalanche effect, an appropriate fixed permutation of MDS symbols after the MDS mapping is expected, which does not cost any gates. The hardware of one round of the cipher includes a 64-bit XOR for round key addition, one layer of S-boxes, one MDS mapping, and another 64-bit XOR appended to the output of the F -function. The cases of the same category in Table 2 only differ in the simplification of the MDS mapping. The performance comparison in Figures 5 and 6 indicates (refer to the Appendix for detailed data):

- It is useful to pick an MDS mapping that has a large branch number (i.e., $m+1$). The cases with such an MDS mapping have significantly higher values in all three performance measures.
- With high η_t values, the cases with 8×8 S-boxes demonstrate high performance in non-pipelined and non-parallel implementations. With high η_s values, the cases with 4×4 S-boxes demonstrate high performance in pipelined and parallel implementations because many independent blocks can be processed simultaneously.

Camellia is a 128-bit Feistel cipher with a 64-bit round function which consists of eight 8×8 invertible S-boxes and a linear transformation. Hence, Camellia

Table 2. 128-bit Feistel networks of $4r$ rounds

Case	S-box size	MDS $l \times (2m, m, m+1)$ over $GF(2^n)$	P_d, P_t
F1-a	8×8	$4 \times (4, 2, 3)$ over $GF(2^8)$	$2^{-6(3r+\lfloor \frac{r}{2} \rfloor)}$
F1-b		$8 \times (4, 2, 3)$ over $GF(2^4)$	
F1-c		$8 \times (4, 2, 3)$ over $GF(2^4)$	
F2-a	8×8	$2 \times (8, 4, 5)$ over $GF(2^8)$	$2^{-6(5r+\lfloor \frac{r}{2} \rfloor)}$
F2-b		$4 \times (8, 4, 5)$ over $GF(2^4)$	
F3-a	8×8	$1 \times (16, 8, 9)$ over $GF(2^8)$	$2^{-6(9r+\lfloor \frac{r}{2} \rfloor)}$
F3-b		$2 \times (16, 8, 9)$ over $GF(2^4)$	
F4-a	4×4	$4 \times (4, 2, 3)$ over $GF(2^8)$	$2^{-2(3r+\lfloor \frac{r}{2} \rfloor)}$
F4-b		$8 \times (4, 2, 3)$ over $GF(2^4)$	
F4-c		$16 \times (4, 2, 3)$ over $GF(2^2)$	
F5-a	4×4	$2 \times (8, 4, 5)$ over $GF(2^8)$	$2^{-2(5r+\lfloor \frac{r}{2} \rfloor)}$
F5-b		$4 \times (8, 4, 5)$ over $GF(2^4)$	
F6-a	4×4	$1 \times (16, 8, 9)$ over $GF(2^8)$	$2^{-2(9r+\lfloor \frac{r}{2} \rfloor)}$
F6-b		$2 \times (16, 8, 9)$ over $GF(2^4)$	

**Fig. 5.** Universal performance comparison of Feistel networks**Fig. 6.** Weighted performance comparison of Feistel networks

is similar to our discussed Feistel networks but does not use an MDS mapping. The branch number of the Camellia linear transformation is 5. An efficient implementation of such a linear transformation costs 176 two-input XOR gates and a delay of 3 gate layers in universal comparison. Thus, Camellia has performance similar to Case F2-a which has 264 XOR gates and a delay of 3 gate layers (see Table A-2 in the Appendix). Compared with the case F3-a, Camellia has a slightly more compact round structure (i.e., about 5% less in gate count than Case F3-a). However, each round of Camellia contributes much less to the security. Eleven rounds of F3-a provides equivalent security to nineteen rounds of Camellia. Further calculation shows that the overall hardware performance of F3-a is about 50% higher than that of Camellia. The weighted performance comparison follows a similar trend.

3.3 Synthesis Results

The above performance analysis is based on theoretical evaluation of hardware complexity. The usability of these analytical results can be verified when VLSI technology is targeted. To avoid arduous work on synthesizing each cipher case, we did a high level synthesis of each component used in Tables 1 and 2. The components are coded in VHDL and synthesized with Synopsys Design Compiler. Two CMOS libraries³ were used where most standard cells have one or two bitwise inputs.

During synthesis, if the minimum area (respectively, delay) is set as the main constraint⁴, the numbers of equivalent gates (respectively, critical delay time) of 8×8 S-boxes are close to their estimates in Tables A-1 and A-2. The gates and delays of 4×4 S-boxes are slightly less than their estimates because it is much easier for CAD tools to simplify smaller S-boxes. This effect indicates that the performance advantage of using small S-boxes as shown in Figures 3 to 6 is decent and slightly understated.

Since the MDS mapping is implemented in XOR gates, the areas and delays closely follow the proportional relation of their estimations in Tables A-1 and A-2. Because XOR gates are larger and slower than other gate types, synthesis tools may replace them with other gates such as NXORs during optimization. Nevertheless, the delays and numbers of equivalent gates imply that a weight of 2 is reasonable for an XOR gate. This effect makes the cases with large MDS mapping worse in weighted performance, e.g., the cases in N8 to N12, F5, and F6. This problem is encountered in the realizations where a large percent of XORs are used. The weighted performance shown in Figures 4 and 6 are thus more useful for a closer comparison than the universal method.

³ lsi_10k.db and TSMC's 0.18 μ m CMOS library are targeted separately.

⁴ When other constraint is set, the absolute values of area and delay will vary, but their comparison trend follows a similar trend.

4 Conclusions

In this paper we have considered two cipher structures composed of S-boxes and MDS mappings. Various cipher cases are generated from these structures with different component configurations. Their security and complexity are examined and integrated into performance metrics.

In hardware, the discussed cipher cases using large S-boxes are suitable for non-pipelined and non-parallel applications where delay is the main design criterion; however, in pipelined and parallel applications, the cipher cases using small S-boxes produce high performance. Further, appropriate selection of an MDS mapping layer is important for security against differential and linear attacks.

Compared with Feistel networks, the nested SPNs generally have higher hardware performance. When the same S-boxes are used, a nested SPN tends to be more efficient in hardware to resist differential and linear attacks. Considering the threat of Square attacks, nested SPNs with smaller S-boxes are preferred. For a Feistel network, more rounds are needed to be secure against differential and linear attacks. With little change in the linear transformation, a suggestion is made to improve Camellia in terms of security and hardware efficiency.

In line with a nested SPN, MISTY [19] can be regarded as a nested Feistel network. Using provable security as the security measure, it will be interesting future work to compare the hardware performance between these two nested structures with similar performance metrics defined in Section 3.

References

- [1] National Institute of Standards and Technology, “Data Encryption Standard (DES)”, *Federal Information Processing Standard 46*, 1977. [176](#)
- [2] J. Daemen and V. Rijmen, “AES Proposal: Rijndael”, *Advanced Encryption Standard*, available on: csrc.nist.gov/encryption/aes/rijndael. [176](#), [179](#)
- [3] K. Ohkuma, H. Muratani, F. Sano, and S. Kawamura, “The Block Cipher Hierocrypt”, *Workshop on Selected Areas in Cryptography - SAC 2000*, Lecture Notes in Computer Science 2012, Springer-Verlag, pp. 72-88, 2001. [176](#), [177](#), [178](#), [179](#), [180](#)
- [4] P. Barreto and V. Rijmen, “The Anubis Block Cipher”, *NESSIE Algorithm Submission*, 2000, available on: www.cosic.esat.kuleuven.ac.be/nessie. [176](#), [190](#)
- [5] P. Barreto and V. Rijmen, “The Khazad Legacy-Level Block Cipher”, *NESSIE Algorithm Submission*, 2000, available on: www.cosic.esat.kuleuven.ac.be/nessie. [176](#)
- [6] L. Xiao and H. M. Heys, “Hardware Design and Analysis of Block Cipher Components”, accepted for presentation at the *5th International Conference on Information Security and Cryptology - ICISC 2002*, Seoul, Korea, November 28-29, 2002. [176](#), [177](#), [183](#), [190](#)
- [7] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, “Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms”, *NESSIE Algorithm Submission*, 2000, available on: www.cosic.esat.kuleuven.ac.be/nessie. [177](#)

- [8] E. Biham and A. Shamir, “Differential cryptanalysis of DES-like cryptosystems”, *Advances in Cryptology - CRYPTO ’90*, Lecture Notes in Computer Science 537, pp. 2-21. Springer-Verlag, 1991. [177](#)
- [9] M. Matsui, “Linear Cryptanalysis Method for DES Cipher”, *Advances in Cryptology - Eurocrypt ’93*, Lecture Notes in Computer Science 765, Springer-Verlag, pp. 386-397, 1993. [177](#)
- [10] R. Anderson, E. Biham and L. Knudsen, “Serpent: A Proposal for the Advanced Encryption Standard”, *AES Algorithm Submission*, available on: www.cl.cam.ac.uk/~rja14/serpent.html [177](#)
- [11] Toshiba Corporation, “Security Evaluation: Hierocrypt-3”, *NESSIE Algorithm Submission*, 2000, available on: www.cosic.esat.kuleuven.ac.be/nessie. [177](#)
- [12] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North-Holland, Amsterdam, 1977. [178](#), [180](#)
- [13] V. Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win, “The cipher SHARK”, *Workshop on Fast Software Encryption - FSE ’96*, Lecture Notes in Computer Science 1039, Springer-Verlag, pp. 99-112, 1997. [178](#), [179](#), [183](#)
- [14] J. Daemen, L. R. Knudsen, and V. Rijmen, “The Block Cipher Square”, *Workshop on Fast Software Encryption - FSE ’97*, Lecture Notes in Computer Science 1267, Springer-Verlag, pp. 54-68, 1997. [179](#)
- [15] M. Kanda, Y. Takashima, T. Matsumoto, K. Aoki, and K.Ohta, “Strategy for Constructing Fast Round Functions with Practical Security Against Differential and Linear Cryptanalysis”, *Workshop on Selected Areas in Cryptography - SAC ’98*, Lecture Notes in Computer Science 1556 , pp. 264-279, 1999. [180](#)
- [16] M. Kanda, “Practical Security Evaluation against Differential and Linear Attacks for Feistel Ciphers with SPN Round Function”, *Workshop on Selected Areas in Cryptography - SAC 2000*, Lecture Notes in Computer Science 2012, Springer-Verlag, pp. 324-338, 2001. [181](#)
- [17] C. Paar, “Efficient VLSI Architectures for Bit-Parallel Computation in Galois Fields”, Ph.D. Thesis, Institute for Experimental Mathematics, University of Essen, Germany, 1994. [182](#)
- [18] J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, “Report on the Development of the Advanced Encryption Standard (AES)”, *Report on the AES Selection from U. S. National Institute of Standards and Technology (NIST)*, available on: csrc.nist.gov/encryption/aes. [181](#)
- [19] M. Matsui, “New Block Encryption Algorithm MISTY”, *Workshop on Fast Software Encryption - FSE ’97*, Lecture Notes in Computer Science 1267, Springer-Verlag, pp. 54-68, 1997. [189](#)

A Complexity Evaluation of Cipher Components

In hardware, the complexity of S-boxes are evaluated through the simplification results deduced from an encoder-switch-decoder model [6]. In this model, S-boxes are composed of low complexity gates (ANDs, ORs, and NOTs). A 4×4 S-box can be implemented using 50 gates and produces a delay of 6 gate layers; an 8×8 S-box can be implemented using 806 gates and produces a delay of 11 gate layers. Involution MDS codes [4] are found by searching Hadamard matrices and have been optimized for hardware [6]. MDS codes are composed of XORs. The evaluated hardware costs of S-boxes, MDS mappings, and round structures are

Table A-1. Complexity and universal performance estimation of one round of 128-bit nested involution SPNs in hardware

Case	S-boxes	MDS_L	MDS_H	Round Total (universal)	η_s	η_t	η	η_r
	Gate#/Delay	XOR#/Delay	XOR#/Delay	Gate#/Delay	(10 ⁻³)	(10 ⁻⁴)	(10 ⁻⁶)	
N1-a	12896 / 11	256 / 3	1728 / 5	15393 / 19	2.63	2.13	1.38	3.42
N1-b	12896 / 11	256 / 3	2048 / 5	15713 / 19	2.58	2.13	1.36	3.35
N2-a	12896 / 11	208 / 2	1728 / 5	15345 / 19	2.64	2.13	1.39	3.43
N2-b	12896 / 11	208 / 2	2048 / 5	15665 / 19	2.59	2.13	1.36	3.36
N3-a	12896 / 11	224 / 2	1728 / 5	15361 / 19	2.64	2.13	1.39	3.43
N3-b	12896 / 11	224 / 2	2048 / 5	15681 / 19	2.58	2.13	1.36	3.36
N4-a	12896 / 11	672 / 4	672 / 4	14753 / 18	2.54	2.08	1.41	3.77
N4-b	12896 / 11	672 / 4	576 / 3	14657 / 18	2.56	2.08	1.42	3.79
N5-a	12896 / 11	576 / 3	672 / 4	14657 / 18	2.56	2.08	1.42	3.79
N5-b	12896 / 11	576 / 3	576 / 3	14561 / 17	2.58	2.21	1.51	4.04
N6-a	12896 / 11	1728 / 5	256 / 3	15393 / 19	2.63	2.13	1.38	3.42
N6-b	12896 / 11	1728 / 5	208 / 2	15345 / 19	2.64	2.13	1.39	3.43
N7-a	12896 / 11	2048 / 5	256 / 3	15713 / 19	2.58	2.13	1.36	3.35
N7-b	12896 / 11	2048 / 5	208 / 2	15665 / 19	2.59	2.13	1.36	3.36
N7-c	12896 / 11	2048 / 5	224 / 2	15681 / 19	2.58	2.13	1.36	3.36
N8	12896 / 11	8064 / 6	8064 / 6	21088 / 18	2.42	2.83	1.34	2.63
N9	1600 / 6	208 / 2	8064 / 6	10257 / 15	2.49	1.70	1.66	6.50
N10	1600 / 6	224 / 2	8064 / 6	10401 / 15	2.45	1.70	1.63	6.41
N11-a	1600 / 6	576 / 3	1728 / 5	4417 / 14	5.09	1.61	3.64	16.2
N11-b	1600 / 6	576 / 3	2048 / 5	4737 / 14	4.75	1.61	3.39	15.1
N12-a	1600 / 6	2048 / 5	672 / 4	4833 / 14	4.66	1.61	3.33	14.8
N12-b	1600 / 6	2048 / 5	576 / 3	4737 / 14	4.75	1.61	3.39	15.1

Table A-2. Complexity and universal performance estimation of one round of 128-bit Feistel networks in hardware

Case	S-boxes	MDS	Round Total (universal)	η_s	η_t	η
	Gate # / Delay	XOR # / Delay	Gate # / Delay	(10 ⁻³)	(10 ⁻⁴)	
F1-a	6448 / 11	76 / 3	6652 / 16	0.79	0.33	0.49
F1-b	6448 / 11	72 / 2	6648 / 15	0.79	0.35	0.53
F1-c	6448 / 11	80 / 2	6656 / 15	0.79	0.35	0.53
F2-a	6448 / 11	264 / 3	6840 / 16	1.21	0.52	0.75
F2-b	6448 / 11	240 / 3	6816 / 16	1.21	0.52	0.76
F3-a	6448 / 11	720 / 4	7296 / 17	1.95	0.84	1.15
F3-b	6448 / 11	864 / 4	7440 / 17	1.92	0.84	1.13
F4-a	800 / 6	76 / 3	1004 / 11	1.74	0.16	1.58
F4-b	800 / 6	72 / 2	1000 / 10	1.75	0.18	1.75
F4-c	800 / 6	80 / 2	1008 / 10	1.74	0.18	1.74
F5-a	800 / 6	264 / 3	1192 / 11	2.31	0.25	2.10
F5-b	800 / 6	240 / 3	1168 / 11	2.35	0.25	2.14
F6-a	800 / 6	720 / 4	1648 / 12	2.88	0.40	2.40
F6-b	800 / 6	864 / 4	1792 / 12	2.65	0.40	2.21

listed in Tables A-1 and A-2 for each cipher case. The values of performance metrics in the two tables are calculated for universal comparison only.

Using these results, the complexity of each 128-bit 2-level nested SPN is evaluated for each round. The hardware of one round SPN includes a 128-bit key addition layer, an S-box layer, two MDS mappings at different levels, and a 128-bit multiplexor. The 128-bit multiplexor selects MDS_L and MDS_H alternatively in consecutive rounds, which costs 385 NAND gates and a delay of two gate

layers. The key addition costs 128 XOR gates and a delay of one gate level. The calculation of the delay per round assumes the highest delay of MDS_L and MDS_H .

The hardware of one round of the Feistel network includes a 64-bit key addition layer, an S-box layer, an MDS mapping layer, and a 64-bit XOR after the F -function (as shown in Figure 2(a)). The key addition costs 64 XOR gates and a delay of one gate level. The XOR after the F -function has the same hardware complexity as the key addition.

Simple Identity-Based Cryptography with Mediated RSA^{*}

Xuhua Ding and Gene Tsudik

Department of Information and Computer Science, University of California, Irvine
`{xhding,gts}@ics.uci.edu`

Abstract. Identity-based public key encryption facilitates easy introduction of public key cryptography by allowing an entity's public key to be derived from an arbitrary identification value, such as name or email address. The main practical benefit of identity-based cryptography is in greatly reducing the need for, and reliance on, public key certificates. Although some interesting identity-based techniques have been developed in the past, none are compatible with popular public key encryption algorithms (such as El Gamal and RSA). This limits the utility of identity-based cryptography as a transitional step to full-blown public key cryptography. Furthermore, it is fundamentally difficult to reconcile fine-grained revocation with identity-based cryptography.

Mediated RSA (mRSA) [9] is a simple and practical method of splitting a RSA private key between the user and a Security Mediator (SEM). Neither the user nor the SEM can cheat one another since each cryptographic operation (signature or decryption) involves both parties. mRSA allows fast and fine-grained control of users' security privileges. However, mRSA still relies on conventional public key certificates to store and communicate public keys. In this paper, we present IB-mRSA, a simple variant of mRSA that combines identity-based and mediated cryptography. Under the random oracle model, IB-mRSA with OAEP [7] is shown as secure (against adaptive chosen ciphertext attack) as standard RSA with OAEP. Furthermore, IB-mRSA is simple, practical, and compatible with current public key infrastructures.

Keywords: Identity-based encryption, mediated RSA, revocation.

1 Introduction

In a typical public key infrastructure (PKI) setting, a user's public key is explicitly encoded in a public key certificate which is, essentially, a binding between the certificate holder's identity and the claimed public key. This common model requires universal trust in certificate issuers (Certification Authorities or CAs). It has some well-known and bothersome side-effects such as the need for cross-domain trust and certificate revocation. The main problem, however, is the basic

* This work was supported by DARPA contract F30602-99-1-0530.

assumption that all certificates are public, ubiquitous and, hence, readily available to anyone. We observe that this assumption is not always realistic, especially, in wireless (or any fault-prone) networks where connectivity is sporadic.

In contrast, identity-based cryptography changes the nature of obtaining public keys by constructing a one-to-one mapping between identities and public keys. Identity-based cryptography thus greatly reduces the need for, and reliance on, public key certificates and certification authorities. In general, identity-based encryption and identity-based signatures are useful cryptographic tools that facilitate easy introduction of, and/or conversion to, public key cryptography by allowing a public key to be derived from arbitrary identification values such as email addresses or phone numbers. At the same time, identity-based methods greatly simplify key management since they reduce both: the need for, and, the number of, public key certificates.

The concept of identity-based public encryption was first proposed by Shamir [20] in 1984. For the following 16 years the progress in this area has been rather slow. However, recently, Boneh and Franklin developed an elegant Identity-Based Encryption system (BF-IBE) based on Weil Pairing on elliptic curves [10]. BF-IBE represents a significant advance in cryptography.

Nevertheless, an identity-based RSA variant has remained elusive for the simple reason that an RSA modulus n (a product of two large primes) can not be safely shared among multiple users. Another notable drawback of current identity-based cryptographic methods is lack of support for fine-grained revocation. Revocation is typically done via Certificate Revocation Lists (CRLs) or similar structures. However, IBE aims to simplify certificate management by deriving public keys from identities, which makes it difficult to control users' security privileges.

In this paper, we propose a simple identity-based cryptosystem developed atop some Mediated RSA (mRSA) by Boneh, et al. [9]. mRSA is a practical and RSA-compatible method of splitting an RSA private key between the user and the security mediator, called a SEM. Neither the user nor the SEM knows the factorization of the RSA modulus and neither can decrypt/sign message without the other's help. By virtue of requiring the user to contact its SEM for each decryption and/or signature operation, mRSA provides fast and fine-grained revocation of users' security privileges.

Built on top of mRSA, IB-mRSA blends the features of identity-based and mediated cryptography and also offers some practical benefits.¹ Like mRSA, it is fully compatible with plain RSA. With the exception of the identity-to-public-key mapping, it requires no special software for communicating parties. IB-mRSA also allows optional public key certificates which facilitates easy transition to a conventional PKI. More generally, IB-mRSA can be viewed as a simple and practical technique inter-operable with common modern PKIs. At the same time, IB-mRSA offers security comparable to that of RSA, provided that a SEM is not compromised. Specifically, it can be shown that, in the random oracle model, IB-mRSA with OAEP [7] is as secure – against adaptive chosen ciphertext attacks – as RSA with OAEP.

¹ A very sketchy version of IB-mRSA was first presented in [8].

The rest of the paper is organized as follows. The next section gives a detailed description of IB-mRSA. The security analysis is presented in Section 3 and the performance analysis – in Section 4. In Section 5, IB-mRSA is compared with Boneh-Franklin’s IBE. Finally, a brief description of the implementation is presented in the last section. Some further security details can be found in the Appendix.

2 Identity-Based mRSA

The main feature of identity-based encryption is the sender’s ability to encrypt messages using the public key derived from the receiver’s identity and other public information. The identity can be the receiver’s email address, user id or any value unique to the receiver; essentially, an arbitrary string. To compute the encryption key, an efficient (and public) mapping function \mathcal{KG} must be set beforehand. This function must be a one-to-one mapping from identity strings to public keys.

The basic idea behind identity-based mRSA is the use of a single common RSA modulus n for all users within a system (or domain). This modulus is public and contained in a system-wide certificate issued, as usual, by some Certificate Authority (CA). To encrypt a message for a certain recipient (Bob), the sender (Alice) first computes $e_{Bob} = \mathcal{KG}(ID_{Bob})$ where ID_{Bob} is the recipient’s identity value, such as Bob’s email address. Thereafter, the pair (e_{Bob}, n) is treated as a plain RSA public key and normal RSA encryption is performed. On Bob’s side, the decryption process is identical to that of mRSA.

We stress that using the same modulus by multiple users in a normal RSA setting is **utterly insecure**. It is subject to a trivial attack whereby anyone – utilizing one’s knowledge of a single key-pair – can simply factor the modulus and compute the other user’s private key. However, in the present context, we make an important assumption that: *Throughout the lifetime of the system, the adversary is unable to compromise a SEM.*

Obviously, without this assumption, IB-mRSA would offer no security whatsoever: a single SEM break-in coupled with the compromise of just one user’s key share would result in the compromise of all users’ (for that SEM) private keys. The IB-mRSA assumption is slightly stronger than its mRSA counterpart. Recall that, in mRSA, each user has a different RSA setting, i.e., a unique modulus. Therefore, to compromise a given user an adversary has to break into both the user and its SEM.

We now turn to the detailed description of the IB-mRSA scheme.

2.1 System Setting and User Key Generation

In the following, we use email addresses as unique identifiers of the public key owners in the system. However, as mentioned above, other identity types can be used just as well, e.g., Unix UIDs, HTTP addresses, physical addresses or even phone numbers. We use the notation ID_{Alice} to denote the user’s (Alice) email address that will be used to derive the public exponent.

Algorithm IB-mRSA.key (executed by CA)

Let k (even) be the security parameter

1. Generate random $k/2$ -bit primes: p', q' s.t. $p = 2p' + 1, q = 2q' + 1$ are also prime.
2. $n \leftarrow pq, e \in \mathcal{R} \mathbb{Z}_{\phi(n)}^*, d \leftarrow e^{-1} \bmod \phi(n)$
3. For each user (Alice):
 - (a) $s \leftarrow k - |\mathcal{KG}()| - 1$
 - (b) $e_{Alice} \leftarrow 0^s || \mathcal{KG}(ID_A) || 1$
 - (c) $d_{Alice} \leftarrow 1/e_{Alice} \bmod \phi(n)$
 - (d) $d_{Alice,u} \xleftarrow{r} Z_n - \{0\}$
 - (e) $d_{Alice,sem} \leftarrow (d - d_{Alice,u}) \bmod \phi(n)$

Fig. 1. IB-mRSA: User key generation

In the initialization phase, a trusted party (CA) sets up the RSA modulus for all users in the same system (organization or domain). First, CA chooses, at random, two large primes p' and q' such that $p = 2p' + 1$ and $q = 2q' + 1$ are also primes, and finally sets $n = pq$. We note that, since n is a product of two strong primes, a randomly chosen odd number in Z_n has negligible probability of not being relatively prime to $\phi(n)$. (See Section 3 for further discussion.) Hence, the mapping function \mathcal{KG} can be quite trivial. (Our current implementation uses the popular *MD5* hash function.)

The public exponent e_{Alice} is constructed as the output of $\mathcal{KG}(ID_{Alice})$ represented as a binary string of the same length as the modulus, with the least significant bit set. This ensures that e_{Alice} is odd and, with overwhelming probability, relatively prime to $\phi(n)$. The complete IB-mRSA key generation proceeds as in Figure 1.

A domain- or system-wide certificate ($Cert_{org}$) is issued by the CA after completion of the key generation algorithm. This certificate contains almost all the usual fields normally found in RSA public key certificates with few exceptions, such as no real public key value is given. In particular, it mainly contains the common modulus n and (if applicable) the common part of the email address for all users, such as the domain name.

For the sake of compatibility with other (not identity-based) RSA implementations – including plain RSA and mRSA – the CA may, upon request, issue an individual certificate to a user. In most cases, however, an individual user certificate would not be needed, since not having such certificates is exactly the purpose of identity-based cryptography.

2.2 IB-mRSA Encryption

To encrypt a message, the sender needs only the recipient's email address and the domain certificate. The encryption algorithm is shown in Figure 2.

Since the receiver's public key is derived from the receiver's unique identifier, the sender does not need a public key certificate to ensure that the intended

Algorithm IB-mRSA.enqr

1. Retrieve n, k and \mathcal{KG} algorithm identifier from the domain certificate;
2. $s \leftarrow k - |\mathcal{KG}()| - 1$
3. $e \leftarrow 0^s || \mathcal{KG}(ID_A) || 1$
4. Encrypt input message m with (e, n) using standard RSA/OAEP, as specified in PKCS#1v2.1 [3]

Fig. 2. IB-mRSA: Encryption

receiver is the correct public key holder. Furthermore, fast revocation provided by mRSA obviates the need for the sender to perform any revocation checks. The decryption process is essentially the same as in mRSA. If a certain user needs to be revoked, the domain security administrator merely notifies the appropriate SEM and the revoked user is unable to decrypt any further messages.

2.3 IB-mRSA Decryption

IB-mRSA decryption is identical to that of mRSA. To make this paper self-contained, we borrow (from [9]) the protocol description in Figure 3. For a detailed description and security analysis of additive mRSA, we refer the reader to [9].²

Protocol IB-mRSA.decr (executed by User and SEM)

1. USER: $m' \leftarrow$ encrypted message
2. USER: send m' to SEM
3. In parallel:
 - 3.1 SEM:
 - (a) If USER revoked return (ERROR)
 - (b) $PD_{sem} \leftarrow m'^{d_{sem}} \bmod n$
 - (c) Send PD_{sem} to USER
 - 3.2 USER:
 - (a) $PD_u \leftarrow m'^{d_u} \bmod n$
4. USER: $M \leftarrow (PD_{sem} * PD_u) \bmod n$
5. USER: $m \leftarrow$ OAEP Decoding of M
6. USER: If succeed, return (m)

Fig. 3. IB-mRSA: Decryption

² There is also a very similar multiplicative mRSA (*mRSA) first proposed by Ganesan [13].

3 Security of Identity-based mRSA

We now examine the security of IB-mRSA/OAEP in a setting with n users. All users share a common RSA modulus N and each user (U_i) is associated with a unique identity ID_i , which is mapped into an RSA public exponent e_i via a mapping function \mathcal{KG} .

3.1 Security Analysis

In the following, we argue that if \mathcal{KG} is an appropriate hash function, IB-mRSA/OAEP is semantically secure against adaptive chosen ciphertext attacks (CCA-2) in the random oracle model. We use the term *indistinguishability* which is a notion equivalent to semantic security. (See [6] for the relevant discussion.)

Our analysis is mainly derived from the results in [5], ([4] has similar results) where it was shown that a public-key encryption system in a multi-user setting is semantically secure against certain types of attacks if and only if the same system in a single-user setting is semantically secure against the same attack types.

IB-mRSA/OAEP is obviously an encryption setting with many users, although they do not physically possess their own private keys. To prove semantic security, we begin by asserting that IB-mRSA in single-user mode is equivalent to the standard RSA/OAEP, which is proven secure against CCA-2 under the random oracle [12]. Next, we apply the theorems in [5] with the condition that all users are honest. To remove this condition, we analyze the distribution of views of the system from users and outside adversaries. Furthermore we introduce an additional requirement for the key generation function (division-intractability) so that we can neglect the possibility of an attack from legitimate (inside) users, which is a problem unique to our setting. In the end, we argue for semantic security of IB-mRSA/OAEP.

We use $Succ_1^{IB}(t, q_d)$ to denote the maximum advantage of all adversary algorithms in polynomial time t , attacking IB-mRSA/OAEP with one user, $Succ_n^{IB}(t, q_d, q_e)$ for the setting with n users, and $Succ^R(t, q_d)$ for RSA/OAEP. In the above, $q_d(q_e)$ denote the maximum number of decryption (encryption) queries allowed for each public key. Throughout the analysis, we consider semantic security against CCA-2 under the random oracle assumption. To conserve space, we omit mentioning them in the following discussion.

We begin with the following lemma.

Lemma 1. *IB-mRSA/OAEP system in a single-user setting is polynomially as secure as standard RSA/OAEP encryption, i.e.,*

$$Succ_1^{IB}(t, q_d) = Succ^R(t', q_d)$$

where c is constant value, $t' = t + c$.

The proof is in Appendix A.2. Basically, if there exists an algorithm breaking the security of IB-mRSA/OAEP in a single-user mode, we can build upon it an algorithm breaking standard RSA/OAEP with the same success probability and constant extra overhead. Of course, it is easy to see that breaking RSA/OAEP implies breaking IB-mRSA. Thus, we claim that they are equally secure.

For the multi-user setting, we cannot claim that IB-mRSA with n users is semantically secure by directly applying the security reduction theorem in [5]. The reason is that our system is not a typical case referred in [5]. Sharing a common RSA modulus among many users results in their respective trapdoors not being independent; consequently, there could be attacks among the users. Furthermore, users in IB-mRSA may have the incentive not only to attack other users, but also to attempt to break the underlying protocol so that they can bypass the mandatory security control of the SEM.

However, assuming for the moment, that all users are honest, we can obtain the following lemma derived from [5].

Lemma 2. *IB-mRSA/OAEP system with n users is semantically secure if all n are honest. More precisely,*

$$\text{Succ}_n^{IB}(t_n, q_d, q_e) \leq q_e n \text{Succ}_1^{IB}(t_1, q_d)$$

where $t_1 = t_n + O(\log(q_e n))$

When all users are honest, there are clearly no attacks. Thus, IB-mRSA with multi-user can be considered as an example of encryption system in [5] where each user has an independent trapdoor. We adapt the original proof in [5] in order to claim security against CCA-2 since no user actually knows its own trapdoor in IB-mRSA. See Appendix A.3 for details.

Unfortunately, in a real application, all users cannot be assumed to be trusted. To remove this condition in Lemma 2, we have to examine both the information an inside user can observe and the operations an inside user can perform.

For a given entity (user or set of users) we use an informal term “system view” to refer to the distribution of all inputs, outputs, local state information as well as scripts of interactions with decryption oracles, encryption oracles, and the SEM. The system view for an outside attacker is denoted as:

$$V_1 ::= \text{Pr}\{N, (e_0, \dots, e_n), \Gamma_O, \Gamma_E, \Gamma_D, \Gamma_{SEM}\}$$

while the system view for a set of users is:

$$V_2 ::= \text{Pr}\{N, (e_0, \dots, e_n), \{d_{u_i}\}, \Gamma_O, \Gamma_E, \Gamma_D, \Gamma_{SEM}, \Gamma_{d_{u,n}}\}$$

where $\{d_{u_i}\}$ is the set of user key-shares; $\Gamma_O, \Gamma_E, \Gamma_D$ are three scripts recording all queries/answers to the random oracle, encryption oracles and decryption oracles, respectively; Γ_{SEM} is the script recording all requests/replies between all users and the SEM; $\Gamma_{d_{u,n}}$ is the script recording all n users’ computation on ciphertexts with their own secret key-share d_{u_i} . We claim in Lemma 3, that being an IB-mRSA user does not afford one extra useful information as compared to an outside adversary.

Lemma 3. *Under the adaptive chosen ciphertext attack, the system view of the outside adversary (V_1), is polynomially indistinguishable from the combined system view (V_2) of a set of malicious insiders, in the random oracle model.*

Proof. See Appendix A.4 for details. \square

Thus far, we have shown that insider adversaries do not gain advantages over outsiders in terms of obtaining extra information. However, we also need to consider the privileged operations that an insider can make. In IB-mRSA, each user is allowed to send legitimate decryption queries to its SEM. In conventional proofs, the adversary is not allowed to query its oracle for the challenge ciphertext. However in our case, an inside adversary can manipulate a challenge ciphertext (intended for decryption with d_i) into another ciphertext that can be decrypted with its own key d_j and legally decrypt it with the aid of the SEM.³

We now have to consider the probability of such attacks in our setting. More generally, let e_{a_0}, \dots, e_{a_v} be the set of public keys of v malicious users, and $E_v = \prod_{a_i} e_{a_i}$. They may attempt to use some function f , which takes a challenge $c = m^x \bmod n$ as input and outputs ciphertext $c' = m^{E_v} \bmod n$. We offer the following lemma to address the conditions for the existence of such f .

Lemma 4. *Given two RSA exponents x, y and modulus n , let f be a polynomial time complexity function s.t. $f(m^x) = m^y \bmod n$. Such f exists iff $x|y$.*

Proof. See Appendix A.5 for details. \square

According to Lemma 4, we require negligible probability of obtaining a user's public key which is a factor of the product of a set of others. A similar requirement appears in a signature scheme by Gennaro et al. in [14]. They introduce the notion of division intractability for a hash function. Informally, a hash function H is **Division intractable** if it is infeasible to find distinct (X_1, \dots, X_n, Y) in its domain, such that $H(Y)|\prod_i(H(X_i))$. Denoting $Pr^{div}(H)$ as the probability that H fails to hold this property, we have the following proposition regarding the security of IB-mRSA in a multi-user setting.

Proposition 1. *IB-mRSA/OAEP encryption offers equivalent semantic security to*

RSA/OAEP against adaptive chosen ciphertext attacks in the random oracle model, if the key generation function is division intractable.

In summary, Lemma 3 and Lemma 4 enable us to remove the condition of Lemma 2 where all users are assumed to be honest, by requiring the key generation function to be division intractable. Thus, we can reduce the security of IB-mRSA/OAEP in multi-user setting into single-user, which is as secure as standard RSA/OAEP according to Lemma 1.

³ A simple example is as follows. Suppose $e_i = 3 * e_j$. Then, given $c = m^{e_j} \bmod n$, User U_i can compute $c' = c^3 = m^{e_i} \bmod n$, which can be decrypted by U_i with the help from its SEM. The notion of non-malleability does not capture this attack since it is defined under a single fixed private/public key pair.

3.2 The Public Key Mapping Function

The key generation function \mathcal{KG} in IB-mRSA is a hash function H . To ensure the security of the scheme, H must satisfy the following requirements.

Availability of Public Keys: The output of H should have an overwhelming probability of being relatively prime to $\phi(n)$. Obviously, for the inverse (private key) to exist, a public exponent can not have common factors with $\phi(n)$.

Note that in Section 2 the RSA modulus n is set to $n = p * q$ and p, q are chosen as strong primes $p = 2p' + 1$, $q = 2q' + 1$ where both p' and q' are also large primes. Considering $\phi(n) = 2^2 p' q'$ with only three factors 2, p , q , the probability of the output from H being co-prime to $\phi(n)$ is overwhelming on the condition that the output is an odd number, because finding an odder number not co-prime to $4p'q'$ is equivalent to find p' or q' and consequently factoring n .

Collision Resistance: H should be a collision-resistant function, i.e., given any two distinct inputs ID_1, ID_2 , the probability of $H(ID_1) = H(ID_2)$ should be negligible. In other words, no two users in the domain can share the same public exponent.

Division Resistance: As discussed in Section 3.1, division intractability of H is essential to the security of IB-mRSA. Gennaro et al. analyzed the probability of division for hash functions in [14].

Moreover, Coron and Naccache showed in [11] that the number of necessary hash-value to find a division relation among a hash function's outputs is sub-exponential to its digest size k : $\exp(\sqrt{2\log 2}/2 + o(1)\sqrt{k \log k})$.

They suggested using 1024-bit hash functions to get a security level equivalent to 1024-bits RSA. However, such a strong hash function is not needed in our case. As a point of comparison, the GHR signature scheme [14] needs a division-intractable hash function to compute message digests, where an adaptive adversary can select any number of inputs to the underlying hash function. IB-mRSA needs a hash function to compute digests from users' identities. In any domain, the number of allowed identities is certainly much fewer compared to the number of messages in [14].

To help select the best hash size for our purposes, we quote from the experiments by Coron and Naccache [11] in Table 1. Taking the first line as an example, an interpretation of the data is that, among at least 2^{36} hash digests, the probability of finding one hash value dividing another is non-negligible. In IB-mRSA setting, the typical personnel of an organization is on the order of $2^{10} \sim 2^{17}$.

Table 1. Estimated complexity of the attack for variable digest sizes

digest size in bits	\log_2 complexity (in # of operations)
128	36
160	39
192	42
1024	86

Consequently, the possible number of operations is far less than 2^{36} . Hence, we can safely use MD5 or SHA-1 as the mapping function (H).

3.3 SEM Security

Suppose that the attacker is able to compromise the SEM and expose the secret key d_{sem} , however, without collusion with any user. This only enables the attacker to “un-revoke” previously revoked, or block possible future revocation of currently valid, certificates. The knowledge of d_{sem} does not enable the attacker to decrypt or sign messages on behalf of the users. The reason is obvious: note that Alice never sends her partial results to her SEM. Thus, the attacker’s view of Alice can be simulated in the normal RSA setting, where the attacker just picks a random number as d_{sem} and make computations on the ciphertext, messages to sign and signatures generated by Alice.

3.4 Security of Common Modulus

As mentioned earlier, using a common RSA modulus is clearly unacceptable in plain RSA setting. In the mediated RSA architecture, sharing a modulus is feasible since no party knows a complete private/public key-pair. In fact, no coalition of users is able to compute a public/private key-pair. The only way to “break” the system appears to be by subverting a SEM and colluding with a user. Thus, in the context of IB-mRSA we need to assume that a SEM is a **fully trusted party**, as opposed to **semi-trusted** in mRSA [9].

4 Performance Analysis

When plain RSA is used for encryption, the public encryption exponent e is typically a small integer with only a few 1-bits. One example is the popular OpenSSL toolkit [17] which uses 65,537 as the default public key value for RSA certificates. Encryption with such small exponents can be accelerated with specialized algorithms for modular exponentiation. However, in IB-mRSA setting, there is no such luxury of choosing special exponents and a typical public exponent is a relatively large integer with (on the average) half of the bits set to 1.

Table 2. IB-mRSA encryption: Performance comparison of different encryption keys

Keys	RSA Modulus 1Kb	RSA Modulus 2Kb	RSA Modulus 4Kb
65,537	2 ms	4 ms	12 ms
128-bit key	7 ms	20 ms	69 ms
160-bit key	8 ms	25 ms	88 ms

We ran some simple tests to assess the cost of IB-mRSA encryption for public keys derived from email addresses. The encryption was tested using OpenSSL on an 800MHz PIII workstation. In the tests, we used: 1) “default” encryption exponent 65,537 and 2) two other exponents of length 128-bit and 160-bit. For each key, we randomly set half of the bits. The results are depicted in Table 2.

From the results in Table 2, we see that encryption with a randomized key does introduce overhead, especially when the RSA modulus size grows. However, it is rather negligible for the 1024-bit case, which is currently the most popular modulus size.

The decryption cost for IB-mRSA is identical to mRSA. The performance of mRSA has been reported on by Boneh, et al. in [9]. For example, a 1024-bit mRSA decryption costs around 35ms on an 800 MHz PIII, as compared to 7.5ms for plain RSA on the same platform. We note that this is still much cheaper than 40ms that is needed for Boneh/Franklin IBE decryption (for 1024 bits of security on a even more powerful hardware platform).

5 IB-mRSA versus Boneh/Franklin IBE

We now provide a detailed comparison of BF-IBE and IB-mRSA. The comparison is done along several aspects, including: practicality, revocation, security and cost of key generation.

Practicality and Performance: Although BF-IBE and IB-mRSA have similar architectures, the underlying cryptographic primitives are completely different. Compared to the elliptic curve primitives used in BF-IBE, IB-mRSA is much easier to deploy since RSA is currently the most popular public key encryption method. Recall that IB-mRSA is fully compatible with standard RSA encryption. Moreover, if optional individual certificates are used, IB-mRSA is fully compatible with current PKI-s. Thus, it offers a smooth and natural transition from normal ID-based to public key cryptography.

In addition, IB-mRSA offers better performance than BF-IBE. As seen from the comparison in Table 3, IB-mRSA is noticeably faster than BF-IBE in both key generation and message encryption.

Table 3. Performance comparison of BF-IBE (on PIII 1GHz) and IB-mRSA (on PIII 800MHz) with 1024-bit security

	BF-IBE	IB-mRSA
Private Key Generation	3ms	< 1ms
Encryption Time	40ms	7ms
Decryption Time	40ms	35ms

Revocation: BF-IBE does not explicitly provide revocation of users' security capabilities. This is natural since it aims to avoid the use of certificates in the course of public key encryption. On the other hand, revocation is often necessary and even imperative.

The only way to obtain revocation in normal IBE is to require fine-grained time-dependent public keys, e.g., public keys derived from identifiers combined with time- or date-stamps. This has an unfortunate consequence of having to periodically re-issue all private keys in the system. Moreover, these keys must be (again, periodically) securely distributed to individual users. In contrast, IB-mRSA inherits its fine-grained revocation functionality from mRSA [9]. IB-mRSA provides per-operation revocation, whereas, BF-IBE provides periodic revocation, which clearly has coarser granularity. Essentially, IB-mRSA allows revocation to commence at any time while BF-IBE revokes users by refusing to issue new private keys. However, BF-IBE does not prevent the type of an attack whereby an adversary who compromises a previous or current key can use them to decrypt previously encrypted messages. This can be a serious attack in some settings, such as military applications.

Trusted Third Parties: Both SEM in IB-mRSA and PKG in BF-IBE are trusted third parties. However, the difference in the degree of trust is subtle. A SEM is fully trusted since its collusion with any user can result in a compromise of all other users' secret keys, due to the shared RSA modulus. Nonetheless, a compromise of a SEM alone does not result in a compromise of any users' secret keys. A PKG is a real TTP since it knows all users' secrets, thus, a compromise of a PKG results in a total system break. While a PKG can also be a CA at the same time, a SEM can never be allowed to play the role of CA.

If BF-IBE is used to provide fine-grained revocation, frequent key generation and secure key distribution are expensive procedures. Although a PKG is not required to be on-line all of the time, in practice, it must be constantly available since users do not all request their current private keys at the same time. Therefore, as the revocation interval in BF-IBE gets smaller, the on-line presence of a PKG becomes more necessary.

6 Implementation

We implemented IB-mRSA for the purposes of experimentation and validation. The implementation is publicly available at <http://sconce.ics.uci.edu/sucses>. The software is composed of three parts:

1. CA and Admin Utilities: domain certificate, user key generation, (optional) certificate issuance and revocation interface.
2. SEM daemon: SEM process as described in Section 2.
3. Client libraries: IB-mRSA user functions accessible via an API.

The code is built on top of the popular OpenSSL [17] library. OpenSSL incorporates a multitude of cryptographic functions and large-number arithmetic

primitives. In addition to being efficient and available on many common hardware and software platforms, OpenSSL adheres to the common PKCS standards and is in the public domain. The SEM daemon and the CA/Admin utilities are implemented on Linux, while the client libraries are available on both Linux and Windows platforms.

In the initialization phase, a CA initializes the domain-wide cryptographic setting, namely (n, p, q, p', q') and selects a mapping function (currently defaulting to MD5) for all domain clients. The set up process follows the description in Section 2. For each user, two structures are exported: 1) SEM *bundle*, which includes the SEM's half-key d_i^{SEM} , and 2) user *bundle*, which includes d_i^u and the entire server bundle.

The server bundle is in PKCS#7 [1] format, which is basically a RSA envelope signed by the CA and encrypted with the SEM's public key. The client bundle is in PKCS#12 [2] format, which is a shared-key envelope also signed by the CA and encrypted with the user-supplied key which can be a pre-set key, a password or a pass-phrase. (A user is not assumed to have a pre-existing public key.)

After issuance, each user bundle is distributed in an out-of-band fashion to the appropriate user. Before attempting any IB-mRSA transactions, the user must first decrypt and verify the bundle. A separate utility program is provided for this purpose. With it, the bundle is decrypted with the user-supplied key, the CA's signature is verified, and, finally, the user's half-key are extracted and stored locally.

To decrypt a message, the user starts with sending an IB-mRSA request, with the SEM bundle piggybacked. The SEM first check the status of the client. Only when the client is deemed to be a legitimate user, does the SEM process the request using the bundle contained therein. As mentioned earlier, in order to encrypt a message for an IB-mRSA, that user's domain certificate needs to be obtained. Distribution and management of domain certificates is assumed to be done in a manner similar to that of normal certificate, e.g., via LDAP or DNS.

6.1 Emailer Client Plug-in

To demonstrate the ease of using IB-mRSA we implemented plug-ins for the popular Eudora [19] and Outlook [16] mailers. The plug-ins allow the sender to encrypt outgoing emails to any client in the common domain using only one domain (organizational) certificate. When ready to send, the sender's plug-in reads the recipient's email address and looks up the organization certificate by using the domain name in the email address. A screen snapshot of the Eudora plug-in is shown in Figure 4.

When an email message encrypted with IB-mRSA is received, an icon for IB-mRSA is displayed in the message window. To decrypt the message, the user just clicks on the IB-mRSA icon. The plug-in then contacts the user's SEM to get a partially decrypted message (if the user is not revoked). This is basically the same process as in mRSA.



Fig. 4. Eudora IBE Plug-in

7 Summary and Future Work

We described IB-mRSA, a practical and secure identity-based encryption scheme. It is compatible with standard RSA encryption and offers fine-grained control (revocation) of users security privileges.

Several issues remain for future work. It is unclear whether IB-mRSA can be shown secure under the standard model (our argument utilizes the random oracle setting). Moreover, we need a more formal analysis of semantic security. Another issue relates to IB-mRSA performance. Using a hash function for public key mapping makes encryption more expensive than RSA since the public exponent is random (and on the average half of the bits are set). We need to investigate alternative mapping functions that can produce more “efficient” RSA exponents.

Acknowledgements

Some of the initial ideas for this work emanated from discussions with Dan Boneh whose contribution is gratefully acknowledged. Thanks also go to the anonymous reviewers for their useful comments.

References

- [1] PKCS #7: Cryptographic message syntax standard. Technical note, RSA Laboratories, Nov. 1993. Version 1.5. [205](#)
- [2] PKCS #12: Personal information exchange syntax. Technical note, RSA Laboratories, 1999. Version 1.0. [205](#)
- [3] PKCS#1v2.1: Rsa cryptography standard. Technical note, RSA Laboratories, 2002. Version 2.1. [197](#)

- [4] O. Baudron, D. Pointcheval, and J. Stern. Extended notions of security for multicast public key cryptosystems. In *27th International Colloquium on Automata, Languages and Programming (ICALP '2000)*, number 1853 in Lecture Notes in Computer Science. Springer-Verlag, Berlin Germany, July 2000. 198
- [5] M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Preneel [18], pages 259–274. 198, 199, 209, 210
- [6] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, number 1462 in Lecture Notes in Computer Science, pages 26–45. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1998. 198
- [7] M. Bellare and P. Rogaway. Optimal asymmetric encryption — how to encrypt with RSA. In A. D. Santis, editor, *Advances in Cryptology – EUROCRYPT '94*, number 950 in Lecture Notes in Computer Science, pages 92–111. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1995. 193, 194
- [8] D. Boneh, X. Ding, and G. Tsudik. Identity based encryption using mediated rsa. In *3rd Workshop on Information Security Application*, Jeju Island, Korea, Aug. 2002. KIISC. 194
- [9] D. Boneh, X. Ding, G. Tsudik, and C. M. Wong. A method for fast revocation of public key certificates and security capabilities. In *10th USENIX Security Symposium*, Washington, D. C., Aug. 2001. USENIX. 193, 194, 197, 202, 203, 204
- [10] D. Boneh and M. Franklin. Identity-based encryption from the Weil Pairing. In Kilian [15], pages 213–229. 194
- [11] J.-S. Coron and D. Naccache. Security analysis of the gennaro-halevi-rabin signature scheme. In Preneel [18], pages 91–101. 201
- [12] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. RSA-OAEP is secure under the rsa assumption. In Kilian [15], pages 260–274. 198
- [13] R. Ganesan. Augmenting kerberos with public-key cryptography. In T. Mayfield, editor, *Symposium on Network and Distributed Systems Security*, San Diego, California, Feb. 1995. Internet Society. 197
- [14] R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In J. Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, number 1592 in Lecture Notes in Computer Science, pages 123–139. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1999. 200, 201
- [15] J. Kilian, editor. *Advances in Cryptology – CRYPTO '2001*, number 2139 in Lecture Notes in Computer Science. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 2001. 207
- [16] Microsoft. Microsoft Outlook®, <http://www.microsoft.com>. 205
- [17] OpenSSL User Group. The OpenSSL Project Web Page, <http://www.openssl.org>. 202, 204
- [18] B. Preneel, editor. *Advances in Cryptology – EUROCRYPT '2000*, number 1807 in Lecture Notes in Computer Science, Brugge, Belgium, 2000. Springer-Verlag, Berlin Germany. 207
- [19] Qualcomm. Qualcomm eudora mailer, <http://www.eudora.com>. 205
- [20] A. Shamir. Identity-based cryptosystems and signature schemes. In G. Blakley and D. Chaum, editors, *Advances in Cryptology – CRYPTO '84*, number 196 in

Lecture Notes in Computer Science, pages 47–53. International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, 1985. 194

A Proof of Security

A.1 Notations and Attack Model

Throughout the appendix, we use the following notations

- \mathcal{KG} : Key Generation Function
- $\mathcal{PE}()$: IB-mRSA/OAEP encryption system.
- \mathcal{DO}_{d_i} : Decryption oracle with private key d_i
- \mathcal{RO} : Random oracle
- N : The common RSA modulus
- e_i/d_i : the i -th user's public key/private key
- n : The number of users in \mathcal{PE}
- q_e : The number of encryptions allowed to be performed by each user
- q_d : The maximum number of decryption queries the adversary can ask

Under the notion of indistinguishability of security, the adversary \mathcal{A} takes the public key and outputs two equal length messages m_0, m_1 . Then, it gets a challenge ciphertext C , computed by an encryption oracle which secretly picks $b \in_R \{0, 1\}$ and encrypts m_b . \mathcal{A} is challenged to output b with a probability non-negligibly greater than $1/2$. In CCA attack model, \mathcal{A} is allowed to send queries to a decryption oracle, with the restriction that \mathcal{A} is not allowed to query on the challenge ciphertext c .

A.2 Proof of Lemma 1

Proof. The lemma means that if there exists an attack algorithm \mathcal{B} with polynomial time complexity, breaking the security of IB-RSA/OAEP with success probability ϵ , then there exists an attack algorithm \mathcal{F} with the same polynomial degree of running time, breaking RSA/OAEP with the same success probability; and vice versa.

The reverse direction is obvious. For any \mathcal{F} that can break the indistinguishability of standard RSA, it breaks IB-mRSA in single-user mode. Thus we have $Succ^R(t', q_d) \leq Succ_1^{IB}(t, q_d)$. Now we show $Succ_1^{IB}(t, q_d) \leq Succ^R(t', q_d)$.

Let \mathcal{B} be the polynomial algorithm attacking on the indistinguishability of $\mathcal{PE}(\mathcal{KG}, N, 1)$ containing the single user U_0 and its public key e_0 and secret bundle d_{u_0} . By allowing \mathcal{B} to know d_{u_0} , we model the concern that the user in the system may be malicious. We construct \mathcal{F} as the adversary algorithm against the standard RSA/OAEP (\hat{N}, \hat{e}) and analyze its success probability and time complexity. Replacing \mathcal{KG} function in \mathcal{PE} by a random oracle and acting as the random oracle and decryption oracle for \mathcal{B} , \mathcal{A} runs \mathcal{F} as follows.

Experiment $\mathcal{F}^{RSA}(\hat{N}, \hat{e}, \mathcal{DO}_d, \mathcal{RO})$

Select two random messages (m_0, m_1) with equal length. The encryption oracle EO_e secretly selects a random bit $b \in \mathcal{R} \{0, 1\}$ and encrypts m_b into the ciphertext c . Given c , \mathcal{F} runs the following to determine b .

1. Generate a random number r and a string id ;
2. Initialize $\mathcal{PE}(\mathcal{KG}, N)$ with single-user setting by $N \leftarrow \hat{N}$, For user $ID_0 \leftarrow id$;
3. \mathcal{PE} queries its random oracle (\mathcal{F}) for e_0 ;
4. $e_0 \leftarrow \hat{e}$;
5. Initialize \mathcal{B} with (m_0, m_1, c) and the target system $\mathcal{PE}(\mathcal{KG}, \hat{N})$ and user public key e_0 , user bundle r ;
6. Run \mathcal{B} . The number of decryption queries is bounded by q_d :
 - (a) For all \mathcal{B} 's random oracle queries on OAEP encoding/decoding, \mathcal{F} forwards them to \mathcal{RO} and hands the answers back;
 - (b) For all \mathcal{B} 's decryption oracle queries, \mathcal{F} forwards them to \mathcal{DO}_d , and hands the answers back;
 - (c) For \mathcal{B} 's requests c to SEM (remember that the adversary might be inside the system): \mathcal{F} queries \mathcal{DO}_d on c . On getting the reply $c^d \bmod n$, \mathcal{F} hands back $c^d/c^r \bmod n$ as the reply from SEM to \mathcal{B} .
7. \mathcal{B} halts outputting a bit b' ;
8. Return b' ;

Clearly, if \mathcal{B} 's output b' equals b , \mathcal{F} successfully discovers b . This holds for all polynomial algorithm \mathcal{B} . Thus we have $Succ_1^{IB}(t, q_d) \leq Succ^R(t', q_d)$. As for the time complexity of \mathcal{F} , the steps 1~5 and steps 7,8 take constant time, in that the cost is independent of the security parameter, and step 6 runs in time t . Hence, the overall time for \mathcal{F} is $t + c$, which leads us to the conclusion of $Succ_1^{IB}(t, q_d) = Succ^R(t', q_d)$. \square

A.3 Proof of Lemma 2

Proof. If all users in \mathcal{PE} are considered trusted, we do not need to consider all attacks originated from the inside users. The \mathcal{PE} is therefore a IB-mRSA/OAEP in multi-user setting, whose security for single-mode is proved in Lemma 1.

To show the polynomial reduction, the proof in [5] constructs an attack algorithm \mathcal{B} for single-setting. \mathcal{B} calls another algorithm \mathcal{A} , which can break the multi-user setting. In order to argue the security in CCA2 model, \mathcal{B} has to simulate the decryption oracles for \mathcal{A} . This is simple in their case, where \mathcal{B} can invoke key generation function to obtain all needed public/private key pairs. Unfortunately, this is not the case in IB-mRSA setting, since the key generation will not give \mathcal{B} the private keys. We slightly revise the original proof.

Still, \mathcal{A} targets at a multi-use setting with public keys $\{N, e_0, \dots, e_n\}$. However, we construct \mathcal{B} , targeting at $\{N, e = \prod_{i=0}^n (e_i)\}$. The algorithm for \mathcal{A} 's decryption query is shown below:

Decryption oracle simulator ($\mathcal{B}, \mathcal{A}, N, e_0, \dots, e_n, e = \prod_{i=0}^n e_i$)

\mathcal{B} simulates decryption oracle for \mathcal{A} with the help from its own oracle \mathcal{DO}_d .
(d is the corresponding secret key to (n, e) .)

1. $\mathcal{A} \rightarrow \mathcal{B}$: (c, e_i) ,
2. $\mathcal{B} \rightarrow \mathcal{DO}_d$: c
3. $\mathcal{B} \leftarrow \mathcal{DO}_d$: $c' = c^d \pmod{n}$
4. \mathcal{B} computes $b = \frac{e}{e_i}$
5. $\mathcal{A} \leftarrow \mathcal{B}$: $a = c'^b \pmod{n}$

One can easily check that the answer a is exactly c^{1/e_i} . Thus, the proof for Theorem 4.1 in [5] still holds, which also proves this lemma. \square

A.4 Proof of Lemma 3

Proof (Sketch). No secret channel is assumed either in IB-mRSA protocol execution or in the attack model. Thus, the outsider observes everything that the insider does, except for $\{d_{u_i}\}$ and $\Gamma_{d_u, n}$. However, an outsider can simulate $\Gamma_{d_u, n}$ with the help of a random oracle and the decryption oracles.

Note that a user's key-share is nothing but a random number derived from an idealized hash function, which can be replaced by the random oracle RO . The outsider can query RO and obtain a set of random values $\{r_i\}$ with the same distribution as $\{d_{u_i}\}$. For each ciphertext c in $\Gamma_{d_u, n}$ (encrypted with e_{u_i}) the adversary constructs $\Gamma'_{d_u, n}$ by computing $c^{d_{u_i}} = c^{d_i}/c^{r_i}$, where c^{d_i} is obtained from the decryption oracle \mathcal{DO}_{d_i} . All c, d_{u_i}, r_i are random integers. (Note that c is also random since OAEP encoding is applied before exponentiation). Thus, $Pr\{\Gamma_{d_u, n}\} = Pr\{\Gamma'_{d_u, n}\}$, which leads to V_1 and V_2 having the same distribution. \square

A.5 Proof of Lemma 4

Proof. We show that $x|y$ is a sufficient and necessary condition for the existence of f .

SUFFICIENCY: if $x|y$, i.e. $\exists k \in \mathcal{N}$, s.t. $y = kx$. We construct f as

$$f : a \rightarrow a^k \pmod{n}$$

One can easily check that f is the desired function.

NECESSITY: Suppose there exists a function f satisfying the requirement, while $y = kx + r$, where $k, r \in \mathcal{N}$ and $1 \leq r \leq x$. Given $c = m^x \pmod{n}$, we can compute $c_1 = f(c) = m^y \pmod{n}$. Suppose $g = gcd(x, y)$, i.e. $\exists a, b \in \mathbb{Z}$, s.t. $ax + by = g$. Thus, in polynomial time, we can get m^g by computing $c^a c_1^b \pmod{n}$. If we let $x = hg$, we have actually constructed a polynomial-time algorithm, which, taking $c = (m^g)^h \pmod{n}$ and h, n as input, outputs $c^{1/h} \pmod{n}$ without knowing the factorization of n . (Note that x is relatively prime to $\phi(n)$, which implies that h is also relatively prime to $\phi(n)$ and is a valid RSA public key exponent.) However, this contradicts the RSA assumption. \square

Two Birds One Stone: Signcryption Using RSA

John Malone-Lee¹ and Wenbo Mao^{2 *}

¹ University of Bristol, Department of Computer Science
Merchant Venturers Building, Woodland Road
Bristol, BS8 1UB, UK
malone@cs.bris.ac.uk

² Hewlett-Packard Laboratories
Filton Road, Stoke Gifford, Bristol, BS34 8QZ, UK
wm@hplb.hpl.hp.com

Abstract. Signcryption is a public key primitive proposed by Zheng [14] to achieve the combined functionality of digital signature and encryption in an efficient manner. We present a signcryption scheme based on RSA and provide proofs of security in the random oracle model [6] for its privacy and unforgeability. Both proofs are under the assumption that inverting the RSA function is hard.

Our scheme has two appealing aspects to it. First of all it produces compact ciphertexts. Secondly it offers non-repudiation in a very straightforward manner.

1 Introduction

Signcryption is a novel public key primitive first proposed by Zheng in 1997 [14]. A signcryption scheme combines the functionality of a digital signature scheme with that of an encryption scheme. It therefore offers the three services: privacy, authenticity and non-repudiation. Since these services are frequently required simultaneously, Zheng proposed signcryption as a means to offer them in a more efficient manner than a straightforward composition of digital signature scheme and encryption scheme. An ingenious scheme was proposed to meet such a goal.

It is only recently that research has been done on defining security for signcryption and providing security arguments for schemes [2, 3]. In [3] a scheme similar to the original one proposed in [14] is analysed. The model in [2] is slightly different. It aims to analyse any primitive that achieves the combined functionality of signature and encryption.

Here we continue this line of research into provable security of signcryption schemes. We present a signcryption scheme based on the RSA trapdoor one-way function. An attractive feature of our scheme is that it offers non-repudiation in a very simple manner. Non-repudiation for signcryption is not a straightforward consequence of unforgeability like it is for digital signature schemes. The reason for this is that a signcrypted message is “encrypted” as well as “signed”.

* This author’s research is partially funded by the EU Fifth Framework Project IST-2001-324467 ”CASENET”.

Therefore, by default, only the intended receiver of a signcryption may verify its authenticity. If a third party is to settle a repudiation dispute over a signcryption, it must have access to some information in addition to the signcryption itself. Of course the receiver could always surrender its private key but this is clearly unsatisfactory. It is often the case that several rounds of zero-knowledge are required. This is not the case for our scheme.

The scheme uses a padding scheme similar to PSS [7, 8]. The PSS padding scheme was originally designed to create a provably secure signature algorithm when used with RSA [7]. It was subsequently pointed out in [8] that a version of PSS could also be combined with RSA to create a provably secure encryption function. As demonstrated here, this makes PSS padding perfect for RSA based signcryption. The resulting scheme is very efficient in terms of bandwidth: a signcryption is half the size of a message signed and encrypted using standard techniques for RSA. For this reason we give it the name Two Birds One Stone.

We envisage that our scheme could be used in an e-commerce scenario such as signencrypting a bankcard payment authorisation. Here one RSA block suffices and, as we have discussed, the scheme offers non-repudiation which is clearly desirable for such an application. An alternative use could be signcryption of session keys in a key transport protocol.

2 Two Birds One Stone (TBOS)

2.1 Abstract TBOS

The TBOS cryptosystem will make use of what we will call a *permutation with trapdoors*. A permutation with trapdoors $f : \{0, 1\}^k \rightarrow \{0, 1\}^k$ is a function that requires some secret, or “trapdoor”, information to evaluate and some different secret information to invert. In the scheme below we will assume that the sender of messages, Alice, knows the secret information necessary to evaluate f , and the receiver, Bob, knows the secret information necessary to evaluate f^{-1} .

The scheme may be used to signcrypt messages from $\{0, 1\}^n$, where $k = n + k_0 + k_1$ for integers k_0 and k_1 . Before f is applied to a message some random padding is applied. The padding used is similar to PSS [7, 8]. We describe how the scheme works below.

Parameters

The scheme requires two hash functions

$$H : \{0, 1\}^{n+k_0} \rightarrow \{0, 1\}^{k_1} \text{ and } G : \{0, 1\}^{k_1} \rightarrow \{0, 1\}^{n+k_0}.$$

Signcryption

For Alice to signcrypt a message $m \in \{0, 1\}^n$ for Bob:

1. $r \xleftarrow{r} \{0, 1\}^{k_0}$
2. $\omega \leftarrow H(m||r)$
3. $s \leftarrow G(\omega) \oplus (m||r)$
4. $c \leftarrow f(s||\omega)$
5. Send c to Bob

Unsigncryption

For Bob to unsigncrypt a cryptogram c from Alice:

1. $s||\omega \leftarrow f^{-1}(c)$
2. $m||r \leftarrow G(\omega) \oplus s$
3. If $H(m||r) = \omega$ accept m
Else reject

As it stands TBOS has no obvious way to provide non-repudiation. We discuss how this problem may be overcome in the next section.

2.2 RSA-TBOS

We now show how RSA is used to create something like a permutation with trapdoors, as in Section 2.1, for use with TBOS. We are not claiming that the resulting function is a permutation. This is not necessary for our proof of security.

To begin with a sender Alice generates an RSA key pair $(N_A, e_A), (N_A, d_A)$, with $N_A = P_A \cdot Q_A$ and $|P_A| = |Q_A| = k/2$. Here and henceforth k is an even positive integer. A receiver Bob does likewise giving him $(N_B, e_B), (N_B, d_B)$. Using G and H as above we describe the scheme below. Here, if a bit string $\alpha||\beta$ represents an integer, then α represents the most significant bits of that integer.

Signcryption

For Alice to signcrypt a message $m \in \{0, 1\}^n$ for Bob:

1. $r \xleftarrow{r} \{0, 1\}^{k_0}$
2. $\omega \leftarrow H(m||r)$
3. $s \leftarrow G(\omega) \oplus (m||r)$
4. If $s||\omega > N_A$ goto 1
5. $c' \leftarrow (s||\omega)^{d_A} \pmod{N_A}$
6. If $c' > N_B$, $c' \leftarrow c' - 2^{k-1}$
7. $c \leftarrow c'^{e_B} \pmod{N_B}$
8. Send c to Bob

Unsigncryption

For Bob to unsigncrypt a cryptogram c from Alice:

1. $c' \leftarrow c^{d_B} \pmod{N_B}$
2. If $c' > N_A$, reject
3. $\mu \leftarrow c'^{e_A} \pmod{N_A}$
4. Parse μ as $s||\omega$
5. $m||r \leftarrow G(\omega) \oplus s$
6. If $H(m||r) = \omega$, return m
7. $c' \leftarrow c' + 2^{k-1}$
8. If $c' > N_A$, reject
9. $\mu \leftarrow c'^{e_A} \pmod{N_A}$
10. Parse μ as $s||\omega$
11. $m||r \leftarrow G(\omega) \oplus s$
12. If $\omega \neq H(m||r)$, reject
13. Return m

The point of step 6 in signcryption is to ensure that $c' < N_B$. If c' initially fails this test then we have $N_A > c' > N_B$. Since both N_A and N_B have k -bits we infer that c' also has k -bits and so the assignment $c' \leftarrow c' - 2^{k-1}$ is equivalent to removing the most significant bit of c' . This gives us $c' < N_B$ as required. Note that this step may cause an additional step in unsigncryption. In particular it may be necessary to perform $c'^{e_A} \pmod{N_A}$ twice (the two c' 's will differ by 2^{k-1}). It would have been possible to define an alternative scheme under which the trial and error occurs in signcryption. This would mean repeating steps 1-5 in signcryption with different values of r until $c' < N_B$ was obtained.

Non-repudiation is very simple for RSA-TBOS. The receiver of a signcryption follows the unsigncryption procedure up until stage 2, c' may then be given to a third party who can verify its validity.

3 Security Notions for Signcryption Schemes

3.1 IND-CCA2 for Signcryption Schemes

We take as our starting point the standard definition of indistinguishability of encryptions under adaptive chosen ciphertext attack (IND-CCA2) for public key encryption schemes [1, 4, 5, 10, 11]. A public key encryption scheme enjoys IND-CCA2 security if it is not possible for an adversary to distinguish the encryptions of two messages of its choice under a particular public key, even when it has access to a decryption oracle for this public key. The adversary is able to query the decryption oracle before choosing its two messages and its queries may be determined given information gleaned from previous queries. The adversary is then given the challenge ciphertext i.e. the encryption under the public key in question of one of the two messages chosen at random. It is allowed to continue to query the decryption oracle subject to the condition that it does not query the challenge ciphertext itself. The adversary wins if it correctly guesses which of the two messages was encrypted.

In our definition of IND-CCA2 security for signcryption we allow the adversary access to an unsigncryption oracle for the target receiver's key in a similar manner to that described above for encryption schemes. The difference here is that an oracle for the target receiver's unsigncryption algorithm must be defined with respect to some sender's public key. We therefore consider an attack on two users: a sender and a receiver.

In the case of public key encryption schemes the adversary is able to encrypt any messages that it likes under the public key that it is attacking. This is not the case for signcryption schemes. The private key of the target sender is required in signcryption and so the adversary is not able to produce signcryptions on its own. We must therefore provide the adversary with a signcryption oracle for the keys of the target sender and the target receiver. For an encryption scheme the adversary is able to use its own choice of randomness to generate encryptions, we therefore allow the adversary to choose the randomness used by the signcryption oracle, except for challenge ciphertext generation.

We give a more concrete description of the attack below.

Setup

Using the global systems parameters two private/public key pairs (x_A, Y_A) and (x_B, Y_B) are generated for a target sender/receiver respectively.

Find

The adversary is given Y_A and Y_B , it is also given access to two oracles: a signcryption oracle for Y_A, Y_B and an unsigncryption oracle for Y_A, Y_B . The adversary is allowed to choose the random input as well as the message for the signcryption oracle. At the end of this phase the adversary outputs two messages m_0 and m_1 with $|m_0| = |m_1|$.

Challenge

A bit b is chosen uniformly at random. The message m_b is signcrypted under Y_A, Y_B to produce c^* which is given to the adversary.

Guess

The adversary may continue to query its oracles subject to the condition that it does not query its unsigncryption oracle with c^* . At the end of this phase the adversary outputs a bit b' . The adversary wins if $b' = b$.

If \mathcal{A} is an adversary as described above we define its advantage as:

$$\mathbf{Adv}(\mathcal{A}) = |2 \cdot \Pr[b' = b] - 1|.$$

We say that a signcryption scheme is IND-CCA2 secure if the advantage of any polynomial-time adversary is a negligible¹ function of the security parameter of the scheme.

3.2 Unforgeability of Signcryption Schemes

We adapt the definition of existential unforgeability under adaptive chosen message attack [13] for signature schemes to the signcryption setting.

When using a signature scheme, the only private key used in signature generation belongs to the sender. An adversary can therefore be anyone, since there is no difference in the ability to forge signatures between a receiver of signed messages and a third party. For a signcryption scheme however, signature generation uses the receiver's public key as well as the sender's keys. In this instance there may be a difference in the ability to forge signcryptions between the receiver and a third party, since only the receiver knows the private key corresponding to its public key. With the above in mind we assume that an adversary has access to the private key of the receiver as well as the public key of the sender. It can therefore perform unsigncryption itself.

We allow an adversary to query a signcryption oracle for the target sender's private key. This oracle takes as input a message, and an arbitrary public key chosen by the adversary. The oracle returns the signcryption of the message under the target sender's key and the key chosen by the adversary.

We say that the adversary wins if it produces a valid forged signcryption on some message under the target sender's public key. This message must not have been queried to the signcryption oracle during the attack.

If \mathcal{A} is an adversary as described above we define its advantage as:

$$\mathbf{Adv}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}].$$

We say that a signcryption scheme is existentially unforgeable under adaptive chosen message attack if the advantage of any polynomial-time adversary is a negligible function of the security parameter of the scheme.

4 IND-CCA2 Security of TBOS

4.1 The Underlying Hard Problem

If the secret information necessary to evaluate a permutation with trapdoors f is made public, then f becomes a standard *trapdoor one-way permutation*. We

¹ A function $\epsilon(k)$ is *negligible* if for every c there exists a k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$.

call this the *induced trapdoor one-way permutation* of f . First of all we consider the security of TBOS under the *partial-domain one-wayness* [12] of the induced trapdoor one-way permutation of f . Let us first state formally the definitions that we will use. Below f will be a trapdoor one-way permutation.

Definition 1 (One-Wayness). *The function f is (t, ϵ) -partial domain one-way if the success probability of any adversary \mathcal{A} wishing to recover the preimage of $f(s||\omega)$ in time less than t is upper bounded by ϵ . We state this as:*

$$\mathbf{Adv}_f^{ow}(\mathcal{A}) \leq \Pr_{s,\omega}[\mathcal{A}(f(s||\omega)) = s||\omega] < \epsilon.$$

For any f we denote the maximum value of $\mathbf{Adv}_f^{ow}(\mathcal{A})$ over all adversaries running for time t as $\mathbf{Adv}_f^{ow}(t)$.

Definition 2 (Partial-Domain One-Wayness). *The function f is (t, ϵ) -partial domain one-way if the success probability of any adversary \mathcal{A} wishing to recover the partial preimage of $f(s||\omega)$ in time less than t is upper bounded by ϵ . We state this as:*

$$\mathbf{Adv}_f^{pd-ow}(\mathcal{A}) \leq \Pr_{s,\omega}[\mathcal{A}(f(s||\omega)) = \omega] < \epsilon.$$

For any f we denote the maximum value of $\mathbf{Adv}_f^{pd-ow}(\mathcal{A})$ over all adversaries running for time t as $\mathbf{Adv}_f^{pd-ow}(t)$.

Definition 3 (Set Partial-Domain One-Wayness). *The function f is (l, t, ϵ) -set partial domain one-way if the success probability of any adversary \mathcal{A} wishing to output a set of l elements which contains the partial preimage of $f(s||\omega)$ in time less than t is upper bounded by ϵ . We state this as:*

$$\mathbf{Adv}_f^{s-pd-ow}(\mathcal{A}) \leq \Pr_{s,\omega}[\omega \in \mathcal{A}(f(s||\omega))] < \epsilon.$$

For any f and l we denote the maximum value of $\mathbf{Adv}_f^{s-pd-ow}(\mathcal{A})$ over all adversaries running for time t as $\mathbf{Adv}_f^{s-pd-ow}(l, t)$.

Suppose that an adversary is given c and successfully returns a set of l elements of which one is ω such that $f(s||\omega) = c$ for some s . It is now possible to break the partial-domain one-wayness of f by selecting one of these elements at random. This tells us that

$$\mathbf{Adv}_f^{pd-ow}(t) \geq \mathbf{Adv}_f^{s-pd-ow}(l, t)/l. \quad (1)$$

4.2 IND-CCA2 Security of Abstract TBOS

Theorem 1. *Let \mathcal{A} be an adversary using a CCA2 attack to break TBOS (as defined in Section 2.1). Suppose that \mathcal{A} has advantage ϵ after running for time t , making at most q_g , q_h , q_s and q_u queries to G , H , the signcryption oracle and the*

unsigncryption oracle respectively. Suppose that TBOS is implemented with k -bit permutation with trapdoors f and let f' be the induced trapdoor one-way permutation of f . We have the following

$$\mathbf{Adv}_{f'}^{pd-ow}(t') \geq \frac{1}{q_g + q_h + q_s} \cdot (\epsilon - 2^{-k_0} \cdot (q_h + q_s) - 2^{-k_1} \cdot q_u)$$

where $t' = t_g \cdot (q_g + q_h + q_s) + t_h \cdot (q_h + q_s) + t_s \cdot q_s + t_u \cdot q_u$, q_g is the time taken to simulate the random oracle G (in the proof of Lemma 1 below) and t_h, t_s and t_u are defined analogously.

This follows from (1) and the following lemma.

Lemma 1. *Using the notation of Theorem 1 we have*

$$\mathbf{Adv}_{f'}^{s-pd-ow}(q_g + q_h + q_s, t') \geq \epsilon - 2^{-k_0} \cdot (q_h + q_s) - 2^{-k_1} \cdot q_u.$$

Proof. We will show how the adversary \mathcal{A} may be used to break the set-partial domain one-wayness of f' by finding the partial preimage of c^* chosen at random from the range of f' . Note that the adversary does not know the secret information necessary to evaluate f . The proof is similar to the corresponding proof in [8].

We will consider an attack on two users Alice, the target sender who knows how to evaluate f , and Bob, the target receiver who knows how to evaluate f^{-1} . We run adversary \mathcal{A} on input of all universal public parameters and the public keys of Alice and Bob. It is necessary to show how to respond to \mathcal{A} 's queries to the random oracles G and H and the signcryption/unsigncryption oracles. We denote the algorithms to do this as $G_{sim}, H_{sim}, S_{sim}$ and U_{sim} respectively and we describe them below. To make our simulations sound we keep two lists, L_G and L_H that are initially empty. The list L_G will consist of query/response pairs to the random oracle G . The list L_H will do the same for H . It will also store some extra information as described in H_{sim} below. At the end of the simulation we hope to find the partial preimage of c^* among the queries in L_G .

$G_{sim}(\omega)$	$H_{sim}(m r)$
If $(\omega, x) \in L_G$ for some x :	If $(m r, \omega, c) \in L_H$ for some ω :
Return x	Return ω
Else:	Else:
$x \xleftarrow{r} \{0, 1\}^{n+k_0}$	$\omega \xleftarrow{r} \{0, 1\}^{k_0}$
Add (ω, x) to L_G	$x \leftarrow G_{sim}(\omega)$
Return x	$s \leftarrow x \oplus (m r)$
	Add $(m r, \omega, f(s \omega))$ to L_H
	Return ω
$S_{sim}(m r)$	$U_{sim}(c)$
Run $H_{sim}(m r)$	If $(m r, \omega, c) \in L_H$ for some m :
Search L_H for entry $(m r, \omega, c)$	Return m
Return c	Else reject

Note that in H_{sim} above we assume that each query has form $m||r$. All this means is each query has length $n + k_0$ bits and so may be parsed as $m||r$ where m has n bits and r has k_0 bits. We make this assumption because, in the random oracle model, it would not help \mathcal{A} to make queries of length different from $n + k_0$.

We also allow \mathcal{A} to make queries of the form $m||r$ to S_{sim} i.e. we allow \mathcal{A} to provide its own random input. This is consistent with a CCA2 attack on an encryption scheme such as RSA-PSS where an adversary can encrypt messages itself using its own random input.

At the end of the find stage \mathcal{A} outputs m_0 and m_1 . We choose a bit b uniformly at random and supply the adversary with c^* as the signcryption of m_b . Suppose $c^* = f(s^*||\omega^*)$, this places the following constraints on the random oracles G and H :

$$H(m_b||r^*) = \omega^* \text{ and } G(\omega^*) = s^* \oplus (m_b||r^*). \quad (2)$$

We denote by AskG the event that during \mathcal{A} 's attack ω^* has ended up in L_G . We denote by AskH the event the query $m||r^*$ has ended up in L_H for some m .

If $\omega^* \notin L_G$, then $G(\omega^*)$ is undefined and so r^* is a uniformly distributed random variable. Therefore the probability that there exists an m such that $m||r^* \in L_H$ is at most $2^{-k_0} \cdot (q_h + q_s)$. This tells us that

$$\Pr[\text{AskH} | \neg \text{AskG}] \leq 2^{-k_0} \cdot (q_h + q_s). \quad (3)$$

Our simulation U_{sim} can only fail if it outputs reject when it is presented with a valid ciphertext. We denote this event UBad . Suppose that U_{sim} is queried with $c = f(s||\omega)$ and let

$$m||r = G(\omega) \oplus s.$$

We may mistakenly reject a valid ciphertext if $H(m||r) = \omega$, while $m||r$ is not in L_H . Suppose that this query occurs before c^* is given to \mathcal{A} then, since $m||r$ is not in L_H , $H(m||r)$ will take its value at random. If this query is made after c^* is given to \mathcal{A} then $c \neq c^*$ means that $(m, r) \neq (m_b, r^*)$ and so (2) is irrelevant. In either case $H(m||r)$ may take its value at random which means that

$$\Pr[\text{UBad}] \leq 2^{-k_1} \cdot q_u. \quad (4)$$

Let us define the event Bad as

$$\text{Bad} = \text{AskG} \vee \text{AskH} \vee \text{UBad}. \quad (5)$$

Let us denote the event that the adversary wins, i.e. it outputs b' such that $b' = b$, by \mathbf{S} . In the event $\neg \text{Bad}$ the bit b is independent of our simulations, and therefore independent of the adversaries view. We infer from this that

$$\Pr[\mathbf{S} | \neg \text{Bad}] = \frac{1}{2}. \quad (6)$$

Also, in the event $\neg \text{Bad}$, the adversary interacts with a perfect simulation of random oracles and signcryption/unsigncryption oracles. This gives us

$$\Pr[\mathbf{S} \wedge \neg \text{Bad}] \geq \frac{1}{2} + \frac{\epsilon}{2} - \Pr[\text{Bad}]. \quad (7)$$

From (6) we obtain

$$\Pr[S \wedge \neg \text{Bad}] = \Pr[S | \neg \text{Bad}] \cdot \Pr[\neg \text{Bad}] = \frac{1}{2} \cdot (1 - \Pr[\text{Bad}]). \quad (8)$$

Combining (7) with (8) gives us

$$\Pr[\text{Bad}] \geq \epsilon. \quad (9)$$

From (5) we have

$$\begin{aligned} \Pr[\text{Bad}] &\leq \Pr[\text{AskG} \vee \text{AskH}] + \Pr[\text{UBad}] \\ &= \Pr[\text{AskG}] + \Pr[\text{AskH} \wedge \neg \text{AskG}] + \Pr[\text{UBad}] \\ &\leq \Pr[\text{AskG}] + \Pr[\text{AskH} | \neg \text{AskG}] + \Pr[\text{UBad}]. \end{aligned} \quad (10)$$

Together (3), (4) and (10) give us

$$\Pr[\text{AskG}] \geq \epsilon - 2^{-k_0} \cdot (q_h + q_s) - 2^{-k_1} \cdot q_u. \quad (11)$$

The result follows.

4.3 IND-CCA2 Security of RSA-TBOS

We now adapt the result of Section 4.2 to give a proof of the IND-CCA2 security of RSA-TBOS (as defined in Section 2.2) in the random oracle model under the assumption that the RSA function is one-way.

As in Lemma 1 we will assume that there is an adversary \mathcal{A} that runs for time t and has advantage ϵ in breaking the IND-CCA2 security of RSA-TBOS after making at most q_g , q_h , q_s and q_u queries to G , H , the signcryption oracle and the unsigncryption oracle respectively. Given an RSA public key (N_B, e_B) , with $N_B = P_B \cdot Q_B$ and $|P_B| = |Q_B| = k/2$, and c^* , we will show how \mathcal{A} may be used to compute the e_B -th root of c^* modulo N_B .

The first step is to generate an RSA key pair (N_A, e_A) , (N_A, d_A) with $N_A = P_A \cdot Q_A$ where $|P_A| = |Q_A| = k/2$. We use G_{sim} , S_{sim} and U_{sim} from Lemma 1, we replace H_{sim} with the algorithm below.

```

 $H_{sim}(m||r)$ 
  If  $(m||r, \omega, c) \in L_H$  for some  $\omega$ , return  $\omega$ 
  Else:
    1.  $\omega \xleftarrow{r} \{0, 1\}^{k_0}$ 
    2.  $x \leftarrow G_{sim}(\omega)$ 
    3.  $s \leftarrow x \oplus (m||r)$ 
    4. If  $s||\omega > N_A$ , goto 1
    5.  $c' \leftarrow (s||\omega)^{d_A} \bmod N_A$ 
    6. If  $c' > N_B$ ,  $c' \leftarrow c' - 2^{k-1}$ 
    7.  $c \leftarrow c'^{e_B} \bmod N_B$ 
    8. Add  $(m||r, \omega, c)$  to  $L_H$ 
    9. Return  $\omega$ 

```

The event **Bad** is defined as in (5) in the proof of Lemma 1. In our simulation here we are again going to supply \mathcal{A} with c^* as the challenge ciphertext. This gives us an extra consideration in our simulation. We say that our simulation is **Good** if (i) $c^{*d_B} \bmod N_B < N_A$ and (ii) $\gcd(c^{*d_B} \bmod N_B, N_A) = 1$. Over the random choices of (N_B, e_B) , (N_B, d_B) , c^* and N_A we have $\Pr[(\text{i})] = 1/2$ and $\Pr[(\text{ii})|(\text{i})] \geq 1 - 2^{-(k/2)+(3/2)}$, hence

$$\Pr[\text{Good}] \geq (2^{-1} - 2^{-\frac{k}{2} + \frac{1}{2}}). \quad (12)$$

Consider (4) in the proof of Lemma 1 for Abstract TBOS. For RSA-TBOS there are two possibilities for a ciphertext to be valid and so we have

$$\Pr[\text{UBad}] \leq 2^{-(k_1-1)} \cdot q_u. \quad (13)$$

We may now use a similar argument as that used to derive (11) in the proof of Lemma 1 to give us

$$\Pr[\text{AskG}|\text{Good}] \geq \epsilon - 2^{-k_0} \cdot (q_h + q_s) - 2^{-(k_1-1)} \cdot q_u \quad (14)$$

in our new simulation. We are interested in the event $\text{AskG} \wedge \text{Good}$. We have

$$\Pr[\text{AskG} \wedge \text{Good}] = \Pr[\text{AskG}|\text{Good}] \cdot \Pr[\text{Good}]. \quad (15)$$

Together (12), (14) and (15) tell us

$$\Pr[\text{AskG} \wedge \text{Good}] \geq (2^{-1} - 2^{-\frac{k}{2} + \frac{1}{2}}) \cdot (\epsilon - 2^{-k_0} \cdot (q_h + q_s) - 2^{-(k_1-1)} \cdot q_u) = \delta. \quad (16)$$

Now, in the event $\text{AskG} \wedge \text{Good}$ we recover a set L_G of size

$$q = q_g + q_h + q_s, \quad (17)$$

containing the k_1 least significant bits of z_0^* where $(z_0^{*d_A} \bmod N_A)^{e_B} \bmod N_B = c^*$. Call these bits ω_0 .

Once we have run our simulation once with challenge ciphertext c^* and obtained L_G we do the following:

For $i = 1, \dots, \nu - 1$:

$$\begin{aligned} \alpha_i &\xleftarrow{r} \mathbb{Z}_{N_B}^* \\ c_i^* &\xleftarrow{r} c^* \cdot \alpha_i^{e_B} \bmod N_B \end{aligned}$$

Run the simulation with challenge ciphertext c_i^*
keeping a list L_{G_i} for G query/response pairs

For $i = 1, \dots, \nu - 1$ after each run we end up with a list L_{G_i} of size q containing the k_1 least significant bits of $z_0^* \cdot \beta_i \pmod{N_A}$ where $\beta_i = \alpha_i^{e_A} \pmod{N_A}$ with probability at least that of AskG \wedge Good as given in (16). Now, if each of the ν runs of our simulation were successful, we have $\omega_0 \in L_G, \omega_1 \in L_{G_1}, \dots, \omega_{\nu-1} \in L_{G_{\nu-1}}$ such that

$$\begin{aligned} z_0^* &= \omega_0 + 2^{k_1} \cdot x_0 \pmod{N_A} \\ \beta_i \cdot z_0^* &= \omega_i + 2^{k_1} \cdot x_i \pmod{N_A} \text{ for } i = 1, \dots, \nu - 1 \end{aligned} \quad (18)$$

where z_0^* and x_0, \dots, x_ν are unknown. Now, for $i = 1, \dots, \nu - 1$ let

$$\gamma_i = 2^{-k_1} \cdot (\beta_i \omega_0 - \omega_i) \pmod{N_A}. \quad (19)$$

From (18) and (19) we derive the following for $i = 1, \dots, \nu - 1$

$$x_i - \beta_i \cdot x_0 = \gamma_i \pmod{N_A}. \quad (20)$$

We have the following lemma from [9].

Lemma 2. Suppose $2^{k-1} \leq N_A < 2^k$, $k_1 > 64$ and $k/(k_1)^2 \leq 2^{-6}$. If the set of equations (20) has a solution $\mathbf{x} = (x_0, \dots, x_{\nu-1})$ such that $\|\mathbf{x}\|_\infty < 2^{k-k_1}$, then for all values of $\beta = (\beta_1, \dots, \beta_{\nu-1})$, except for a fraction

$$\frac{2^{\nu \cdot (k-k_1+\nu+2)}}{N_A^{\nu-1}} \quad (21)$$

of them, this solution is unique and can be computed in time polynomial in ν and in the size of N_A .

It is also shown in [8] that taking $\nu = \lceil (5k)/(4k_1) \rceil$ gives

$$\frac{2^{\nu \cdot (k-k_1+\nu+2)}}{N_A^{\nu-1}} \leq 2^{-k/8}. \quad (22)$$

If we have ν successful runs of our simulation we still do not know which elements of the L_G 's form the equations (20) and so to use this method we will have to apply the Lemma 2 algorithm q^ν times. Once we have a solution to (20) we know z_0^* such that $c^* = ((z_0^{*d_A} \pmod{N_A})^{e_B} \pmod{N_B})$. From this we may use d_A to compute z^* , the e_B -th root of c^* , as

$$z^* = z_0^{*d_A} \pmod{N_A}. \quad (23)$$

Now, from (16), (20), (22), (23) and Lemma 2 we obtain the result below.

Theorem 2. Let \mathcal{A} be an adversary that uses a CCA2 attack to attempt to break RSA-TBOS with security parameter k . Suppose that \mathcal{A} succeeds with probability ϵ in time t after making at most q_g, q_h, q_s and q_u queries to G, H , the signcryption oracle and the unsigncryption oracle respectively. In the random oracle model for G and H we may use \mathcal{A} to invert RSA with probability ϵ' in time t' where

$$\begin{aligned} \epsilon' &\geq \delta^\nu - 2^{-k/8}, \\ t' &\leq \nu \cdot t + (q_g + q_h + q_s)^\nu \cdot \text{poly}(k) + 2 \cdot \nu \cdot (q_h + q_s) \cdot T, \end{aligned}$$

$\nu = \lceil (5k)/(4k_1) \rceil$, and T is the time it takes for a modular exponentiation.

Note that as is the case in the proofs of security for RSA-OAEP [12], and PSS with standard RSA [8], our reduction is far from tight. Consequently, for the proof of security to be meaningful, we recommend using 2048-bit RSA moduli.

5 Unforgeability of RSA-TBOS

Before we give our security result we must discuss exactly what constitutes a forged RSA-TBOS signcryption. Suppose that we have a user of RSA-TBOS with public key (N_B, e_B) . This user can produce a random $c \in \mathbb{Z}_{N_B}^*$ and claim to have forged a signcryption from user who owns key (N_A, e_A) . Without knowing (N_B, d_B) it would not be possible to verify this claim. A forged signcryption by the owner of (N_B, d_B) must therefore be presented by following the unsigncryption procedure up until stage 2, c' may then be given to a third party who can verify its validity.

Let us suppose that we have an RSA public key (N_A, e_A) and $c \in \mathbb{Z}_{N_A}^*$ whose e_A -th root we wish to compute. We show in the appendix how to use \mathcal{A} , a forging adversary of RSA-TBOS, to do this. This gives the result below.

Theorem 3. *Let \mathcal{A} be an adversary attempting to forge RSA-TBOS signcryptions. Let k be the security parameter of RSA-TBOS. Suppose that \mathcal{A} succeeds with probability ϵ in time t after making at most q_g , q_h and q_s queries to G , H and the signcryption oracle respectively. In the random oracle model we may use \mathcal{A} to invert RSA with probability ϵ' in time t' where*

$$\begin{aligned} \epsilon' &\geq \epsilon - q_s \cdot \left(2^{-(k_0+1)} \cdot (2q_h + q_s - 1) + 2^{-(k_1+1)} \cdot (2q_g + 2q_h + q_s - 1) \right) \\ &\quad - 2^{-(k_1+1)} \cdot q_h \cdot (2q_g + q_h + 2q_s - 1), \\ t' &\leq t + (q_h + 2q_s) \cdot T, \end{aligned} \tag{24}$$

where T is the time it takes for a modular exponentiation.

6 Conclusion

We have proposed provably secure signcryption scheme based on the RSA function. This scheme is attractive in that it produces very compact signcryptions with little extra computational cost. Also, our scheme offers non-repudiation in a very simple manner.

In the future it would be interesting to adapt these ideas to produce a scheme that is provably secure under the stronger definitions of security proposed for signcryption in [3]. It is also important to investigate the possibility of a padding scheme for which there exists a tighter security reduction.

Acknowledgements

Thanks to Nigel Smart for pointing out a good acronym for our scheme and to David Soldner for discussion on an early draft of this paper.

References

- [1] M. Abdalla, M. Bellare and P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES. In Topics in Cryptology - CT-RSA 2001, volume 2020 of Lecture Notes in Computer Science, pages 143-158. Springer-Verlag, 2001. [214](#)
- [2] J. H. An and Y. Dodis and T. Rabin. On the Security of Joint Signature and Encryption, In Advances in Cryptology - EUROCRYPT 2002, volume 2332 of Lecture Notes in Computer Science, pages 83-107. Springer-Verlag, 2002. [211](#)
- [3] J. Baek, R. Steinfeld and Y. Zheng. Formal Proofs for the Security of Signcryption. In Public Key Cryptography 2002, volume 2274 of Lecture Notes in Computer Science, pages 80-98. Springer-Verlag, 2002. [211](#), [222](#)
- [4] M. Bellare, A. Desai, D. Pointcheval and P. Rogaway. Relations Among Notions of Security for Public-Key Encryption Schemes. In Advances in Cryptology - CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 26-45. Springer-Verlag, 1998. [214](#)
- [5] M. Bellare and P. Rogaway. Optimal Asymmetric Encryption - How to Encrypt with RSA. In Advances in Cryptology - EUROCRYPT '94, volume 950 of Lecture Notes in Computer Science, pages 92-111. Springer-Verlag, 1994. [214](#)
- [6] M. Bellare and P. Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In Proceedings of the First ACM Conference on Computer and Communications Security, pages 62-73. 1993. [211](#)
- [7] M. Bellare and P. Rogaway. The Exact Security of Digital Signatures - How to sign with RSA and Rabin. In Advances in Cryptology - EUROCRYPT '96, volume 1070 of Lecture Notes in Computer Science, pages 399-416. Springer-Verlag, 1996. [212](#), [224](#)
- [8] J.-S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal Padding Schemes for RSA. In Advances in Cryptology - CRYPTO 2002, volume 2442 of Lecture Notes in Computer Science, pages 226-241. Springer-Verlag, 2002. [212](#), [217](#), [221](#), [222](#)
- [9] J.-S. Coron, M. Joye, D. Naccache, and P. Paillier. Universal Padding Schemes for RSA. Full version from <http://eprint.iacr.org/2002/115/>. 2002. [221](#)
- [10] R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In Advances in Cryptology - CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 13-25. Springer-Verlag, 1998. [214](#)
- [11] R. Cramer and V. Shoup. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. Available at <http://eprint.iacr.org/2001/108/>, 2001. [214](#)
- [12] E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern. RSA-OAEP Is Secure under the RSA Assumption. In Advances in Cryptology - CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pages 260-274. Springer-Verlag, 2001. [216](#), [222](#)
- [13] S. Goldwasser, S. Micali and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM Journal on Computing, 17(2):281-308, 1988. [215](#)
- [14] Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) << Cost(Signature) + Cost(Encryption). In Advances in Cryptology - CRYPTO '97, volume 1294 of Lecture Notes in Computer Science, pages 165-179. Springer-Verlag, 1997. [211](#)

A Proof of Theorem 2

Our proof technique is similar to one used in [7]. We are going to run the adversary \mathcal{A} in a simulated environment. We first describe or simulation before analysing how it could fail and showing how it could be used to invert the RSA function.

Our simulation must respond \mathcal{A} 's queries to the random oracles G and H and the signcryption oracle. We denote the algorithms to do this G_{sim} , H_{sim} , and S_{sim} respectively and we describe them below. To make our simulations sound we keep two lists L_G and L_H that are initially empty. The list L_G will consist of query/response pairs. At the end of the simulation we hope to find the partial preimage of c^* among queries in L_G .

$G_{sim}(\omega)$ If $(\omega, x) \in L_G$ for some x : Return x Else: $x \xleftarrow{r} \{0, 1\}^{n+k_0}$ Add (ω, x) to L_G Return x	$H_{sim}(m r)$ If $(m r, \omega, -) \in L_H$ for some ω : Return ω Else: $x \xleftarrow{r} \mathbb{Z}_{N_A}^*$ $z \leftarrow x^{e_A} \pmod{N_A}$ $y \leftarrow c^* z \pmod{N_A}$ Parse y as $s \omega$ Add $(m r, \omega, x, y, z)$ to L_H Add $(\omega, s \oplus (m r))$ to L_G Return ω	$S_{sim}(m r, (N_B, e_B))$ $x \xleftarrow{r} \mathbb{Z}_{N_A}^*$ $y \leftarrow x^{e_A} \pmod{N_A}$ Parse y as $s \omega$ Add $(m r, \omega, -, -, -)$ to L_H Add $(\omega, s \oplus (m r))$ to L_G If $x > N_B$, $x \leftarrow x - 2^{k-1}$ $c \leftarrow x^{e_B} \pmod{N_B}$ Return c
--	---	---

Let us now analyse our simulation. Consider events that would cause the adversary's view in our simulated run to differs from it's view in a real attack. Such an event could be caused by an error in G_{sim} , H_{sim} or S_{sim} . We let AskG be the event that there is an error in G_{sim} and define AskH and SBad analogously.

It is easily verified that

$$\Pr[\text{AskG}] = 0. \quad (25)$$

An error in H_{sim} will only occur if it attempts to add $(\omega, s \oplus (m||r))$ to L_G when $G(\omega)$ is already defined. We conclude that

$$\begin{aligned} \Pr[\text{AskH}] &\leq 2^{-k_1} \cdot \sum_{i=0}^{q_h-1} (q_g + q_s + i) \\ &= 2^{-(k_1+1)} \cdot q_h \cdot (2q_g + q_h + 2q_s - 1). \end{aligned} \quad (26)$$

An error in S_{sim} will occur if it attempts to add $(m||r, \omega, -, -, -)$ to L_H when $H(m||r)$ is already defined. The only other possibility for an error in S_{sim} is attempting to add $(\omega, s \oplus (m||r))$ to L_G when $G(\omega)$ is already defined. We conclude that

$$\begin{aligned} \Pr[\text{SBad}] &\leq 2^{-k_0} \cdot \left(\sum_{i=0}^{q_s-1} (q_h + i) \right) + 2^{-k_1} \cdot \left(\sum_{i=0}^{q_s-1} (q_g + q_h + i) \right) \\ &= q_s \cdot \left(2^{-(k_0+1)} \cdot (2q_h + q_s - 1) + 2^{-(k_1+1)} \cdot (2q_g + 2q_h + q_s - 1) \right). \end{aligned} \quad (27)$$

We also define the event FBad to be that when \mathcal{A} outputs a valid forged signcryption c on some message m , but $m||r$ was never a query to H_{sim} . Clearly we have

$$\Pr[\text{FBad}] \leq 2^{-k_1}. \quad (28)$$

We define the event Bad to be

$$\text{Bad} = \text{AskG} \vee \text{AskH} \vee \text{SBad} \vee \text{FBad}. \quad (29)$$

Let us consider the event \mathcal{A} wins $\wedge \neg \text{Bad}$ in our simulated run of \mathcal{A} . If this event occurs then \mathcal{A} outputs a forged signcryption c of some m such that $(m||r, \omega, x, y, z) \in L_H$ for some r, ω, x, y, z . Now, looking at the construction of H_{sim} we see that we have

$$(c/x)^{e_A} = (y/x^{e_A}) = (y/z) = (c^* z/z) = c^* \pmod{N_A}. \quad (30)$$

Therefore $(c/x) \pmod{N_A}$ is the e_A -th root of c^* modulo N_A as required. We denote the event that we manage to find the e_A -th root modulo N_A of c^* by Invert . We see from (30) that

$$\Pr[\text{Invert}]_{sim} \geq \Pr[\mathcal{A} \text{ wins} \wedge \neg \text{Bad}]_{sim}, \quad (31)$$

where the subscript sim denotes the fact that these are probabilities in our simulated run of \mathcal{A} . We will denote probabilities in a real execution of \mathcal{A} with the subscript $real$. From (31) and the definition of Bad we see that

$$\Pr[\text{Invert}]_{sim} \geq \Pr[\mathcal{A} \text{ wins} \wedge \neg \text{Bad}]_{real} \geq \Pr[\mathcal{A} \text{ wins}]_{real} - \Pr[\text{Bad}]_{real}. \quad (32)$$

The result now follows from (25), (26), (27), (28), (29) and (32).

Cryptography after the Bubble: How to Make an Impact on the World

Tom Berson

Anagram Laboratories

berson@anagram.com

<http://anagram.com/berson>

Abstract. The hype is over. Cryptographers can fire their PR agents, tear up their stock options, and get back to work. But what to work on? Cryptography cuts across many spheres of human activity. Important challenges remain open at levels of difficulty to suit every taste. We suggest scientific, engineering, social and political agendas in the hope that we will encourage cryptographers to do great things.

Rethinking Chosen-Ciphertext Security under Kerckhoffs' Assumption

Seungjoo Kim¹, Masahiro Mambo², and Yuliang Zheng³

¹ KISA (Korea Information Security Agency)

78, Garag-Dong, Songpa-Gu, Seoul 138-803, Korea

skim@kisa.or.kr

<http://www.crypto.re.kr>

² Graduate School of Information Sciences, Tohoku University

Kawauchi Aoba Sendai, 980-8576 Japan

mambo@icl.isc.tohoku.ac.jp <http://www.icl.isc.tohoku.ac.jp/~mambo/>

³ UNC Charlotte

9201 University City Blvd, Charlotte, NC 28223

yzheng@uncc.edu

<http://www.sis.uncc.edu/~yzheng/>

Abstract. Kerckhoffs' assumption states that an attacker must be assumed to have full knowledge of all the details of a cryptosystem except information about encryption/decryption keys upon which security of the cryptosystem rests entirely. In this paper we generalize the assumption to allow an attacker to have access to intermediate results during the computational process of cryptographic operations. We show that the generalized assumption models quite well such real world attacks as the “memory reconstruction attack” and the “memory core-dump attack” which may be mounted by computer forensic software or computer viruses. We further analyze a number of public key encryption schemes under the generalized Kerckhoffs' assumption, and demonstrate that some of the schemes, although provably secure under some computational assumptions, may be broken if an attacker has access to intermediate results during a decryption operation.

Keywords: Kerckhoffs' assumption, provable security, chosen-ciphertext security.

1 Introduction

A basic rule of cryptography is to use published, well scrutinized algorithms and protocols. This principle, called *Kerckhoffs' assumption* (*also called Kerckhoffs' law or Kerckhoffs' principle*) was first stated in 1883 by Auguste Kerckhoffs: A cryptosystem should be designed to withstand cryptanalysis even if everything about it is known to an attacker, except information on keys used. It was reformulated by Claude Shannon as “the enemy knows the system” or Shannon's Maxim.

Kerckhoffs' assumption was in fact one of six design principles laid down by Kerckhoffs for military ciphers. These six cipher design principles were [22]:

1. The system must be practically, if not mathematically, undecipherable.
2. It must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience.
3. Its key must be communicable and retainable without the help of written notes, and changeable or modifiable at the will of the correspondents.
4. It must be applicable to telegraphic correspondence.
5. It must be portable, and its usage and function must not require the concourse of many people.
6. Finally, it is necessary, seeing the circumstances that the application commands, that the system be easy to use, requiring neither mental strain nor the knowledge of a long series of rules to observe.

Among the above six design principles, the second statement means that the method used to encipher data is known to the opponent, and that security must lie in the choice of a key. A corollary of Kerckhoffs' second principle is that the smaller the number of secrets a system has, the more secure it is. In particular, if the loss of any secret key causes the compromise of security, then a system with fewer secrets is necessarily more secure.

In this paper, we explore a generalization of Kerckhoffs' assumption fitted for the computer era. By using this generalization, we study the (in)security of existing provably secure cryptosystems against chosen-ciphertext attacks.

2 Definitions of Security

Definition 1 (Generalized Kerckhoffs' Assumption). *Let's consider a Turing Machine(TM) which models an encryption or decryption program. The TM has 5 tapes: input, output, random, work, and key tapes. The generalized Kerckhoffs' assumption means an attacker knows all details of the TM except data on the key tape, where "all details of the TM" consist of*

- (External Details: the original Kerckhoffs' assumption deals with only this situation) a triplet (G, E, D) satisfying the following conditions and the bit sequence contained in the input/output tape of TM; and
 - key generation algorithm G : G , on input 1^k (the security parameter), produces a pair of encryption/decryption keys (e, d) . The encryption key e is stored in the input tape, but the decryption key d is stored in the key tape.
 - encryption algorithm E : E takes as input a security parameter 1^k , an encryption key e from the range of $G(1^k)$, and a message $m \in \{0, 1\}^k$, and produces as output a ciphertext $c \in \{0, 1\}^{*\dagger}$.
 - decryption algorithm D : D takes as input a security parameter 1^k , a secret decryption key d from the range of $G(1^k)$, and a ciphertext c from the range of $E(1^k, e, m)$ and produces as output the message m .

¹ We use the notation $c \in E(1^k, e, m)$ to denote c being an encryption of a message m using a key e with a security parameter k . When it is clear, we use $c \in E_e(m)$, or $c \in E(m)$ for short.

- (*Internal Details*) The bit sequence contained in the random/work tape of the TM. (If the attacker is given the ability not only to read but also to write or modify the random/work tape of the TM, resulting in erroneous outputs, we can then simulate the fault attacks [8].)

Note here that, when D_d (resp. E_e) is being applied to a target ciphertext itself(resp. a plaintext corresponding to a target ciphertext), the attacker cannot get access to the input/output/random/work tape of the TM.

Under the assumption of Definition 1, attacks are classified based on what information an attacker has access to in addition to intercepted ciphertexts. The most prominent classes of attack for public-key cryptosystems are : ciphertext-only attack, known-plaintext attack, chosen-plaintext attack, and chosen-ciphertext attack[29, 34]. In the following, we consider the chosen-ciphertext attack under the generalized Kerckhoffs' assumption. We note that the scope of our assumption is very broad and may apply to other cryptographic primitives as well (e.g., digital signatures).

Definition 2 (Chosen-Ciphertext Query under Generalized Kerckhoffs' Assumption). Let k be a security parameter that generates matching encryption/decryption keys (e, d) for each user in the system. A chosen-ciphertext query under generalized Kerckhoffs' assumption is a process which, on input 1^k and e , obtains

- (*External Details*) the plaintext (relative to d), in the output tape of decryption oracle, corresponding to a chosen ciphertext; or an indication that the chosen ciphertext is invalid²; and also
- (*Internal Details*) unerased internal states contained in the random/work tape of the decryption oracle for the submitted ciphertext.

Definition 3 ([Static/Adaptive] Chosen-Ciphertext Attack under Generalized Kerckhoffs' Assumption). A static chosen-ciphertext attack³ under generalized Kerckhoffs' assumption consists of the following scenario:

² As explicitly pointed out in [21], we stress that the adversary may query the decryption oracle with invalid ciphertexts. Although seemingly useless, such attacks are not innocuous. The decryption oracle can for example be used to learn whether a chosen ciphertext is valid or not. From this single bit of information and by iterating the process, Bleichenbacher successfully attacked several implementations of protocols based on PKCS #1 v1.5 [9]. More recently, Manger pointed out the importance of preventing an attacker from distinguishing between rejections at the various steps of the decryption algorithm, say, using timing analysis [28]. The lesson is that implementors must ensure that the reasons for which a ciphertext is rejected are hidden from the outside world.

³ In the past, this attack has also been called “lunch-time attack” or “midnight attack”.

1. On input a security parameter k , the key generation algorithm G is run, generating a public encryption key $e \in G(1^k)$ and a private decryption key $d \in G(1^k)$ for the encryption algorithm E . The attacker of course knows the public key e , but the private key d is kept secret.
2. [Find stage] The attacker makes polynomially (in k) many chosen-ciphertext queries (as in Definition 2) to a decryption oracle. (The attacker is free to construct the ciphertexts in an arbitrary way — it is certainly not required to compute them using the encryption algorithm.)
3. The attacker prepares two messages m_0, m_1 and gives these to an encryption oracle. The encryption oracle chooses $b \in_R \{0, 1\}$ at random, encrypts m_b , and gives the resulting “target ciphertext” c' to the attacker. The attacker is free to choose m_0 and m_1 in an arbitrary way, except that they must be of the same length.
4. [Guess stage] The attacker outputs $b' \in \{0, 1\}$, representing its “guess” on b .

In an adaptive chosen-ciphertext attack under generalized Kerckhoffs’ assumption, the attacker has additional access to the decryption oracle after having received the target ciphertext : a second series of polynomially (in k) many chosen-ciphertext queries may be run. The unique restriction is not to probe the decryption oracle with the target ciphertext c' .

The success probability in the previous attack scenario is defined as

$$\Pr[b' = b] .$$

Here, the probability is taken over coin tosses of the adversary, the key generation algorithm G and the encryption algorithm E , and $(m_0, m_1) \in M^2$ where M is the domain of the encryption algorithm E .

The attack of Definition 3 is very powerful. The attacker can not only obtain the plaintexts corresponding to chosen ciphertexts, but also invade user’s computer and read the contents of its memory (i.e., unerased internal data) [23, 24]. We believe that this attack model approximates more closely existing security systems, many of which are built on such operating systems as Unix and Windows where a reasonably privileged user can interrupt the operation of a computing process and inspect its intermediate results at ease. Thus our attack model can deal with the “memory reconstruction attack [10]” or the “memory core-dump attack [23, 24]” mounted by computer forensic software⁴, information stealing viruses, or other accidental reasons.

Definition 4 (Chosen-Ciphertext Security under Generalized Kerckhoffs’ Assumption). “An encryption scheme is (t, q, ε) -secure under $(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$ -generalized Kerckhoffs’ assumption” means

⁴ Computer forensic is to gather and analyze data in a manner as free from distortion or bias as possible to reconstruct data or what has happened in the past on a system.

- that an attacker does not know the decryption key d stored in the key tape of decryption oracle, and that at least $(l_{\text{in}} + l_{\text{out}} + l_{\text{rand}} + l_{\text{work}})$ -bit sequence collectable from input tape, output tape, random tape and work tape, respectively, of the decryption oracle is kept secret to the attacker.
- that the chosen-ciphertext attacker of Definition 3 can break the encryption scheme with an advantage not larger than ε , when s/he runs for time t and makes q queries to the decryption oracle.

We note that the original Kerckhoffs' assumption corresponds to $(\infty, \infty, \infty, \infty)$ -generalized Kerckhoffs' assumption, meaning all information on the tapes must be kept secret to an attacker.

We also note that, under the generalized Kerckhoffs' assumption, the power of a chosen-ciphertext attack deeply depends on the four-tuple of $(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$, so we can consider this four-tuple as one of the security evaluation criteria of a given cryptosystem. Thus, for example, when two encryption schemes based on the same cryptographic assumptions, E_e and E'_e , are given, we would say something like : “Even if both of an encryption scheme E_e and an E'_e are IND-CCA2 in terms of the definition of [4], when an attacker intentionally chooses and sends invalid ciphertexts, $(\delta_{\text{in}}, \delta_{\text{out}}, \delta_{\text{rand}}, \delta_{\text{work}})$ more bits of E'_e , except the key information, should be kept secret than E_e using the same security parameter as E'_e .”

2.1 Forward Security versus Ours

Forward security is introduced based on our desire that the compromise of a long-term private key should not result in the compromise of any earlier session key. With a forward secure system, even if the private decryption key is compromised and an eavesdropper has captured encrypted messages which has previously been sent, s/he will not be able to decrypt them.

Proactive cryptography [31], exposure-resilient cryptography [11], forward-secure signatures [2, 5], and key-insulated cryptography [15, 16] may all be viewed as different approaches to the same goal.

While the forward security model deals with a resistance to *known-key attacks*, our model is motivated by work on *strong chosen-ciphertext attack with memory dump* [23, 24]. In our model, an attacker knows even the internal states of a system. The sole restriction is that private decryption keys are not accessible to attackers.

3 How to Measure the Amount of Information Appeared in Work Tape

To determine the amount of bit sequence collectable from work tape, first of all, we suppose a crypto library which provides the subroutines of Table 1, Table 2, Table 3 and Table 4 for manipulating arbitrary length integers over the integers

Table 1. Subroutines for basic calculations

Subroutine	Functionality
<code>Abs(x;a)</code>	x is the absolute value of a
<code>Negate(x;a)</code>	$x = -a$
<code>Add(x;a,b)</code>	$x = a + b$
<code>Sub(x;a,b)</code>	$x = a - b$
<code>Mul(x;a,b)</code>	$x = a \times b$
<code>DivRem(q,r;a,b)</code>	division of a by b , quotient in q , remainder in r
<code>Power(x;a,e)</code>	$x = a^e$
<code>Sqr(x;a)</code>	$x = a^2$
<code>GCD(x;a,b)</code>	$x = \gcd(a,b)$
<code>XGCD(d,s,t;a,b)</code>	$d = \gcd(a,b) = a \times s + b \times t$

Table 2. Subroutines for modular arithmetic

Subroutine	Functionality
<code>Mod(x;a,n)</code>	$x = a \bmod n$
<code>NegateMod(x;a,n)</code>	$x = -a \bmod n$
<code>AddMod(x;a,b,n)</code>	$x = a + b \bmod n$
<code>SubMod(x;a,b,n)</code>	$x = a - b \bmod n$
<code>MulMod(x;a,b,n)</code>	$x = a \times b \bmod n$
<code>InvMod(x;a,n)</code>	$x = a^{-1} \bmod n$
<code>PowerMod(x;a,e,n)</code>	$x = a^e \bmod n$
<code>SqrMod(x;a,n)</code>	$x = a^2 \bmod n$
<code>Jacobi(a,n)</code>	compute Jacobi symbol of $(a n)$

and over finite fields⁵. Note here that the (unit) operations contained inside a subroutine must be executed without external interruptions (e.g., memory core dump, system crash, user signal to kill the transaction, and so on). After the execution being successfully completed, the internal variables used by subroutine are erased. Refer to [14, 23, 24] for details.

Now, we write pseudo-code to describe the given algorithm by using our crypto library, and then count the size of the temporary variables used for intermediate values or return value of subroutines. When writing the pseudo-code, we assume that, the data can be packed into a variable bit-by-bit(or byte-by-byte) as the bit-field structure of C programming language. Thus, for variables, we use the notation $V : V = v_{l-1}256^{l-1} + v_{l-2}256^{l-2} + \cdots + v_1256 + v_0 \stackrel{\text{def}}{=} \{v_{l-1}v_{l-2}\dots v_1v_0\}$, where $0 \leq v_i < 256$. To indicate some consecutive bytes of an integer V , we use a shorter notation $[V]_a^b$, meaning a sequence of bytes from the b -th byte to a -th byte of v , $\{v_b \dots v_a\}$.

⁵ We name the module according to the NTL(A Library for doing Number Theory) of Victor Shoup, and use “;” between output and input of all subroutines.

Table 3. Subroutines for bitwise operations

Subroutine	Functionality
<code>LeftShift(x;a,n)</code>	x is to shift the bits of a left by n
<code>RightShift(x;a,n)</code>	x is to shift the bits of a right by n (the sign is preserved)
<code>Bit_And(x;a,b)</code>	x is the bitwise AND of a and b
<code>Bit_Or(x;a,b)</code>	x is the bitwise OR of a and b
<code>Bit_Xor(x;a,b)</code>	x is the bitwise XOR of a and b

Table 4. Etc.

Subroutine	Functionality
<code>GenPrime(n;l,err)</code>	generating a random prime n of length l so that the probability that n is composite is bounded by 2^{-err}
<code>ProbPrime(n,NumTrials)</code>	perform up to <code>NumTrials</code> Miller-witness tests of n
<code>HashH(x;a[,b,…])</code>	x is the hash value of $H(a[,b,…])$
<code>HashG(x;a[,b,…])</code>	x is the hash value of $G(a[,b,…])$
<code>MemKill(a[,b,…])</code>	delete and set to zero the memory allocated for $a[,b,…]$

Programming Style : Each programmer will, of course, have his or her own preferences in wiring style. Since the programming style may effect the memory use and the performance, we should restrict something on writing style of pseudo-code : (i) do not reuse variables. For example,

```
result = function1(input); result = function2(result);
```

(ii) do not use nested functions as

```
result = function2(function1(input));
```

but call them on consecutive lines as

```
result1 = function1(input); result2 = function2(result1);.
```

From now on, when we express “a cryptosystem is implemented with the crypto library described in this section”, we assume

- the given cryptosystem is implemented with the crypto library and the programming style defined in this section; and,
- operations contained inside a subroutine of library must be executed without external interruptions; and,
- decryption key itself appeared in the pseudo-code is always protected even if it is place on the work tape; and,
- the faulty behavior[8] of the cryptosystem is excluded from the evaluation.

4 Evaluation of Provably Secure Cryptosystems

For the last few years, many new schemes have been proposed with provable security against chosen-ciphertext attacks. Before 1994, only theoretical (i.e., not very practical) schemes were proposed. Then Bellare and Rogaway [7] came up with the *random oracle model* [6] and subsequently designed in [7] a generic padding, called OAEP (Optimal Asymmetric Encryption Padding), to transform a one-way (partially) trapdoor *permutation* into a chosen-ciphertext secure cryptosystem. Other generic paddings, all validated in the random oracle model, were later given by Fujisaki and Okamoto [18] (improved in [19]), by Pointcheval [33], and by Okamoto and Pointcheval [30]. Recently, Coron *et al.* proposed a generic IND-CCA2 conversion, which reduced the encryption size and/or speeded up the decryption process [12]. The first practical cryptosystem with provable security in the *standard model* is due to Cramer and Shoup [13]. They present an extended ElGamal encryption provably secure under the *decisional* Diffie-Hellman problem. (Cramer-Shoup system is also provably secure under the weaker *computational* Diffie-Hellman assumption in the random oracle model.)

In this section, we try to evaluate the existing provably secure cryptosystems under the generalized Kerckhoffs' assumption. Note here that, when evaluating the existing provably secure cryptosystems, we do not consider the computational difficulties of the underlying cryptographic assumptions.

4.1 RSA-OAEP

We give here a brief overview of RSA-OAEP. We describe the RSA-OAEP scheme as described in [32], but in our description, the decryption phase of RSA-OAEP is divided into three parts : (i) RSA decryption, (ii) validity test, and (iii) output.

Let k -byte string $n = pq$ denote an RSA modulus, which is the product of two large primes p and q . Furthermore, let e and d , satisfying $ed \equiv 1 \pmod{\text{lcm}(p-1, q-1)}$, respectively denote the public encryption exponent and the private decryption exponent.

We assume a hash function $H : \{0, 1\}^{k-hLen-1 \text{ bytes}} \rightarrow \{0, 1\}^{hLen \text{ bytes}}$ and a “generator” function $G : \{0, 1\}^{hLen \text{ bytes}} \rightarrow \{0, 1\}^{k-hLen-1 \text{ bytes}}$. The public parameters are $\{n, e, G, H\}$ and the secret parameters are $\{d, p, q\}$.

A $mLen$ -byte plaintext message m is encrypted through RSA-OAEP as :

1. Form a data block DB of length $k - hLen - 1$ bytes as

$$DB = (H(L) \| 0x00^{k-mLen-2hLen-2} \| 0x01 \| m),$$

where L is an optional label to be associated with the message; the default value for L , if L is not provided, is the empty string; and

2. Let $maskedDB = DB \oplus G(seed)$, with a random string $seed$ of length $hLen$ -bytes⁶; and
3. Let $maskedSeed = seed \oplus H(maskedDB)$ ⁷; and

⁶ In [32], $maskedDB = DB \oplus MGF(seed, k - hLen - 1)$.

⁷ In [32], $maskedDB = seed \oplus MGF(maskedDB, hLen)$.

4. Form an encoded message EM of length k bytes as

$$EM = (0x00\|maskedSeed\|maskedDB); \text{ and}$$

5. RSA encryption $c = EM^e \bmod n$.

Given a ciphertext c , m is recovered as :

(i) RSA decryption

```

1 : PowerMod(V1; c, d, n);
/* V1  $\stackrel{\text{def}}{=} \underbrace{\{v1_{k-1} v1_{k-2} \dots v1_{k-hLen-1}\}}_{0x00} \underbrace{\{v1_{k-hLen-2} \dots v1_0\}}_{maskedDB} */$ 
```

2 : HashH(V2; $[V1]_0^{k-hLen-2}$);
3 : Bit_Xor(V3; $[V1]_{k-hLen-1}^{k-2}, V2$);
4 : HashG(V4; V3);
5 : Bit_Xor(V5; $[V1]_0^{k-hLen-2}, V4$);
/* V5 $\stackrel{\text{def}}{=} \underbrace{\{v5_{k-hLen-2} \dots v5_{k-2hLen-1}\}}_{H(L)} \underbrace{\{v5_{k-2hLen-2} \dots v1_{mLen+1}\}}_{0x00^{k-mLen-2hLen-2}}$
 $\underbrace{\{v5_{mLen} v5_{mLen-1} \dots v5_0\}}_{0x01} */$

(ii) Validity Test

If $(v5_{mLen} \neq 0x01 \vee [V5]_{k-2hLen-1}^{k-hLen-2} \neq H(L) \vee v1_{k-1} \neq 0x00)$ then
• MemKill($V1, V2, V3, V4, V5$); and
• Return("Invalid Ciphertext").

(iii) Output

- Return($[V5]_0^{mLen-1}$).

Theorem 1. RSA-OAEP, implemented with crypto library of Section 3, is IND-CCA2 under $(0, 0, 0, 2|n|)$ -generalized Kerckhoffs' assumption. Here, $|n|$ denotes the bit length of modulus n .

Proof (Sketch). The problem with RSA-OAEP resides in that the validity test cannot be performed (i.e., the adversary cannot be detected) until *after* the RSA decryption is completed. Thus an attacker can freely mount an adaptive chosen-ciphertext attack and extract the partial information from internal data in the decryption oracle's memory [23, 24]. To prevent this type of attack, $V1 = 0x00\|V3 \oplus V2\|V5 \oplus V4$ should not leak out. Thus, the variables $V1$, $V3$ and $V5$ should be protected against the attacker.

4.2 Abe Scheme

Let (n, e) be RSA public encryption key and d be private decryption key. We select a hash function $H : \mathbf{Z}_n^* \times \mathbf{Z}_n^* \rightarrow \{0, 1\}^{l_m}$, $G : \{0, 1\}^* \rightarrow \{0, 1\}^l$ where l is a security parameter.

In Abe scheme⁸, ciphertext C of message $m \in \{0, 1\}^{l_m}$ is $C = (h, c, u, s)$ such that $h = r^e \bmod n$, $c = m \oplus H(h, r)$, $u = G(c, w^e \bmod n)$, $s = w \cdot r^u \bmod n$, where $r, w \in_U \mathbf{Z}_n^*$. Given a ciphertext C , m is recovered as follows. Contrary to RSA-OAEP, the validity test of Abe Scheme is performed *before* the RSA decryption :

(i) Validity Test

- 1 : PowerMod($V1; s, e, n$);
- 2 : PowerMod($V2; h, u, n$);
- 3 : InvMod($V3; V2, n$);
- 4 : MulMod($V4; V1, V3, n$);
- 5 : HashG($V5; c, V4$);
- 6 : Check if $u \neq V5$.

(ii) ElGamal decryption

- If ($u \neq V5$) then
 - MemKill($V1, V2, V3, V4, V5$); and
 - Return("Invalid Ciphertext").
- Else
 - 7 : PowerMod($V6; h, d, n$);
 - 8 : HashH($V7; h, V6$);
 - 9 : Bit_Xor($V8; c, V7$);

(iii) Output

- Return($V8$).

Theorem 2. *Abe scheme, implemented with crypto library of Section 3, is IND-CCA2 under Kerckhoffs' assumption.*

Proof (Sketch). Suppose that, given a target ciphertext C , a chosen-ciphertext attacker modifies it and sends C' to a decryption oracle. Because Abe scheme is a chosen-ciphertext secure encryption scheme, the work tape of decryption oracle contains only $V1, V2, V3, V4$ and $V5$. However, even though the attacker can get access to these temporary variables, s/he has no advantage, since whatever the attacker can compute with $V1, V2, V3, V4$ and $V5$, s/he can also compute only with C' by her/himself.

⁸ Although the motivation is quite different, a similar “validation-then-decryption”-type RSA scheme was imagined by the authors of this paper : ciphertext C of message m is (c, v_1, v_2) such that : $c = w^e \bmod n$, where $w = s \| t$, $s = m \oplus F(r)$, and $t = r \oplus G(s)$, $v_1 = H(n, e, c, ID_{sender}, x^{-e} \bmod n)$ with $x \in_R \mathbf{Z}_n^*$, $v_2 = x \cdot w^{v_1} \bmod n$.

4.3 Cramer-Shoup Scheme

We give here a brief overview of Cramer-Shoup scheme and refer the reader to [13] for details. Here, for simplicity, the same notation H is used, but function H may have different domain and image from that used in RSA-OAEP of Section 4.1 and Abe scheme of Section 4.2. Let $\mathcal{G} = \langle g \rangle$ be a cyclic group of large prime order q , generated by g .

The encryption of Cramer-Shoup scheme is $(c_1, c_2, c_3, c_4) = (g_1^s, g_2^s, X_1^s \cdot X_2^s \cdot X_3^{s \cdot H(c_1, c_2, c_3)})$ with public keys $X_1 = g_1^z$, $X_2 = g_1^{x_1} \cdot g_2^{x_2}$, and $X_3 = g_1^{y_1} \cdot g_2^{y_2}$. The decryption is as follows :

(i) Validity Test

- 1 : $\text{HashH}(V1; c_1, c_2, c_3);$
- 2 : $\text{MulMod}(V2; y_1, V1, q);$
- 3 : $\text{AddMod}(V3; x_1, V2, q);$
- 4 : $\text{PowerMod}(V4; c_1, V3, p);$
- 5 : $\text{MulMod}(V5; y_2, V1, q);$
- 6 : $\text{AddMod}(V6; x_2, V5, q);$
- 7 : $\text{PowerMod}(V7; c_2, V6, p);$
- 8 : $\text{MulMod}(V8; V4, V7, p);$
- 9 : Check if $c_4 \neq V8.$

(ii) ElGamal decryption

If $(c_4 \neq V8)$ then

- MemKill ($V1, V2, V3, V4, V5, V6, V7, V8$); and
- Return("Invalid Ciphertext").

Else

- 10 : $\text{NegateMod}(V9; z, q);$
- 11 : $\text{PowerMod}(V10; c_1, V9, p);$
- 12 : $\text{MulMod}(V11; V10, c_3, p);$

(iii) Output

- Return($V11$).

Theorem 3. *Cramer-Shoup scheme, implemented with crypto library of Section 3, is IND-CCA2 under $(4|q|, 0, 0, 2|q|)$ -generalized Kerckhoffs' assumption.*

Proof (Sketch). For the scheme by Cramer and Shoup, some secret data are involved in the validity test. As a consequence, not only the private decryption key (i.e., z) of key tape but also the private validation keys (i.e., (x_1, x_2) and (y_1, y_2)) of input tape need to be kept secret. Also $V2$ and $V5$ of work tape should be protected.

4.4 Other Provably Secure Cryptosystems

Several variants of the basic RSA scheme[35] and the basic ElGamal scheme[17] were proposed in order to make it provably secure against chosen-ciphertext attacks. In this section, we review some of them in the chronological order of their appearance and analyze their resistance under our security model. Table 5 shows, when implemented with crypto library of Section 3, how many bits of information of decryption oracle except the decryption key should be protected from the attacker of Definition 3. Here, l_d denotes the bit length of decryption key of key tape.

Tsiounis and Yung [37]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, X^y \cdot m, g^k, y \cdot H(g, c_1, c_2, c_3) + k)$
- decryption:
 - 1) PowerMod($V1; g, c_4, p$);
 - 2) HashH($V2; g, c_1, c_2, c_3$);
 - 3) PowerMod($V3; c_1, V2, p$);
 - 4) MulMod($V4; V3, c_3, p$);
 - 5) If ($V1 \neq V4$) then
 - 6) • MemKill($V1, V2, V3, V4$); and
 - 7) • Return("Invalid Ciphertext").
 - Else
 - 8) • NegateMod($V5; x, q$);
 - 9) • PowerMod($V6; c_1, V5, p$);
 - 10) • MulMod($V7; V6, c_2, p$);
 - 11) • Return($V7$).

Fujisaki and Okamoto [18]

- encryption: $(c_1, c_2) = (g^{H(m\|s)}, (m\|s) \oplus X^{H(m\|s)})$
- decryption:
 - 1) PowerMod($V1; c_1, x, p$);
 - 2) Bit_Xor($V2; V1, c_2$);
 - 3) HashH($V3; V2$);
 - 4) PowerMod($V4; g, V3, p$);
 - 5) If ($c_1 \neq V4$) then
 - 6) • MemKill($V1, V2, V3, V4$); and
 - 7) • Return("Invalid Ciphertext").
 - 8) Return(m of $V2$).
- attack:
 - 1) set $(c'_1, c'_2) = (c_1, c_2 \oplus r)$ for a random r
 - 2) recover m from $(m' \|\dots) \oplus r = m \|\dots$

Fujisaki and Okamoto [19]

- encryption: $(c_1, c_2, c_3) = (g^{H(s,m)}, X^{H(s,m)} \cdot s, \mathcal{E}_{G(s)}^{\text{sym}}(m))$

- decryption:
 - 1) **NegateMod**($V1; x, q$);
 - 2) **PowerMod**($V2; c_1, V1, p$);
 - 3) **MulMod**($V3; V2, c_2, p$);
 - 4) **HashG**($V4; V3$);
 - 5) $V5 = \mathcal{D}_{V4}^{\text{sym}}(c_3)$;
 - 6) **HashH**($V6; V3, V5$);
 - 7) **PowerMod**($V7; X, V6, p$);
 - 8) **MulMod**($V8; V7, V3, p$);
 - 9) If ($c_2 \neq V8$) then
 - 10) • **MemKill**($V1, V2, V3, V4, V5, V6, V7, V8$); and
 - 11) • **Return**("Invalid Ciphertext").
 - 12) **Return**($V5$).
- attack:
 - 1) set $(c'_1, c'_2, c'_3) = (g^r \cdot c_1, X^r \cdot c_2, c_3)$ for a random r
 - 2) recover $m = m'$

Pointcheval [33]

- encryption: $(c_1, c_2, c_3) = (g^{H(m\|s)}, X^{H(m\|s)} \cdot k, (m\|s) \oplus G(k))$
- decryption:
 - 1) **NegateMod**($V1; x, q$);
 - 2) **PowerMod**($V2; c_1, V1, p$);
 - 3) **MulMod**($V3; V2, c_2, p$);
 - 4) **HashG**($V4; V3$);
 - 5) **Bit_Xor**($V5; V4, c_3$);
 - 6) **HashH**($V6; V5$);
 - 7) **PowerMod**($V7; g, V6, p$);
 - 8) If ($c_1 \neq V7$) then
 - 9) • **MemKill**($V1, V2, V3, V4, V5, V6, V7$); and
 - 10) • **Return**("Invalid Ciphertext").
 - 11) **Return**(m of $V5$).
- attack:
 - 1) set $(c'_1, c'_2, c'_3) = (c_1, c_2, c_3 \oplus r)$ for a random r
 - 2) recover m from $(m'\|\dots) \oplus r = m\|\dots$

Baek, Lee, and Kim [3]

- encryption: $(c_1, c_2) = (g^{H(m\|s)}, (m\|s) \oplus G(X^{H(m\|s)}))$
- decryption:
 - 1) **PowerMod**($V1; c_1, x, p$);
 - 2) **HashG**($V2; V1$);
 - 3) **Bit_Xor**($V3; V2, c_2$);
 - 4) **HashH**($V4; V3$);
 - 5) **PowerMod**($V5; g, V4, p$);
 - 6) If ($c_1 \neq V5$) then
 - 7) • **MemKill**($V1, V2, V3, V4, V5$); and
 - 8) • **Return**("Invalid Ciphertext").
 - 9) **Return**(m of $V3$).
- attack:
 - 1) set $(c'_1, c'_2) = (c_1, c_2 \oplus r)$ for a random r
 - 2) recover m from $(m'\|\dots) \oplus r = m\|\dots$

Schnorr and Jakobsson [36]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, G(X^y) + m, H(g^s, c_1, c_2), s + c_3 \cdot y)$

- decryption:
 - 1) $\text{PowerMod}(V1; g, c_4, p);$
 - 2) $\text{NegateMod}(V2; c_3, q);$
 - 3) $\text{PowerMod}(V3; c_1, V2, p);$
 - 4) $\text{MulMod}(V4; V1, V3, p);$
 - 5) $\text{HashH}(V5; V4, c_1, c_2);$
 - 6) If ($c_3 \neq V5$) then
 - 7) • $\text{MemKill}(V1, V2, V3, V4, V5);$ and
 - 8) • $\text{Return}(\text{"Invalid Ciphertext"}).$
 - Else
 - 9) • $\text{PowerMod}(V6; c_1, x, p);$
 - 10) • $\text{HashG}(V7; V6);$
 - 11) • $\text{SubMod}(V8; c_2, V7, p);$
 - 12) • $\text{Return}(V8).$

Okamoto and Pointcheval [30]

- encryption: $(c_1, c_2, c_3, c_4) = (g^y, X^y \oplus R, \mathcal{E}_{G(R)}^{\text{sym}}(m), H(R, m, c_1, c_2, c_3))$
- decryption:
 - 1) $\text{PowerMod}(V1; c_1, x, p);$
 - 2) $\text{Bit-Xor}(V2; V1, c_2);$
 - 3) $\text{HashG}(V3; V2);$
 - 4) $V4 = \mathcal{D}_{V3}^{\text{sym}}(c_3)$
 - 5) $\text{HashH}(V5; V2, V4, c_1, c_2, c_3);$
 - 6) If ($c_4 \neq V5$) then
 - 7) • $\text{MemKill}(V1, V2, V3, V4, V5);$ and
 - 8) • $\text{Return}(\text{"Invalid Ciphertext"}).$
 - 9) $\text{Return}(V4).$
- attack:
 - 1) set $(c'_1, c'_2, c'_3, c'_4) = (c_1, c_2, c_3, c'_4)$ with $c'_4 \neq c_4$
 - 2) recover $m = m'$

Table 5. Analysis of several RSA and ElGamal variants

Variant	Type	l_d	$(l_{\text{in}}, l_{\text{out}}, l_{\text{rand}}, l_{\text{work}})$
RSA-OAEP [32, 7]	DtV	$ \text{lcm}(p - 1, q - 1) $	$(0, 0, 0, 2 n)$
Tsiounis-Yung [37]	VtD	$ q $	$(0, 0, 0, 0)$
Cramer-Shoup [13]	VtD	$ q $	$(4 q , 0, 0, 2 q)$
Fujisaki-Okamoto [18]	DtV	$ q $	$(0, 0, 0, 2 p)$
Fujisaki-Okamoto [19]	DtV	$ q $	$(0, 0, 0, q + 2 p + G(\cdot))$
Pointcheval [33]	DtV	$ q $	$(0, 0, 0, q + 2 p + 2 (m s))$
Baek-Lee-Kim [3]	DtV	$ q $	$(0, 0, 0, p + 2 (m s))$
Schnorr-Jakobsson [36]	VtD	$ q $	$(0, 0, 0, 0)$
Okamoto-Pointcheval [30]	DtV	$ q $	$(0, 0, 0, 2 p + G(\cdot) + m)$
Abe [1]	VtD	$ \text{lcm}(p - 1, q - 1) $	$(0, 0, 0, 0)$

DtV : Decryption-then-Validation

VtD : Validation-then-Decryption

5 Conclusion

In this paper we have generalized Kerckhoffs' assumption, and used the generalized assumption to analyze the security of a number of cryptosystems proposed in the literature. We hope that this work can serve as a starting point for further research into building cryptosystems that are resistant against such real world attacks as "memory core-dump" and "memory reconstruction" attacks.

Acknowledgements

The authors are grateful to Marc Joye and Jaechul Sung for helpful discussions on the subject of this paper. We would also like to thank the anonymous referees for valuable comments.

References

- [1] M. Abe, "Securing "encryption + proof of knowledge" in the random oracle model", *Topics in Cryptology – CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 277–289, Springer-Verlag, 2002. [240](#)
- [2] R. Anderson, Invited lecture, *Fourth ACM Conference on Computer and Communications Security*, ACM, 1997. [231](#)
- [3] J. Baek, B. Lee, and K. Kim, "Secure length-saving ElGamal encryption under the computational Diffie-Hellman assumption", *Information Security and Privacy (ACISP 2000)*, volume 1841 of *Lecture Notes in Computer Science*, pages 49–58, Springer-Verlag, 2000. [239](#), [240](#)
- [4] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes", *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 26–45, Springer-Verlag, 1998. [231](#)
- [5] M. Bellare and S. Miner, "A forward-secure digital signature scheme", *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448, Springer-Verlag, 1999. [231](#)
- [6] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols", *First ACM Conference on Computer and Communications Security*, pages 62–73, ACM Press, 1993. [234](#)
- [7] M. Bellare and P. Rogaway, "Optimal asymmetric encryption", *Advances in Cryptology – EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 92–111, Springer-Verlag, 1995. [234](#), [240](#)
- [8] Bellcore Press Release, "New threat model breaks crypto codes", Sept. 1996, <http://www.bellcore.com/PRESS/ADVSRY96/facts.html/>. [229](#), [233](#)
- [9] D. Bleichenbacher, "A chosen ciphertext attack against protocols based on the RSA encryption standard PKCS #1", *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, Springer-Verlag, 1998. [229](#)
- [10] S. Burnett and S. Paine, "RSA Security's official guide to cryptography", *RSA Press*, 2001. [230](#)

- [11] R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai, “Exposure-resilient functions and all-or-nothing-transforms”, *Advances in Cryptology – Eurocrypt ’00*, volume 1807 of *Lecture Notes in Computer Science*, pages 453–469, Springer-Verlag, 2000. [231](#)
- [12] J. S. Coron, H. Handshuch, M. Joye, P. Paillier, D. Pointcheval, and C. Tymen, “GEM: A generic chosen-ciphertext secure encryption method”, *Topics in Cryptology - CT-RSA 2002*, volume 2271 of *Lecture Notes in Computer Science*, pages 263–276, Springer-Verlag, 2002. [234](#)
- [13] R. Cramer and V. Shoup, “A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack”, *Advances in Cryptology – CRYPTO ’98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25, Springer-Verlag, 1998. [234](#), [237](#), [240](#)
- [14] G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson, “How to forget a secret”, *Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 500–509, 1999. [232](#)
- [15] Y. Dodis, J. Katz, S. Xu, and M. Yung, “Strong key-insulated signature schemes”, Unpublished Manuscript. [231](#)
- [16] Y. Dodis, J. Katz, S. Xu, and M. Yung, “Key-insulated public key cryptosystems”, *Advances in Cryptology – Eurocrypt ’02*, volume 2332 of *Lecture Notes in Computer Science*, pages 65–82, Springer-Verlag, 2002. [231](#)
- [17] T. ElGamal, “A public key cryptosystems and a signature schemes based on discrete logarithms”, *IEEE Transactions on Information Theory*, **IT-31**(4):469–472, 1985. [238](#)
- [18] E. Fujisaki and T. Okamoto, “How to enhance the security of public-key encryption at minimum cost”, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68, Springer-Verlag, 1999. [234](#), [238](#), [240](#)
- [19] E. Fujisaki and T. Okamoto, “Secure integration of asymmetric and symmetric encryption schemes”, *Advances in Cryptology – CRYPTO ’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–544, Springer-Verlag, 1999. [234](#), [238](#), [240](#)
- [20] L. C. Guillou and J.-J. Quisquater, “A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory”, *Advances in Cryptology – EUROCRYPT ’88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128, Springer-Verlag, 1988.
- [21] M. Joye, J.-J. Quisquater, and M. Yung, “On the power of misbehaving adversaries”, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 208–222, Springer-Verlag, 2001. [229](#)
- [22] A. Kerckhoffs, “La cryptographie militaire (Military Cryptography)”, *Journal des sciences militaires*, vol.IX, pages 5-83, Jan. 1883, pages 161-191, Feb. 1883. [227](#)
- [23] S. Kim, J. H. Cheon, M. Joye, S. Lim, M. Mambo, D. Won, and Y. Zheng “Strong adaptive chosen-ciphertext attack with memory dump (Or : The importance of the order of decryption and validation)”, *Eighth IMA Conference on Cryptography and Coding 2001*, volume 2260 of *Lecture Notes in Computer Science* pages 114–127, Springer-Verlag, 2001. [230](#), [231](#), [232](#), [235](#)
- [24] S. Kim, J. H. Cheon, M. Joye, S. Lim, M. Mambo, D. Won, and Y. Zheng, “Security analysis of “provably” secure cryptosystems under strong adaptive chosen-ciphertext attack”, *Technical Report of IEICE*, ISSN 0913-5685, ISEC2001-89, Vol.101, No.507, pages 17-24, 2001. [230](#), [231](#), [232](#), [235](#)
- [25] P. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems”, *Advances in Cryptology – CRYPTO ’96*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113, Springer-Verlag, 1996.

- [26] P. Kocher, J. Jaffe, and B. Jun, “Introduction to differential power analysis and related attacks”, 1998, <http://www.cryptography.com/dpa/technical>.
- [27] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis”, *Advances in Cryptology – CRYPTO ’99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397, Springer-Verlag, 1999.
- [28] J. Manger, “A chosen ciphertext attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as standardized in PKCS #1”, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238, Springer-Verlag, 2001. [229](#)
- [29] M. Naor and M. Yung, “Public-key cryptosystems provably secure against chosen ciphertext attacks”, *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, ACM Press, 1990. [229](#)
- [30] T. Okamoto and D. Pointcheval, “REACT: Rapid enhanced-security asymmetric cryptosystem transform”, *Topics in Cryptology – CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 159–175, Springer-Verlag, 2001. [234](#), [240](#)
- [31] R. Ostrovsky and M. Yung, “How to withstand mobile virus attacks”, *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing – PODC ’91*, pages 51–59, ACM Press, 1991. [231](#)
- [32] PKCS #1 v2.1 : RSA Cryptography Standard, June 14, 2002.
<http://www.rsasecurity.com/rsalabs/pkcs/> [234](#), [240](#)
- [33] D. Pointcheval, “Chosen-ciphertext security for any one-way cryptosystem”, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 129–146, Springer-Verlag, 2000. [234](#), [239](#), [240](#)
- [34] C. Rackoff and D. Simon, “Noninteractive zero-knowledge proof of knowledge and chosen ciphertext attack”, *Advances in Cryptology – CRYPTO ’91*, volume 576 of *Lecture Notes in Computer Science*, pages 433–444, Springer-Verlag, 1992. [229](#)
- [35] R. L. Rivest, A. Shamir, and L. M. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM*, **21**(2):120–126, 1978. [238](#)
- [36] C. P. Schnorr and M. Jakobsson, “Security of signed ElGamal encryption”, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 73–89, Springer-Verlag, 2000. [239](#), [240](#)
- [37] Y. Tsiounis and M. Yung, “On the security of ElGamal-based encryption”, *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 117–134, Springer-Verlag, 1998. [238](#), [240](#)

Provably Secure Public-Key Encryption for Length-Preserving Chaumian Mixes

Bodo Möller

Technische Universität Darmstadt, Fachbereich Informatik
moeller@cdc.informatik.tu-darmstadt.de

Abstract. Mix chains as proposed by Chaum allow sending untraceable electronic e-mail without requiring trust in a single authority: messages are recursively public-key encrypted to multiple intermediates (mixes), each of which forwards the message after removing one layer of encryption. To conceal as much information as possible when using variable (source routed) chains, all messages passed to mixes should be of the same length; thus, message length should not decrease when a mix transforms an input message into the corresponding output message directed at the next mix in the chain. Chaum described an implementation for such length-preserving mixes, but it is not secure against active attacks. We show how to build practical cryptographically secure length-preserving mixes. The conventional definition of security against chosen ciphertext attacks is not applicable to length-preserving mixes; we give an appropriate definition and show that our construction achieves provable security.

1 Introduction

Chaum's *mix* concept [6] is intended to allow users to send untraceable electronic e-mail without having to trust a single authority. The idea is to use a number of intermediates such that it suffices for just one of these to be trustworthy in order to achieve untraceability. (The sender does not have to decide which particular intermediate he is willing to trust, he just must be convinced that at least one in a given list will behave as expected.) These intermediates, the *mixes*, are remailers accepting public-key encrypted input. Messages must be of a fixed size (shorter messages can be padded, longer messages can be split into multiple parts). To send a message, it is routed through a chain of mixes M_1, \dots, M_n : the sender obtains the public key of each mix; then he recursively encrypts the message (including the address of the final recipient) yielding $E_{M_1}(E_{M_2}(\dots E_{M_n}(payload) \dots))$ where E_{M_i} denotes encryption with M_i 's public key, and sends the resulting ciphertext to mix M_1 . (The public-key cryptosystem will typically be hybrid, i.e. involve use of symmetric-key cryptography for bulk data encryption.) Each mix removes the corresponding layer of encryption and forwards the decrypted message to the next mix; thus mix M_n will finally recover *payload*.

Each mix is expected to collect a large batch of messages before forwarding the decryption results. The messages in the batch must be reordered (mixed) to prevent message tracing. It is important to prevent replay attacks – a mix must not process the same message twice, or active adversaries would be able to trace messages by duplicating them at submission to cause multiple delivery. (Timestamps can be used to limit the timespan for which mixes have to remember which messages they have already processed; see [6] and [7].)

Usually it is desirable to allow *source routing*, i.e. let senders choose mix chains on a per-message basis. This increases the flexibility of the whole scheme: senders can make use of new mixes that go into operation, and they can avoid mixes that appear not to work properly; in particular, they can avoid mixes that suppress messages (be it intentionally or because of technical problems), which might be noticed when sending probe messages to oneself over mix chains. For source routing, in the recursively encrypted message, each layer must contain the address of the next mix in the chain so that each mix knows where to forward the message. A problem with the straightforward implementation of source routing is that messages will shrink as they proceed through the chain, not only because of the forwarding address for each layer that must be included, but also because public-key encryption increases message sizes. For optimal untraceability, we need *length-preserving mixes*: the messages that mixes receive should essentially look like the resulting messages that they forward to other mixes.

A construction of length-preserving mixes is given in [6] (a variant of this is used in the fielded system Mixmaster [15]): mix messages consist of a fixed number of slots of a fixed size. The first slot is public-key encrypted so that it can be read by the first mix in the chain. Besides control information directed at the respective mix (such as the address of the next mix in the chain or, in case of the last mix, the address of the final recipient), decryption yields a symmetric key that the mix uses to decrypt all the other slots. Then slots are shifted by one position: the decryption of the second slot becomes the new first slot, and so on. A new final slot is added consisting of random (garbage) data to obtain a mix message of the desired fixed length, which can then be forwarded to the next mix. On the way through the chain, each mix message consists of a number of slots with valid data followed by enough slots with garbage to fill all available slots; mixes are oblivious of how many slots contain valid data and how many are random. Each mix in the chain, when decrypting and shifting the slots, will “decrypt” the garbage slots already present and add a new final slot, thus increasing the number of random slots by one. We note that while the transformation of an incoming mix message to the corresponding outgoing mix message is *length-preserving*, the decryption step itself is actually *length-expanding* because some of the data obtained by decryption is control data directed at the current mix.

The problem with this method for obtaining a hybrid public-key cryptosystem with length-expanding decryption is that it is not secure against active attacks: assume that an adversary controls at least one mix, and that all senders submit well-formed messages. Now when the victim submits a message, the adversary can mark it by modifying one of the slots. This mark will persist as

the message is forwarded through the mix chain: assuming CBC or CFB mode block cipher encryption, certain blocks of the corresponding slot will essentially contain garbage due to the decryption and slot shifting performed by each mix (whereas unmarked messages will not show such defects). If the final mix is controlled by the adversary, the adversary may be able to notice the modification and thus break untraceability.

To rule out such attacks, the hybrid public-key cryptosystem should provide security against adaptive chosen ciphertext attacks (CCA security): that is, it should be secure against an adversary who can request the decryption of arbitrary ciphertexts.

In this paper, we present a secure and practical hybrid construction for length-preserving mixes, which is also more flexible than the slot-based approach. The structure of each layer of encryption resembles that of the public-key encryption scheme DHAES/DHIES ([1], [2]). The standard notion of CCA security for public-key cryptosystems is not applicable to our hybrid public-key cryptosystem with length-expanding decryption because the encryption operation must be defined differently for our application: encryption cannot be treated as an atomic black-box operation that takes a plaintext and returns a ciphertext – in this model, we could not have length-expanding decryption. Rather, our encryption process first is input part of the desired *ciphertext* and determines a corresponding part of what will be the decryption result (it is this step that provides length expansion); then it is input the payload plaintext and finally outputs the complete ciphertext, which includes the portion requested in the first input.

We describe the hybrid construction in section 2. Section 3 discusses appropriate security notions and gives provable security results for the construction.

This paper shows essentially how Mixmaster ([15], [16]) should have been specified to be cryptographically secure against active attacks. (Note that an entirely different model of operation for mix networks is assumed by Ohkubo and Abe [17] and Jakobsson and Juels [11]: these constructions assume the existence of a shared authenticated bulletin board, whereas in the present paper we are interested in an open system that can be used with point-to-point communication by e-mail or similar means.) We use many ideas and techniques described by Cramer and Shoup in [8], but adapt them to fit the new notions of security needed in the context of length-preserving mixes.

1.1 Notation

Strings are binary, i.e. elements of $\{0, 1\}^*$. The concatenation of strings s and t is denoted $s \parallel t$. The length of string s is $|s|$. For $|s| \geq w$, $\text{prefix}_w(s)$ denotes the string consisting of the leftmost w bits of s ; thus, $\text{prefix}_{|s|}(s \parallel t) = s$. Messages (plaintexts, ciphertexts) are strings.

Algorithms are usually probabilistic (randomized).

2 A Construction for Length-Preserving Mixes

Our construction allows much flexibility in the choice of cryptographic schemes, even in a single chain. The single parameter that must be fixed for all mixes is

the desired mix message length ℓ . Also it may be desireable to define a maximum length for the actual message payload, i.e. the part of the plaintext as recovered by the final mix that can be chosen by the sender: as a message proceeds through the mix chain, more and more of the data will be pseudo-random gibberish; the length of the useful part of the final plaintext reveals that chains leaving less than this amount cannot have been used. The length n of the mix chain need not be fixed.

For purposes of exposition, we number the mixes M_1, \dots, M_n according to their position in the chain chosen by the sender (note that the same mix might appear multiple times in one chain). For each mix M_i , the following must be defined and, with the exception of the secret key SK_{M_i} , known to senders who want to use the mix:

- A *key encapsulation mechanism*

$$KEM_{M_i}$$

and a key pair

$$(PK_{M_i}, SK_{M_i})$$

consisting of a *public key* PK_{M_i} and a *secret key* SK_{M_i} for this key encapsulation mechanism. A key encapsulation mechanism, similarly to a public-key encryption mechanism, provides an algorithm

$$KEM_{M_i}.\text{Encrypt}$$

using the public key and an algorithm

$$KEM_{M_i}.\text{Decrypt}$$

using the secret key. Associated with KEM_{M_i} is a *key generation algorithm*

$$KEM_{M_i}.\text{KeyGen}$$

that returns appropriate key pairs (PK, SK) . (Note that our formalization uses no explicit security parameter, so desired key sizes are implicit to $KEM_{M_i}.\text{KeyGen}$.) In contrast with public-key encryption, the `Encrypt` algorithm of a key encapsulation mechanism takes no input apart from the public key: $KEM_{M_i}.\text{Encrypt}(PK_{M_i})$ generates a ciphertext \mathfrak{K} of a fixed length

$$KEM_{M_i}.\text{CipherLen}$$

corresponding to a (pseudo-)random string K of a fixed length

$$KEM_{M_i}.\text{OutLen}$$

and outputs the pair (K, \mathfrak{K}) . Evaluation of $KEM_{M_i}.\text{Decrypt}(SK_{M_i}, \mathfrak{K})$ will return said string K if the key pair (PK_{M_i}, SK_{M_i}) has been produced by $KEM_{M_i}.\text{KeyGen}$. In this sense, ciphertext \mathfrak{K} encapsulates the random message K (which can be used as a key for symmetric-key cryptography). On

arbitrary inputs \mathfrak{K} , the computation $\text{KEM}_{M_i}.\text{Decrypt}(\text{SK}_{M_i}, \mathfrak{K})$ may either return some string K or fail (return the special value `invalid`).

One example of a key encapsulation mechanism is the Diffie-Hellman key exchange [9] with static keys for one party and ephemeral keys for the other party where the concatenation of the ephemeral public key with the common secret group element is hashed to obtain K (cf. [1], [19]); $\text{KEM}_{M_i}.\text{CipherLen}$ can be kept particularly small for elliptic curve Diffie-Hellman using just x coordinates of points (see [14]). Specifications for various key encapsulation mechanisms can be found in [19].

- A *one-time message authentication code*

$$\text{MAC}_{M_i}$$

with key length

$$\text{MAC}_{M_i}.\text{KeyLen}$$

and output length

$$\text{MAC}_{M_i}.\text{OutLen}.$$

A one-time message authentication code specifies an efficient deterministic algorithm that takes as input a key K of length $\text{MAC}_{M_i}.\text{KeyLen}$ and a bit string s and returns a string $\text{MAC}_{M_i}(K, s)$ of length $\text{MAC}_{M_i}.\text{OutLen}$. In our construction, MAC_{M_i} will only be used for strings s of fixed length $\ell - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}$.

Candidate one-time message authentication codes are UHASH [12]¹ and HMAC [3].

- A *pseudo-random bit string generator*

$$\text{STREAM}_{M_i}$$

taking as input a key K of length

$$\text{STREAM}_{M_i}.\text{KeyLen}$$

and deterministically generating an output string $\text{STREAM}_{M_i}(K)$ of length

$$\text{STREAM}_{M_i}.\text{OutLen}.$$

This will be used for an XOR-based stream cipher. A convenient example implementation is the so-called *counter mode* of a block cipher (the output sequence is the prefix of appropriate length of $E_K(0) \parallel E_K(1) \parallel E_K(2) \parallel \dots$ where E_K denotes block cipher encryption using key K ; see [4] and [13]).

¹ UHASH is the combination of a *key derivation function* with an *almost strongly universal hash function* [20] and is the core of UMAC as specified in [12]. Note that the security arguments provided for the earlier version of UMAC described in [5] are based on a different approach.

– An integer

$$\text{PlainLen}_{M_i}$$

specifying the length of the prefix of each decrypted message that is considered control data directed to mix M_{M_i} and will not be forwarded. (This is the amount of message expansion: the decrypted message minus the prefix must be of size ℓ because that is what will be sent to the next mix.)

The parameters must fulfil the following conditions:

$$\text{KEM}_{M_i}.\text{CipherLen} + \text{MAC}_{M_i}.\text{OutLen} + \text{PlainLen}_{M_i} < \ell \quad (1)$$

$$\text{KEM}_{M_i}.\text{OutLen} = \text{STREAM}_{M_i}.\text{KeyLen} + \text{MAC}_{M_i}.\text{KeyLen} \quad (2)$$

$$\text{STREAM}_{M_i}.\text{OutLen} = \text{PlainLen}_{M_i} + \ell \quad (3)$$

2.1 Encryption

We now describe encryption for sending a message through a chain M_1, \dots, M_n . Let *payload* be the message of length

$$|\text{payload}| = \ell - \sum_{1 \leq i \leq n} (\text{KEM}_{M_i}.\text{CipherLen} + \text{MAC}_{M_i}.\text{OutLen} + \text{PlainLen}_{M_i}) \quad (4)$$

(messages shorter than this maximum should be randomly padded on the right). For each i , let plain_i be the control message of length PlainLen_{M_i} directed to the respective mix. The encryption algorithm for these arguments is denoted

$$\text{chain-encrypt}_{M_1, \dots, M_n}(\text{plain}_1, \dots, \text{plain}_n; \text{payload})$$

or

$$\text{chain-encrypt}_{M_1, \dots, M_n}(\text{plain}_1, \dots, \text{plain}_n; \text{payload}; \lambda)$$

where λ is the empty string. The algorithm is defined recursively. Let $1 \leq i \leq n$, and let C_i be a string of length

$$|C_i| = \sum_{1 \leq k < i} (\text{KEM}_{M_k}.\text{CipherLen} + \text{MAC}_{M_k}.\text{OutLen} + \text{PlainLen}_{M_k}) \quad (5)$$

(thus specifically $|C_1| = 0$). Then algorithm

$$\text{chain-encrypt}_{M_i, \dots, M_n}(\text{plain}_i, \dots, \text{plain}_n; \text{payload}; C_i)$$

works as follows:

1. Use $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK}_{M_i})$ to generate a pair (K_i, \mathfrak{K}_i) .
2. Split K_i in the form

$$K_i = K_{i,\text{MAC}} \parallel K_{i,\text{STREAM}}$$

such that $|K_{i,\text{MAC}}| = \text{MAC}_{M_i}.\text{KeyLen}$ (so $|K_{i,\text{STREAM}}| = \text{STREAM}_{M_i}.\text{KeyLen}$ by (2)).

3. Compute $\text{STREAM}_{M_i}(K_{i,\text{STREAM}})$ and split this string in the form

$$\text{stream}_{i,\text{L}} \parallel \text{stream}_{i,\text{R}}$$

such that the left part $\text{stream}_{i,\text{L}}$ is of length

$$\ell - |C_i| - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}.$$

4. If $i = n$ (last mix), then by (4) and (5) it follows that $|\text{stream}_{n,\text{L}}| = \text{PlainLen}_n + |\text{payload}|$. In this case, set

$$\mathfrak{C}_n = (\text{stream}_{n,\text{L}} \oplus (\text{plain}_n \parallel \text{payload})) \parallel C_n.$$

Otherwise, let

$$C_{i+1} = \text{stream}_{i,\text{R}} \oplus (C_i \parallel 0^{\text{KEM}_{M_i}.\text{CipherLen} + \text{MAC}_{M_i}.\text{OutLen} + \text{PlainLen}_{M_i}}),$$

by recursion compute

$$x_i = \text{chain_encrypt}_{M_{i+1}, \dots, M_n}(\text{plain}_{i+1}, \dots, \text{plain}_n; \text{payload}; C_{i+1}),$$

and let

$$\mathfrak{C}_i = (\text{stream}_{i,\text{L}} \oplus (\text{plain}_i \parallel \text{prefix}_{|\text{stream}_{i,\text{L}}| - \text{PlainLen}_{M_i}}(x_i))) \parallel C_i.$$

5. Compute $\mathfrak{M}_i = \text{MAC}_i(K_{i,\text{MAC}}, \mathfrak{C}_i)$.
6. Return the ciphertext $\mathfrak{K}_i \parallel \mathfrak{M}_i \parallel \mathfrak{C}_i$, which is of length ℓ .

Appendix A.1 illustrates a ciphertext generated by $\text{chain_encrypt}_{M_1, \dots, M_n}$.

2.2 Decryption

The decryption algorithm for mix M_i ($1 \leq i \leq n$) works as follows, given a length- ℓ ciphertext $\mathfrak{K} \parallel \mathfrak{M} \parallel \mathfrak{C}$ (split into its three components according to parameters $\text{KEM}_{M_i}.\text{CipherLen} = |\mathfrak{K}|$ and $\text{MAC}_{M_i}.\text{OutLen} = |\mathfrak{M}|$). We denote it

$$\text{mix_decrypt}_{M_i}(\mathfrak{K} \parallel \mathfrak{M} \parallel \mathfrak{C}).$$

Remember that M_i in general is not aware of the mix chain used by the sender or even of its own position i in the chain.

1. Compute $K = \text{KEM}_{M_i}.\text{Decrypt}(\text{SK}_{M_i}, \mathfrak{K})$. If this computation fails, abort with an error (return invalid).
2. Split K in the form

$$K = K_{\text{MAC}} \parallel K_{\text{STREAM}}$$

such that $|K_{\text{MAC}}| = \text{MAC}_{M_i}.\text{KeyLen}$.

3. Compute

$$\widetilde{\mathfrak{M}} = \text{MAC}_{\mathbf{M}_i}(K_{\text{MAC}}, \mathfrak{C})$$

and test whether

$$\widetilde{\mathfrak{M}} = \mathfrak{M}.$$

If this is not the case, abort with an error (return *invalid*).

4. Compute the string

$$\textit{stream} = \text{STREAM}_{\mathbf{M}_i}(K_{\text{STREAM}})$$

of length $\text{STREAM}_{\mathbf{M}_i}.\text{OutLen}$.

5. Compute

$$\textit{stream} \oplus (\mathfrak{C} \parallel 0^{\text{KEM}_{\mathbf{M}_i}.\text{CipherLen} + \text{MAC}_{\mathbf{M}_i}.\text{OutLen} + \text{PlainLen}_{\mathbf{M}_i}})$$

and split the resulting string in the form

$$\textit{plain}_i \parallel P$$

where \textit{plain}_i is of length $\text{PlainLen}_{\mathbf{M}_i}$ (and thus, by (3), P is of length ℓ).

6. Return the pair (\textit{plain}_i, P) .

If this algorithm finishes without an error, \textit{plain}_i is a control message directed to the current mix, and P is the message to be forwarded to another mix or to the final recipient (as requested by the control message).

It is straightforward to verify that for ciphertexts computed as

$$\textit{chain-encrypt}_{\mathbf{M}_1, \dots, \mathbf{M}_n}(\textit{plain}_1, \dots, \textit{plain}_n; \textit{payload}),$$

iterative decryption by mixes $\mathbf{M}_1, \dots, \mathbf{M}_n$ will indeed work without an error and recover the respective strings \textit{plain}_i and finally also the message $\textit{payload}$ concatenated with some (useless) extra data. Appendix A.2 illustrates decryption.

3 Provable Security

The mix concept is intended to provide security when at least one mix can be trusted and behaves correctly. (However note that denial-of-service attacks by incorrectly operating mixes cannot be ruled out; the only option is to avoid mixes that appear to malfunction.) Thus we assume that some single mix \mathbf{M}_i works as expected while all other mixes are controlled by the adversary and may not follow the protocol (also the cryptographic schemes $\text{KEM}_{\mathbf{M}_j}$, $\text{MAC}_{\mathbf{M}_j}$, and $\text{STREAM}_{\mathbf{M}_j}$ associated with mixes \mathbf{M}_j , $j \neq i$, might be not secure, and $\text{KEM}_{\mathbf{M}_j}$ might even not be a valid key encapsulation mechanism according to the description given in section 2).

This leaves only \mathbf{M}_i to be attacked: outer layers of encryption for mixes that appear before \mathbf{M}_i in a mix chain are easily removed by the adversary and thus

are not relevant for security; and inner layers of encryption for mixes that appear after M_i in a mix chain are involved in the encryption process

$$\text{chain-encrypt}_{M_1, \dots, M_n}(\text{plain}_1, \dots, \text{plain}_n; \text{payload}; C_i)$$

described in section 2.1, but cannot provide protection against the adversary.

An important security property for mixes is *unlinkability*: an adversary who observes a batch of encrypted messages as these are sent to a mix and who sees the resulting decrypted messages when these are forwarded should not be able to tell better than by chance which decrypted messages correspond to which encrypted messages.

Beyond this basic security notion, we also want to achieve security against active attacks. As noted in the introduction, it is crucial to detect message replay and ignore replayed message. However, this does not rule out active attacks based on message modifications. To show that our construction for length-preserving mixes is secure against this kind of attacks, we will model an active adversary who is able to launch an *adaptive chosen ciphertext attack (CCA)*.

We point out that in the model of operation that we assume, unlinkability cannot be fully guaranteed in the presence of active attacks: an active adversary who suppresses all but a single one of the legitimate encrypted messages in a batch and substitutes new messages for them can easily trace the remaining legitimate message. This can be avoided only heuristically (each sender of mix messages should occasionally route a message to himself to see if it gets through). To avoid related flooding attacks, mixes should not start a new batch whenever a certain fixed number of messages has arrived; instead, each batch should be kept open for additional messages during some time frame.

We show how the security of the mix construction can be captured formally by describing unlinkability (section 3.1) and CCA security (section 3.2). We then provide security definitions for the underlying key encapsulation method (section 3.3), one-time message authentication code (section 3.4), and pseudo-random bit string generator (section 3.5). Finally, section 3.6 present security results that relates the security of the mix encryption scheme to the security of these cryptographic schemes: we will see that if the mix encryption scheme is not secure, then this is because one of the underlying cryptographic schemes is not secure.

3.1 Unlinkability of the Mix Encryption Scheme

An intuitive approach to defining unlinkability is as follows: the adversary sees a batch of m ciphertexts and a random permutation of the resulting m decryption results; for each decryption result, the adversary makes a guess on the index of the corresponding ciphertext. With random guessing, the expected number of correct guesses is one. An adversary who can do better than this breaks unlinkability.

More specifically, we let the adversary select the plaintexts as far as possible (remember that as we have length-expanding decryption, it is not possible to

choose all of the decryption result arbitrarily when generating a ciphertext). In the formal definition, we confine to a setting with two plaintexts where only one ciphertext is shown to the adversary (an adversary in the unrestricted setting can be used to build an adversary in this setting). This is captured in the following attack game:

1. The adversary queries a *key generation oracle*, which uses $\text{KEM}_{M_i}.\text{KeyGen}$ to compute a key pair

$$(\text{PK}, \text{SK})$$

and responds with PK (and secretly stores SK).

2. The adversary uses an *encryption oracle* as follows (cf. the encryption algorithm from section 2.1 in the case of a length-1 mix chain, i.e. with no recursion and $|C_i| = 0$):

- The adversary submits a pair of string pairs

$$((\text{plain}_{i,0}, m_0), (\text{plain}_{i,1}, m_1))$$

where

$$|\text{plain}_{i,b}| = \text{PlainLen}_{M_i}$$

and

$$|m_b| = \ell - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen} - \text{PlainLen}_{M_i}$$

for $b = 0, 1$.

- The encryption oracle chooses a bit $b \in \{0, 1\}$ uniformly at random. Then it uses $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK})$ to generate a pair $(K_{\text{oracle}}, \mathfrak{K}_{\text{oracle}})$ and splits K_{oracle} in the form

$$K_{\text{oracle}} = K_{\text{MAC}} \parallel K_{\text{STREAM}}$$

such that $|K_{\text{MAC}}| = \text{MAC}_{M_i}.\text{KeyLen}$; it computes $\text{STREAM}_{M_i}(K_{\text{STREAM}})$ and splits the resulting string *stream* in the form

$$\text{stream} = \text{stream}_L \parallel \text{stream}_R$$

such that $|\text{stream}_L| = \ell - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}$; and it computes

$$\mathfrak{C} = \text{stream}_L \oplus (\text{plain}_{i,b} \parallel m_b)$$

and

$$\mathfrak{M} = \text{MAC}_{M_i}(K_{\text{MAC}}, \mathfrak{C}).$$

Then it outputs the pair

$$(\mathfrak{K}_{\text{oracle}} \parallel \mathfrak{M} \parallel \mathfrak{C}, \text{stream}_R).$$

3. The adversary outputs a bit $\tilde{b} \in \{0, 1\}$.

The bit \tilde{b} output by the adversary is its guess for the value of b . Note that the decryption result corresponding to the ciphertext $K_{\text{oracle}} \parallel M \parallel C$ is

$$(plain_{i,b}, m_b \parallel stream_R)$$

(see section 2.2), and all of this except for the choice of the bit b is known to the adversary.

Let A be any adversary (interactive probabilistic algorithm with bounded runtime) in this attack game. Its *advantage against unlinkability* for M_i 's instantiation of the mix encryption scheme is

$$\text{AdvLink}_{M_i, A} = \left| \Pr[\tilde{b} = 1 \mid b = 1] - \Pr[\tilde{b} = 1 \mid b = 0] \right|.$$

3.2 CCA Security of the Mix Encryption Scheme

Due to the recursive nature of our encryption algorithm for mix chains, we cannot directly apply the usual definitions of security under adaptive chosen ciphertext attack (CCA) for ordinary public-key encryption. We adapt the attack game described in [8, section 3.2] (which goes back to [10] and [18]) as follows to take into account the special properties of our construction:

1. The adversary queries a *key generation oracle*, which uses $\text{KEM}_{M_i}.\text{KeyGen}$ to compute a key pair

$$(\text{PK}, \text{SK})$$

and responds with PK (and secretly stores SK).

2. The adversary makes a sequence of queries to a *decryption oracle*. Each query is an arbitrary string s of length ℓ , and the oracle responds with

$$mix_decrypt_{M_i}(s),$$

using the secret key SK from step 1. That is, each oracle response is either a pair $(plain, P)$ where $|plain| = \text{PlainLen}_{M_i}$ and $|P| = \ell$, or the special value *invalid*.

3. The adversary uses an *interactive encryption oracle* as follows (compare with the encryption algorithm in section 2.1):

- First the adversary submits some string C_i subject only to the condition that

$$0 \leq |C_i| < \ell - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}.$$

- The interactive encryption oracle uses $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK})$ to generate a pair $(K_{\text{oracle}}, K_{\text{oracle}})$ and splits K_{oracle} in the form

$$K_{\text{oracle}} = K_{\text{MAC}} \parallel K_{\text{STREAM}}$$

such that $|K_{\text{MAC}}| = \text{MAC}_{M_i}.\text{KeyLen}$; it computes $\text{STREAM}_{M_i}(K_{\text{STREAM}})$ and splits the resulting string $stream$ in the form

$$stream = stream_L \parallel stream_R$$

such that $|stream_L| = \ell - |C_i| - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}$; and it computes

$$C_{i+1} = stream_R \oplus (C_i \parallel 0^{\text{KEM}_{M_i}.\text{CipherLen} + \text{MAC}_{M_i}.\text{OutLen} + \text{PlainLen}_{M_i}})$$

and sends C_{i+1} to the adversary.

- The adversary submits a pair of string pairs

$$((plain_{i,0}, m_0), (plain_{i,1}, m_1))$$

satisfying

$$|plain_{i,b}| = \text{PlainLen}_{M_i}$$

and

$$|m_b| = \ell - |C_i| - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen} - \text{PlainLen}_{M_i}$$

for $b = 0, 1$.

- The interactive encryption oracle chooses a uniformly random bit $b \in \{0, 1\}$, determines

$$\mathfrak{C} = (stream_L \oplus (plain_{i,b} \parallel m_b)) \parallel C_i$$

and

$$\mathfrak{M} = \text{MAC}_{M_i}(K_{\text{MAC}}, \mathfrak{C}),$$

and responds with the *challenge ciphertext*

$$\mathfrak{K}_{\text{oracle}} \parallel \mathfrak{M} \parallel \mathfrak{C}.$$

4. The adversary again makes a sequence of queries to a *decryption oracle* as in step 2, except that this time the decryption oracle refuses being asked for the challenge ciphertext $\mathfrak{K}_{\text{oracle}} \parallel \mathfrak{M} \parallel \mathfrak{C}$ from step 3 (it returns invalid for this case).
5. The adversary outputs a bit $\tilde{b} \in \{0, 1\}$.

The bit \tilde{b} output by the adversary is its guess for the value of b .

The essential difference to the adaptive chosen ciphertext attack game for ordinary public-key encryption is that we have made the encryption oracle interactive to reflect the recursiveness of the encryption algorithm for mix chains, where encryption to mixes later in the chain than M_i can have potentially arbitrary effects on a large part of the plaintext. Appendix A.3 illustrates the ciphertext resulting from the invocation of the interactive encryption oracle.

Let A be any adversary (interactive probabilistic algorithm with bounded runtime) in the above attack game. Its *CCA advantage* against M_i 's instantiation of the mix encryption scheme is

$$\text{AdvCCA}_{M_i, A} = \left| \Pr[\tilde{b} = 1 \mid b = 1] - \Pr[\tilde{b} = 1 \mid b = 0] \right|.$$

3.3 Security of the Key Encapsulation Mechanism

CCA security for the key encryption mechanism KEM_{M_i} is defined through the following attack game (cf. [8, section 7.1.2]):

1. The adversary queries a *key generation oracle*, which uses $\text{KEM}_{M_i}.\text{KeyGen}$ to compute a key pair (PK, SK) and responds with PK (and secretly stores SK).
2. The adversary makes a sequence of queries to a *decryption oracle*. Each query is an arbitrary string s of length $\text{KEM}_{M_i}.\text{CipherLen}$; the oracle responds with

$$\text{KEM}_{M_i}.\text{Decrypt}(\text{SK}, s).$$

Thus each oracle response is either a string of length $\text{KEM}_{M_i}.\text{OutLen}$ or the special value *invalid*.

3. The adversary queries an *encryption oracle*, which works as follows: it uses $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK})$ to obtain a pair $(K_0, \mathfrak{K}_{\text{oracle}})$, it generates a uniformly random string K_1 such that $|K_0| = |K_1|$, chooses a uniformly random bit $b_{\text{KEM}} \in \{0, 1\}$, and responds with $(K_{b_{\text{KEM}}}, \mathfrak{K}_{\text{oracle}})$.
4. The adversary again makes a sequence of queries to a *decryption oracle* as in step 2, but this time the oracle refuses the specific query $\mathfrak{K}_{\text{oracle}}$ and responds *invalid* in this case.
5. The adversary outputs a bit $\tilde{b}_{\text{KEM}} \in \{0, 1\}$.

The *CCA advantage* of an adversary A (an interactive probabilistic algorithm with bounded runtime) against KEM_{M_i} in this attack game is

$$\text{AdvCCA}_{\text{KEM}_{M_i}, A} = \left| \Pr[\tilde{b}_{\text{KEM}} = 1 \mid b_{\text{KEM}} = 1] - \Pr[\tilde{b}_{\text{KEM}} = 1 \mid b_{\text{KEM}} = 0] \right|.$$

3.4 Security of the One-Time Message Authentication Code

To define the security of MAC_{M_i} , we use the following attack game (cf. [8, section 7.2.2]):

1. The adversary submits a string s (which for our purposes may be assumed to be of fixed length $\ell - \text{KEM}_{M_i}.\text{CipherLen} - \text{MAC}_{M_i}.\text{OutLen}$) to an oracle. The oracle generates a uniformly random string K of length $\text{MAC}_{M_i}.\text{KeyLen}$ and responds with $\text{MAC}_{M_i}(K, s)$.
2. The adversary outputs a list $(s_1, t_1), (s_2, t_2), \dots, (s_m, t_m)$ of pairs of strings.

An adversary A again is an interactive probabilistic algorithm with bounded runtime; the runtime bound implies a bound on the list length m . We say that adversary A has *produced a forgery* if $s_k \neq s$ and $\text{MAC}_{M_i}(K, s_k) = t_k$ for some k ($1 \leq k \leq m$). Its advantage against MAC_{M_i} , denoted $\text{AdvForge}_{\text{MAC}_{M_i}, A}$, is the probability that it produces a forgery in the above game.

3.5 Security of the Pseudo-Random Bit String Generator

To define the security of STREAM_{M_i} , we use the following attack game:

1. The adversary queries an oracle. The oracle generates a uniformly random string K of length $\text{STREAM}_{M_i}.\text{KeyLen}$, computes $stream_0 = \text{STREAM}_{M_i}(K)$, generates a uniformly random string $stream_1$ with $|stream_0| = |stream_1|$, chooses a uniformly random bit $b_{\text{STREAM}} \in \{0, 1\}$, and responds with $stream_b$.
2. The adversary outputs a bit $\tilde{b}_{\text{STREAM}} \in \{0, 1\}$.

The advantage of an adversary A (again an interactive probabilistic algorithm with bounded runtime) against STREAM_{M_i} is

$$\begin{aligned} & \text{Adv}_{\text{STREAM}_{M_i}, A} \\ &= \left| \Pr[\tilde{b}_{\text{STREAM}} = 1 \mid b_{\text{STREAM}} = 1] - \Pr[\tilde{b}_{\text{STREAM}} = 1 \mid b_{\text{STREAM}} = 0] \right|. \end{aligned}$$

3.6 Security Results

First let A be an adversary against the unlinkability of the mix encryption scheme, attacking a mix M_i as described in section 3.1. It can be shown that there are adversaries A_1 and A_2 against KEM_{M_i} and STREAM_{M_i} , respectively, with essentially the same runtime as A such that

$$\text{AdvLink}_{M_i, A} \leq 2 \cdot (\text{AdvCCA}_{\text{KEM}_{M_i}, A_1} + \text{Adv}_{\text{STREAM}_{M_i}, A_2}).$$

Details of the proof can be found in appendix B.1.

Now let A be an adversary against the CCA security of the mix encryption scheme, attacking a mix M_i as described in section 3.2. It can be shown that there are adversaries A_1, A_2, A_3 against $\text{KEM}_{M_i}, \text{MAC}_{M_i}, \text{STREAM}_{M_i}$ all having essentially the same runtime as A such that

$$\text{AdvCCA}_{M_i, A} \leq 2 \cdot (\text{AdvCCA}_{\text{KEM}_{M_i}, A_1} + \text{AdvForge}_{\text{MAC}_{M_i}, A_2} + \text{Adv}_{\text{STREAM}_{M_i}, A_3}).$$

Details of the proof are given in appendix B.2.

Acknowledgement

David Hopwood and the anonymous reviewers provided valuable comments on earlier versions of this paper.

References

- [1] ABDALLA, M., BELLARE, M., AND ROGAWAY, P. DHAES: An encryption scheme based on the Diffie-Hellman problem. Submission to IEEE P1363a. <http://grouper.ieee.org/groups/1363/P1363a/Encryption.html>, 1998. 246, 248
- [2] ABDALLA, M., BELLARE, M., AND ROGAWAY, P. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *Progress in Cryptology - CT-RSA 2001* (2001), D. Naccache, Ed., vol. 2020 of *Lecture Notes in Computer Science*, pp. 143–158. 246
- [3] BELLARE, M., CANETTI, R., AND KRAWCZYK, H. Keying hash functions for message authentication. In *Advances in Cryptology – CRYPTO ’96* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, pp. 1–15. 248
- [4] BELLARE, M., DESAI, A., JOKIPPI, E., AND ROGAWAY, P. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science (FOCS ’97)* (1997), IEEE Computer Society, pp. 394–403. 248
- [5] BLACK, J., HALEVI, S., KRAWCZYK, H., KROVETZ, T., AND ROGAWAY, P. UMAC: Fast and secure message authentication. In *Advances in Cryptology – CRYPTO ’99* (1999), M. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, pp. 216–233. 248
- [6] CHAUM, D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24 (1981), 84–88. 244, 245
- [7] COTTRELL, L. Mixmaster & remailer attacks. <http://www.obscura.com/%7Eloki/remailer/remailer-essay.html>, 1997. 245
- [8] CRAMER, R., AND SHOUP, V. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. Manuscript, <http://shoup.net/papers/>, 2001. 246, 254, 256
- [9] DIFFIE, W., AND HELLMAN, M. E. New directions in cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976), 644–654. 248
- [10] GOLDWASSER, S., AND MICALI, S. Probabilistic encryption. *Journal of Computer and System Sciences* 28 (1984), 270–299. 254
- [11] JAKOBSSON, M., AND JUELS, A. An optimally robust hybrid mix network. In *20th Annual ACM Symposium on Principles of Distributed Computing (PODC 2001)* (2001), ACM Press, pp. 284–292. 246
- [12] KROVETZ, T., BLACK, J., HALEVI, S., HEVIA, A., KRAWCZYK, H., AND ROGAWAY, P. UMAC: Message authentication code using universal hashing. Internet-Draft *draft-krovetz-umac-01.txt*, <http://www.cs.ucdavis.edu/~rogaway/umac/>, 2000. 248
- [13] LIPMAA, H., ROGAWAY, P., AND WAGNER, D. Comments to NIST concerning AES modes of operation: CTR-mode encryption. <http://csrc.nist.gov/encryption/modes/workshop1/papers/lipmaa-ctr.pdf>, 2000. 248
- [14] MILLER, V. S. Use of elliptic curves in cryptography. In *Advances in Cryptology – CRYPTO ’85* (1986), H. C. Williams, Ed., vol. 218 of *Lecture Notes in Computer Science*, pp. 417–428. 248
- [15] Mixmaster anonymous remailer software. <http://sourceforge.net/projects/mixmaster/>. 245, 246
- [16] MÖLLER, U., AND COTTRELL, L. Mixmaster protocol version 2. <http://www.eskimo.com/~rowdenw/crypt/Mix/draft-moeller-v2-01.txt>, 2000. 246

- [17] OHKUBO, M., AND ABE, M. A length-invariant hybrid mix. In *Advances in Cryptology – ASIACRYPT 2000* (2000), T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, pp. 178–191. [246](#)
- [18] RACKOFF, C. W., AND SIMON, D. R. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In *Advances in Cryptology – CRYPTO ’91* (1992), J. Feigenbaum, Ed., vol. 576 of *Lecture Notes in Computer Science*, pp. 433–444. [254](#)
- [19] SHOUP, V. A proposal for an ISO standard for public key encryption. Version 2.1, December 20, 2001. <http://shoup.net/papers/>. [248](#)
- [20] WEGMAN, M. N., AND CARTER, J. L. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences* 22 (1981), 265–279. [248](#)

A Illustrations

A.1 Encryption

We depict a ciphertext generated by the encryption algorithm

$$\text{chain-encrypt}_{M_1, M_2, M_3}(\text{plain}_1, \text{plain}_2, \text{plain}_3; \text{payload})$$

as described in section 2.1. In the illustration, we have concatenation horizontally and XOR vertically (i.e. boxes in the same row represent bit strings that are concatenated, and the ciphertext is the XOR of the multiple rows shown).

			plain_3	payload
			plain_2	\mathfrak{K}_3
			\mathfrak{M}_3	$\text{stream}_{3,L}$
plain_1	\mathfrak{K}_2	\mathfrak{M}_2		$\text{stream}_{2,L}$
\mathfrak{K}_1	\mathfrak{M}_1			$\text{stream}_{1,L}$

A.2 Decryption

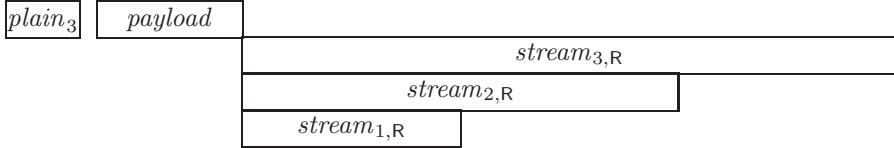
The result obtained by mix M_1 when applying the decryption algorithm from section 2.2 to the ciphertext from appendix A.1 is composed as follows:

			plain_3	payload
			plain_2	\mathfrak{K}_3
			\mathfrak{M}_3	$\text{stream}_{3,L}$
plain_1	\mathfrak{K}_2	\mathfrak{M}_2		$\text{stream}_{2,L}$
				$\text{stream}_{1,R}$

The string plain_1 is directed to M_1 . The remainder of the decryption result is a ciphertext that should be forwarded to the next mix in the chain, M_2 , which will then obtain the following result:

			plain_3	payload
			plain_2	\mathfrak{K}_3
			\mathfrak{M}_3	$\text{stream}_{3,L}$
				$\text{stream}_{2,R}$
				$\text{stream}_{1,R}$

Similarly, mix M_3 will obtain the following final decryption result:



A.3 Interactive Encryption Oracle

In the adaptive chosen ciphertext attack game from section 3.2, recursive encryption is replaced by an interactive encryption oracle to generate the challenge ciphertext, which has the following structure:

$\mathfrak{K}_{\text{oracle}}$	$plain_{i,b}$	m_b	C_i
\mathfrak{M}	$stream_L$		

Applying algorithm $mix_decrypt_{M_i}$ from section 2.2 to this ciphertext would yield the following result:

$plain_{i,b}$	m_b	C_i	$stream_R$
---------------	-------	-------	------------

This can also be written as follows:

$plain_{i,b}$	m_b	C_{i+1}
---------------	-------	-----------

B Security Proofs

B.1 Unlinkability

We prove the security result from section 3.6 for the unlinkability of the mix encryption scheme. Let \mathbf{G}_0 denote the attack game from section 3.1. Let \mathbf{G}_1 be like \mathbf{G}_0 except that a uniformly random string is used for K_{oracle} , whereas $\mathfrak{K}_{\text{oracle}}$ is still generated by $\text{KEM}_{M_i}.\text{Encrypt}(PK)$. Let \mathbf{G}_2 be like \mathbf{G}_1 except that the encryption oracle uses a uniformly random string $stream$ instead of computing $\text{STREAM}_{M_i}(K_{\text{STREAM}})$.

Now we consider an adversary A as in section 3.1, exposed to these different attack games \mathbf{G}_x , $0 \leq x \leq 2$, and look at the respective success probabilities $\Pr_{\mathbf{G}_x}[\tilde{b} = b]$. Based on A , adversaries A_1 and A_2 against KEM_{M_i} and STREAM_{M_i} , respectively, will be built, each with essentially the same runtime as A .

A_1 attacks KEM_{M_i} as follows (cf. section 3.3; note that this A_1 never actually uses its decryption oracle). At first, it generates $b \in \{0, 1\}$ uniformly at random. Then, it runs the adversary A ; when A queries its encryption oracle, A_1 queries its own encryption oracle to obtain a pair $(K_{\text{oracle}}, \mathfrak{K}_{\text{oracle}})$ and performs step 2

from section 3.1 using this pair and the pregenerated bit b . Finally, when A outputs its bit \tilde{b} , A_1 outputs 1 if $\tilde{b} = b$ and 0 otherwise. Observe that

$$\left| \Pr_{\mathbf{G}_1} [\tilde{b} = b] - \Pr_{\mathbf{G}_0} [\tilde{b} = b] \right| = \text{AdvCCA}_{\text{KEM}_{M_i}, A_1}$$

(\mathbf{G}_0 corresponds to $b_{\text{KEM}} = 0$, \mathbf{G}_1 corresponds to $b_{\text{KEM}} = 1$ in section 3.3).

A_2 attacks STREAM_{M_i} as follows (cf. section 3.5). First, it generates $b \in \{0, 1\}$ and a string K_{oracle} of length $\text{KEM}_{M_i}.\text{OutLen}$ uniformly at random. Then, it runs the adversary A , playing the role of the key generation oracle and the role of the encryption oracle, wherein it substitutes the pregenerated string K_{oracle} and the pregenerated bit b in step 2 of section 3.1. When A outputs its bit \tilde{b} , A_2 outputs 1 if $\tilde{b} = b$ and 0 otherwise. Observe that

$$\left| \Pr_{\mathbf{G}_2} [\tilde{b} = b] - \Pr_{\mathbf{G}_1} [\tilde{b} = b] \right| = \text{Adv}_{\text{STREAM}_{M_i}, A_2}$$

(\mathbf{G}_1 corresponds to $b_{\text{STREAM}} = 0$, \mathbf{G}_2 corresponds to $b_{\text{STREAM}} = 1$ in section 3.5).

Finally observe that $\Pr_{\mathbf{G}_2} [\tilde{b} = b] = \frac{1}{2}$.

From this we obtain the inequality

$$\begin{aligned} \text{AdvLink}_{M_i, A} &= 2 \cdot \left| \frac{1}{2} - \Pr_{\mathbf{G}_0} [\tilde{b} = b] \right| \\ &= 2 \cdot \left| \sum_{1 \leq x \leq 2} \left(\Pr_{\mathbf{G}_x} [\tilde{b} = b] - \Pr_{\mathbf{G}_{x-1}} [\tilde{b} = b] \right) \right| \\ &\leq 2 \cdot (\text{AdvCCA}_{\text{KEM}_{M_i}, A_1} + \text{Adv}_{\text{STREAM}_{M_i}, A_2}), \end{aligned}$$

which concludes the proof.

B.2 CCA Security

We prove the CCA security result given in section 3.6. Let \mathbf{G}_0 denote the attack game from section 3.2. We will modify it in multiple steps, essentially disabling the underlying cryptographic schemes (key encryption method, one-time message authentication code, and pseudo-random bit string generator) one after another.

\mathbf{G}_1 is like \mathbf{G}_0 except that a uniformly random string is used for K_{oracle} , whereas $\mathfrak{K}_{\text{oracle}}$ is still generated by $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK})$. In the decryption oracle, K is substituted whenever $\text{KEM}_{M_i}.\text{Decrypt}(\text{SK}, \mathfrak{K}_{\text{oracle}})$ would have to be computed.

\mathbf{G}_2 is like \mathbf{G}_1 except that the decryption oracle always responds with invalid when faced with any query prefixed with string $\mathfrak{K}_{\text{oracle}}$. (This applies to both step 2 and step 4 in section 3.2. Thus, the invocation of $\text{KEM}_{M_i}.\text{Encrypt}(\text{PK})$ to generate $(K_{\text{oracle}}, \mathfrak{K}_{\text{oracle}})$ must be advanced from step 3 to an earlier step; in \mathbf{G}_1 , this is only a descriptive change and does not affect the behaviour observed by the adversary.)

\mathbf{G}_3 is like \mathbf{G}_2 except that the interactive encryption oracle uses a uniformly random string *stream* instead of computing $\text{STREAM}_{M_i}(K_{\text{STREAM}})$.

Now we consider an adversary A as in section 3.2, exposed to these different attack games \mathbf{G}_x , $0 \leq x \leq 3$, and look at the respective success probabilities $\Pr_{\mathbf{G}_x}[\tilde{b} = b]$. Based on A , adversaries A_1 , A_2 , A_3 against KEM_{M_i} , MAC_{M_i} , STREAM_{M_i} will be built all having essentially the same runtime as A .

A_1 attacks KEM_{M_i} as follows (cf. section 3.3). At first, it generates $b \in \{0, 1\}$ uniformly at random. Then, it runs the adversary A ; when A queries its decryption oracle, A_1 uses its own decryption oracle to perform the decryption algorithm from section 2.2, and when A queries its interactive encryption oracle, A_1 queries its own encryption oracle to obtain a pair $(K_{\text{oracle}}, \mathfrak{K}_{\text{oracle}})$ and performs step 3 from section 3.2 using this pair and the pregenerated bit b . Finally, when A outputs its bit \tilde{b} , A_1 outputs 1 if $\tilde{b} = b$ and 0 otherwise. Observe that

$$\left| \Pr_{\mathbf{G}_1}[\tilde{b} = b] - \Pr_{\mathbf{G}_0}[\tilde{b} = b] \right| = \text{AdvCCA}_{\text{KEM}_{M_i}, A_1}$$

(\mathbf{G}_0 corresponds to $b_{\text{KEM}} = 0$, \mathbf{G}_1 corresponds to $b_{\text{KEM}} = 1$ in section 3.3).

A_2 attacks MAC_{M_i} as follows (cf. section 3.4). At first, it generates $b \in \{0, 1\}$ and a string K_{oracle} of length $\text{KEM}_{M_i}.\text{OutLen}$ uniformly at random. Then, it runs the adversary A , playing the roles of the key generation oracle, decryption oracle, and interactive encryption oracle, substituting the pregenerated string K_{oracle} and the pregenerated bit b in step 3 of section 3.2. Whenever A submits a query $\mathfrak{K} \parallel \mathfrak{M} \parallel \mathfrak{C}$ to the decryption oracle, A_2 adds the pair $(\mathfrak{C}, \mathfrak{M})$ to its own output (A 's final output bit \tilde{b} is ignored). Observe that

$$\left| \Pr_{\mathbf{G}_2}[\tilde{b} = b] - \Pr_{\mathbf{G}_1}[\tilde{b} = b] \right| \leq \text{AdvForge}_{\text{MAC}_{M_i}, A_2}$$

(\mathbf{G}_2 behaves differently from \mathbf{G}_1 only if a forgery has been produced).

A_3 attacks STREAM_{M_i} as follows (cf. section 3.5). First, it generates $b \in \{0, 1\}$ and a string K_{oracle} of length $\text{KEM}_{M_i}.\text{OutLen}$ uniformly at random. Then, it runs the adversary A , playing the roles of the key generation oracle, decryption oracle, and interactive encryption oracle, substituting the pregenerated string K_{oracle} and the pregenerated bit b in step 3 of section 3.2. When A outputs its bit \tilde{b} , A_3 outputs 1 if $\tilde{b} = b$ and 0 otherwise. Observe that

$$\left| \Pr_{\mathbf{G}_3}[\tilde{b} = b] - \Pr_{\mathbf{G}_2}[\tilde{b} = b] \right| = \text{AdvSTREAM}_{M_i, A_3}$$

(\mathbf{G}_2 corresponds to $b_{\text{STREAM}} = 0$, \mathbf{G}_3 corresponds to $b_{\text{STREAM}} = 1$ in section 3.5).

Finally observe that $\Pr_{\mathbf{G}_3}[\tilde{b} = b] = \frac{1}{2}$.

From this we obtain the inequality

$$\begin{aligned} \text{AdvCCA}_{M_i, A} &= 2 \cdot \left| \frac{1}{2} - \Pr_{\mathbf{G}_0}[\tilde{b} = b] \right| \\ &= 2 \cdot \left| \sum_{1 \leq x \leq 3} \left(\Pr_{\mathbf{G}_x}[\tilde{b} = b] - \Pr_{\mathbf{G}_{x-1}}[\tilde{b} = b] \right) \right| \\ &\leq 2 \cdot (\text{AdvCCA}_{\text{KEM}_{M_i}, A_1} + \text{AdvForge}_{\text{MAC}_{M_i}, A_2} + \text{AdvSTREAM}_{M_i, A_3}), \end{aligned}$$

which concludes the proof.

Fault Tolerant and Distributed Broadcast Encryption

Paolo D'Arco¹ and Douglas R. Stinson²

¹ Dipartimento di Informatica ed Applicazioni, Università degli Studi di Salerno
84081, Baronissi (SA), Italy
paodar@dia.unisa.it

² School of Computer Science, University of Waterloo
N2L 3G1, Waterloo Ontario, Canada
dstinson@cacr.math.uwaterloo.ca

Abstract. A broadcast encryption scheme enables a server to broadcast information in a secure way over an insecure channel to an arbitrary subset of privileged recipients. In a set-up phase, the server gives pre-defined keys to every user of the system, using secure point-to-point channels. Later on, it broadcasts an encrypted message along a broadcast channel, in such a way that only users in a privileged subset can decrypt it, by using the pre-defined keys received in set-up phase. Usually, the broadcast message contains a fresh session key, which can subsequently be used for secure broadcast transmission to the privileged set of recipients. In this paper we deal with two aspects of secure broadcast transmission: *reliability* and *trust* in the broadcaster. The first is a well-studied issue in communication over unreliable channels: packets can get lost and some redundancy is required to provide reliable communication. The second aspect concerns with the assumption that the broadcaster, who receives information for broadcasting from several entities, must be trusted. This issue has not previously been addressed in the broadcast transmission setting. We provide a motivating scenario in which the assumption does not hold and, for both problems, we review and extend some existing broadcast encryption schemes, in order to gain fault tolerance and to remove the need for trust in the broadcaster.

1 A Motivating Scenario

A movie company is willing to distribute its re-make of *Gone with the Wind*, and it is looking for an agreement with a pay-per-view broadcaster. The broadcaster uses a broadcast encryption scheme to distribute a common key to the users who have subscribed to the service, and later it broadcasts an encrypted version of the movie that can be decrypted only by the users who had received the common key when running the broadcast encryption scheme. Therefore, only users who have paid can enjoy the movie.

However, since people have been waiting for a long time for this re-make of the famous movie, several broadcasters would like to sign a contract with the movie company. It is supposed to be a good deal! Aware of that, the company,

in its contract proposal, asks for a high percentage of the money paid by the users to subscribe to the broadcast service. One of the broadcasters, close to bankruptcy, accepts the offer, even if he thinks the contract is not fair to him. Therefore he decides, in order to increase his income, to sell illegally, on the “black market”, a copy of the movie that will later be re-distributed by other means.

Notice that the broadcaster is almost completely safe: he could always say that some user has stored and illegally re-distributed the movie. There is no way to establish who is guilty, even if some watermark has been embedded by the company into the digital representation of the movie¹.

The main problem of the above transmission model is that the broadcaster gets a copy *in clear* of the movie. So he must be completely trusted by the movie company.

The question we are going to study in this paper is the following: Is there any way by means of which the company can broadcast the movie without trusting the broadcaster? And, actually, a more general question is: is it possible to design a system by means of which an entity can delegate another one to *blindly broadcast* information in a secure way to a given set of users?

A trivial solution could be that the broadcaster just receives an encrypted copy of the movie, and the company runs a broadcast encryption scheme (BES, for short) with the users in order to give them the decryption key. But this means that the company must be involved directly in transmission issues and must interact with millions of users.

The idea we would like to investigate is the following: instead of a single server, there are n servers across the system. The movie company, in a set-up phase, gives to each server an encrypted partial version of the movie and some key material. Later on, the n servers run a protocol with the users, enabling a given subset of them to recover the decryption key. Finally, each server starts broadcasting the encrypted partial version of the movie he got from the company. The privileged users, by means of the broadcasts and of the decryption key, recover and watch the original movie. Notice that, with this approach, the movie company only needs to communicate with a fixed and *small number* of servers, compared to the scenario in which the company directly manages the key distribution with the users. Moreover, a single server does not get any information about the movie.

¹ The Digital Watermarking Technology in recent years has produced plenty of possible solutions for copyright related problems. We point out that the notion of a distributed broadcast encryption scheme, which we are going to introduce in the following, is not a substitute for such schemes and it does not address the same problems. The problem it solves is how to remove trust in a third party responsible for the transmission of some valuable information. Maybe, in some settings or in the design of a complex architecture, such a scheme can be used in conjunction with digital watermarking schemes, but we stress that the goals are different.

A small example. To exemplify the idea we are thinking about, let us consider the following simple protocol. Let $\mathcal{U} = \{U_1, \dots, U_m\}$ be a set of m users and let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a set of n servers. The value n is relatively small ($n < 10$, say) but m can be very large (e.g., $m > 1000000$). Every user has a secure *point-to-point channel* with every server. Servers and users have access to a common *broadcast channel*. Let Σ be a broadcast encryption scheme. A distributed solution for the movie problem works as follows:

Set-up Phase.

1. The movie company chooses a random key k .
2. Then it uses a (t, n) secret sharing scheme to compute n shares of k , say y_1, \dots, y_n .
3. At the same time, every server runs an *independent* set up phase of a BES scheme Σ with the users.

Broadcast Phase.

1. The movie company splits the movie into n parts say, $stream_1, \dots, stream_n$, and encrypts every part by computing $E_k(stream_i)$ for $i = 1, \dots, n$. Then, it sends to server S_i , the encrypted stream $E_k(stream_i)$, for $i = 1, \dots, n$.
2. Every server S_i , using Σ , broadcasts the value y_i .
3. From any subsets of t out of the n values y_i 's, each privileged user recovers the decryption key k .
4. Every server S_i broadcasts $E_k(stream_i)$.
5. Every user belonging to the privileged set collects and re-arranges the n parts, decrypts them, and watches the movie.

The above protocol is trivial. The company encrypts the movie and shares, according to a threshold scheme, the secret key among the servers. Then, each server broadcasts to the privileged users his share. Finally, any t -subset of shares enables the recovering of the key. With such a system any $t - 1$ servers have no information about the movie (*security*) and there is enough redundancy in order to recover the decryption key, because t out of the n broadcasts are sufficient to recover the key k .

The question is: can we do better? Can we design a scheme *from scratch* where each server broadcasts *less information* compared to the trivial protocol (in which basically each server runs a broadcast encryption scheme)? Can we design a distributed broadcast encryption scheme where, during the set-up phase, the movie company generates information for a sort of “global broadcast encryption scheme” that it distributes to the servers and which is used by the servers to send pre-defined keys to the users?

Remark. Notice that, if the company, instead of just splitting the movie into n parts, divides the movies in packets in such a way that *at least* two different servers broadcast the same packet, or it uses a threshold secret sharing scheme to share the encrypted movie among the servers, or it uses other techniques like the IDA (Information Dispersal Algorithm) then, even during the broadcast of the movie, if some server crashes, the transmission is still successful (i.e., we enhance the reliability of the scheme). Of course, the technique used in this phase (i.e., simple splitting of the movie, secret sharing, IDA, etc.), implies different performance, storage requirement for the servers, and communication complexity of the transmission.

Previous Work. Broadcast encryption was first considered in [4] and, subsequently, formally defined and studied in [16]. Since then, it has become a major topic in Cryptography, due to the large number of possible applications, and it has evolved in several directions. These directions include multicast communications [44, 9, 10, 33, 34, 13], where the privileged subset of recipients *dynamically* changes by means of join and remove operations; traitor tracing [11, 32, 14, 8, 41, 3, 19, 17, 35, 37, 21, 22], where the emphasis is on catching dishonest users who set up illegal decrypting devices; and revoking schemes [2, 30, 24, 29, 20, 25], which allow efficient and fast revocation of a small group of users. Moreover, several schemes presented in the literature achieve more than one of the above-mentioned aspects, e.g., broadcast plus traitor tracing capabilities [18, 40], or revocation and tracing capabilities [30, 29, 20]. Broadcast encryption requirements have been studied in several papers, e.g., [5, 6, 27]. A survey on unconditionally secure broadcast encryption schemes can be found in [38]. Recently, some papers have considered a setting in which packets can get lost during transmission. In [31], error correction techniques have been employed. In [45], short “hint” messages are appended to the packets. The schemes given in [24], by carefully choosing the values of the parameters, can provide resistance to packet loss as well. In [36, 26], the group key establishment problem over unreliable networks has been addressed, and a key recovery mechanism, quite similar in both papers, has been provided: each packet sent by the broadcaster enables the users to recover the current session key and a share of previous and subsequent ones.

2 An Informal Model

In the following, we are mainly interested in schemes for the distribution of the encryption-decryption keys to the users by means of the servers. More precisely, we consider the following model. Let \mathcal{CP} be a content provider (e.g., a movie company), let $\mathcal{U} = \{U_1, \dots, U_m\}$ be a set of users, and let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a set of servers. The content provider is connected to the servers by means of *secure point-to-point channels*, and each server is connected to the users by means of *secure point-to-point channels*. Servers and users have access to a common *broadcast channel*.

An (m, n, t, ω) -DBES scheme enables a content provider \mathcal{CP} to transmit a message to a privileged subset of users via the set of n servers in such a way

that no coalition of $t - 1$ servers and no coalition of at most ω unauthorized users can compute any information about the message. The protocol is divided into four steps:

- *Set up Phase - Step 1: Content Provider-Servers.* The content provider \mathcal{CP} sends information to the n servers along secure point-to-point channels.
- *Set up Phase - Step 2: Servers-Users.* The n servers send private information to every user of the system along secure point-to-point channels.
- *Broadcast Phase - Step 1.* The \mathcal{CP} encrypts the message k he wishes to transmit, and sends encrypted information to the servers along secure point-to-point channels.
- *Broadcast Phase - Step 2.* The n servers broadcast encrypted information along the broadcast channel in such a way that only a privileged subset of users $P \subseteq \mathcal{U}$ can recover the original message k . Notice that $P \subseteq \mathcal{U}$ could be arbitrary, and not known before the broadcast phase.

The properties that an (m, n, t, ω) -DBES must satisfy are the following:

1. *Fault-tolerant Reconstruction.* From the broadcast of any t out of the n servers, each of the users belonging to the privileged subset P can reconstruct the key k .
2. *Security with respect to receivers.* No coalition of at most ω unprivileged users can determine any information about k .
3. *Security with respect to servers.* No $t - 1$ servers can determine any information² about k .
4. *Efficiency.* The amount of broadcasted and secret information is as small as possible.

Property 1 implies that we have *fault tolerant* transmissions: privileged users are still able to compute k even if some broadcasts get lost during the transmission, or some servers of the system crashes. As we will show, *fault tolerance* and *distribution* are independent targets. Some of the techniques we propose in the following apply to both BES and DBES Schemes.

3 Blacklisting Problem Constructions

An interesting approach to a special kind of broadcast encryption has been introduced in [24]. The problem considered therein is how to broadcast information in a secure way to *all but a few users* of the system. This special case of broadcast encryption is also referred to as *broadcast exclusion* or *blacklisting problem*.

Let us state the following combinatorial definition:

² This is the strongest possible security condition. Indeed, the reconstruction property implies that *any* subset of t servers, from the private information received by \mathcal{CP} during the set up phase, can compute the secret key k . This is unavoidable if we require that the broadcast of t servers is sufficient for the recipients to recover the secret key k .

Definition 1 ([15]). Let $\mathcal{K} = \{k_1, \dots, k_n\}$ be a ground set, and let d, ω be positive integers. A family $\mathcal{A} = \{A_1, \dots, A_m\}$ of subsets of \mathcal{K} is an (ω, d) -cover-free family if, for all $A, A_1, \dots, A_\omega \in \mathcal{A}$, such that $A \notin \{A_1, \dots, A_\omega\}$, it holds that

$$|A \setminus \cup_{i=1}^\omega A_i| \geq d.$$

The definition guarantees that the union of ω subsets does not completely cover any other subset. More precisely, at least d elements are not covered.

Given a cover-free family, a broadcast exclusion scheme can be easily set up as follows: let \mathcal{K} be the set of keys, and let $m = |\mathcal{A}|$ be the number of users of the system.

1. Each user u_i gets the keys associated with A_i .
2. When the broadcaster wishes to send a message to all but users $u_{i_1}, \dots, u_{i_\omega}$, he encrypts the message m with each key $k \in \mathcal{K} \setminus \cup_{i=1}^\omega A_i$, and broadcasts $E_k(m)$.

Because $d \geq 1$, every privileged user recovers the message, while for $u_{i_1}, \dots, u_{i_\omega}$ there is no way to them to get any information since their keys are not used (i.e., they are excluded) by the broadcaster. The size of the broadcast can be improved if, instead of repeating the encryption of m for each key in $\mathcal{K} \setminus \cup_{i=1}^\omega A_i$, the message m is encoded through an *erasure code* into $|\mathcal{K} \setminus \cup_{i=1}^\omega A_i|$ smaller but redundant pieces, where any d of them enable recovering m . Several constructions were given in [24], and the reader is referred to that paper for details. Actually, it turns out that a more general goal can be achieved by means of *ramp schemes* (see Appendix A). By means of an accurate choice of the parameters of the ramp scheme, it is possible to gain fault tolerant transmission as well. For example, if the message m is shared according to a $(0, e, r)$ -RS, where $e < d \leq r$ and $r = |\mathcal{K} \setminus \cup_{i=1}^\omega A_i|$, then any e shares enable to recover m . In this case, $d - e$ is the fault tolerance factor provided by the scheme.

The construction we have seen before can be adapted to our distributed setting by simply using $|\mathcal{K}|$ servers. More precisely, during the set up phase, server S_i gets key k_i from \mathcal{CP} , and user u_j gets all keys $k_i \in A_j$ from the corresponding servers S_i . The two-step broadcast phase, is carried out as follows: \mathcal{CP} shares the message m according to a $(0, e, r)$ -RS ramp scheme, and sends a different share to each server whose key does not belong to $\cup_{s=1}^\omega A_{j_s}$. Then, every server S_i encrypts with key k_i and broadcasts the received share.

4 A Small Subset of Privileged Users

The same approach applied in [24] for the broadcast exclusion problem can be applied in another setting: namely, when *both the size of the privileged subset of users and the size of the possible forbidden subsets of users are small*. More precisely, we can use the same idea to set up a broadcast encryption scheme where users in P recover the message, but *any subset F of size at most ω has no information about it*. We now need a generalization of the idea of (ω, d) -cover-free family. More precisely, we state

Definition 2 ([43]). Let $\mathcal{K} = \{k_1, \dots, k_n\}$ be a ground set, and let r, ω, d be positive integers. A family $\mathcal{A} = \{A_1, \dots, A_m\}$ of subsets of \mathcal{K} is an (r, ω, d) -cover-free family if, for all subsets A_{i_1}, \dots, A_{i_r} , and $A_{j_1}, \dots, A_{j_\omega} \in \mathcal{A}$, such that $A_{j_\ell} \neq A_{j_k} \forall k, \ell$, it holds that

$$|\cap_{s=1}^r A_{i_s} \setminus \cup_{s=1}^\omega A_{j_s}| \geq d.$$

The definition establishes that the intersection of any r subsets still has at least d cover-free elements, for any choice of ω other subsets. Efficient constructions of (r, ω, d) -cover-free family are given by Stinson and Wei in [43].

A broadcast encryption scheme can be set up as follows: Given an (r, ω, d) -cover-free family on the set of keys \mathcal{K} , let $P = \{u_{i_1}, \dots, u_{i_r}\}$ be the subset of privileged users. Then,

1. Each user u_i gets the keys associated with A_i . Let $Y = \cap_{s=1}^r A_{i_s}$ be the set of common keys held by users in P .
2. When the broadcaster wishes to send a message to users in P , in such a way that any disjoint subset of size ω , say $F = \{u_{j_1}, \dots, u_{j_\omega}\}$, does not recover the message, he shares the message m according to a certain $(y - d, e, y)$ -RS, where $y = |Y|$, and encrypts share m_i with key $k_i \in Y$, and broadcasts $E_{k_i}(m_i)$.

It is not difficult to see that every user in P has all y keys in $Y = \cap_{s=1}^r A_{i_s}$. On the other hand, every F of size ω has at most $y - d$ of these keys, due to the property of the (r, ω, d) -cover-free family. The use of a $(y - d, e, y)$ -RS guarantees that privileged users recover the message, while a forbidden subset gains no information at all about the message. Moreover, the above construction has a fault tolerance factor equal to $y - e$. It can be distributed along the same line as the blacklisting construction. We skip the details.

The above construction can still be improved in terms of the length of the broadcast, by applying the same technique given in subsection 3.1 of [38] in the context of key predistribution schemes. Basically, the idea is that, instead of using *all* the y keys to generate the broadcast, a smaller subset of z keys is derived from the y keys and used with a $(0, e, z)$ -RS to encrypt the shares for the privileged users. The construction is secure because *only* the privileged users can derive the actual keys used in the broadcast. The fault tolerance factor in this case is given by $z - e$. Formally, we state the following:

Definition 3. A function $f : GF(q)^n \rightarrow GF(q)^z$ is said to be balanced if, for each $y \in GF(q)^z$

$$|\{x \in GF(q)^n | f(x) = y\}| = q^{n-z}.$$

In other words, each value $f(x) \in GF(q)^z$ has the same number of pre-images x .

Definition 4. A function $f : GF(q)^n \rightarrow GF(q)^z$ is said to be k -resilient if, for every subset $\{j_1, \dots, j_k\} \subseteq \{1, \dots, n\}$ and every $(a_1, \dots, a_k) \in GF(q)^k$, the function $f(x_1, \dots, x_n)|_{x_{j_1}=a_1, \dots, x_{j_k}=a_k}$ is balanced.

Roughly speaking, if f is k -resilient, the knowledge of any k input values *does not* give any information about the output (i.e., the outputs are still uniformly distributed).

Hence, in the above protocol, assuming that a public $(y - d)$ -resilient function $f : GF(q)^y \rightarrow GF(q)^z$ is available, and that $Y = GF(q)$, the broadcaster can use the keys in Y to derive the actual keys for encrypting the broadcast. Every user in P can compute such keys as well; while, the $(y - d)$ -resilience property ensures that any coalition F does not get any information about them.

5 Distribution of Some BES Constructions

Many BES constructions can be rewritten to fit our model. The first ones we are going to consider are the one-level and multi-level schemes described in [16].

The idea underlying the following schemes is of first constructing a scheme that works for excluding a single user and then, using multi-layered hashing techniques, to break up coalitions of excluded users into single exclusions from sub-groups.

5.1 Basic Construction

The first broadcast encryption scheme given in [16], which will be used with $\omega = 1$ in the multi-layered constructions, enables the broadcaster to send a message to any $P \in \mathcal{P} \subseteq 2^{\mathcal{U}}$ in such a way that no $F \in \mathcal{F} = \{F \in \mathcal{U} : |F| \leq \omega\}$ such that $F \cap P = \emptyset$ gains any information about it. The scheme can be described as follows:

Basic Fiat-Naor BES.

- *Set up Phase.* For every subset $F \in \mathcal{F}$, \mathcal{CP} chooses a random value $s_F \in GF(q)$ and gives s_F to every member of $\mathcal{U} \setminus F$. The key associated with a privileged set P is defined to be

$$\kappa_P = \sum_{F \in \mathcal{F}: F \cap P = \emptyset} s_F.$$

- *Broadcast Phase.* \mathcal{CP} broadcasts the value

$$b_P = \kappa_P + m_P \bmod q.$$

It is not difficult to see that each user in P recovers the message, while any coalition F , disjoint from P , cannot, since the coalition does not have the value s_F .

5.2 One-Level and Multi-level Schemes

In order to describe the multi-level constructions, let us recall the following definition:

Definition 5. An (n, m, ω) -perfect hash family is a set \mathcal{H} of functions

$$f : \{1, \dots, n\} \rightarrow \{1, \dots, m\}$$

such that, for every subset $X \subseteq \{1, \dots, n\}$ of size ω , there exists a function $f \in \mathcal{H}$ whose restriction to X is one-to-one.

An (n, m, ω) -perfect hash family is usually denoted by $\text{PHF}(N, n, m, \omega)$, where $|\mathcal{H}| = N$. Using several one-resilient BES schemes and a $\text{PHF}(N, n, m, \omega)$, it is possible to set up a ω -resilient BES scheme as follows [16]: For $1 \leq i \leq N$ and $1 \leq j \leq m$, let $R(i, j)$ be a $(n, 1)$ -BES scheme, and let $\text{PHF}(N, n, m, \omega)$ be a family of perfect hash functions.

- *Set-up Phase.* \mathcal{CP} sends to every user $i \in \{1, \dots, n\}$ the keys associated with him in the scheme $R(i, f_j(i))$, for $j = 1, \dots, N$.
- *Broadcast Phase.* \mathcal{CP} , in order to send a message m , uses an (N, N) threshold scheme: he chooses $N - 1$ random elements, m_1, \dots, m_{N-1} , and computes

$$m_N = \bigoplus_{i=1}^{N-1} m_i \oplus m$$

where we assume that m, m_1, \dots, m_N are elements of a finite field and \oplus denotes the sum operator.

- Then, he broadcasts, for $j = 1, \dots, N$, the values m_j to the users belonging to $P \subseteq \{1, \dots, n\}$ by means of the schemes $R(j, f_j(u))$, for every $u \in P$.

Every user in P can recover all the m_j 's, and then can compute the message by a simple addition in a finite field. On the other hand, the properties of the hash family guarantee that, for any subset $X = \{i_1, \dots, i_\omega\}$ of users, one of the functions $f_j \in \mathcal{H}$ is one-to-one on X . Hence, the users in X cannot break any of the schemes $R(j, f_j(i_1)), \dots, R(j, f_j(i_\omega))$ since they are one-resilient and can be broken only if at least two dishonest users are associated with the same scheme, i.e., $f_j(i_k) = f_j(i_\ell)$ for $k \neq \ell$. As a consequence, even if some user in P receives m_j , by means of one of the schemes $R(j, f_j(i_1)), \dots, R(j, f_j(i_\omega))$, message m_j cannot be recovered by X . Therefore, m cannot be computed by X .

The above scheme can be easily distributed by associating some of the functions in \mathcal{H} with each server. More precisely, an (m, n, n, ω) -DBES can be set up as follows:

- *Set-up Phase - Step 1:* \mathcal{CP} -Servers. \mathcal{CP} generates a $\text{PHF}(N, n, m, \omega)$ and sends the descriptions of N/r of them to each server. Server S_i , for $i = 1, \dots, r$, gets the functions $f_{(i-1)\frac{N}{r}+1}, \dots, f_{(i-1)\frac{N}{r}+\frac{N}{r}}$.
- *Set-up Phase - Step 2:* Servers-Users. Each server, for $\ell = 1, \dots, N/r$ and for $j = 1, \dots, m$, constructs $\frac{N}{r} \times m$ one-resilient schemes $R(\ell, j)$. Then, he sends to every user u , the values associated with the scheme $R(\ell, f_\ell(u))$, for every $\ell = 1, \dots, N/r$.
- *Broadcast Phase - Step 1:* \mathcal{CP} chooses $N - 1$ random elements, m_1, \dots, m_{N-1} , and computes

$$m_N = \bigoplus_{i=1}^{N-1} m_i \oplus m$$

Then he sends, for $i = 1, \dots, r$, the values $m_{(i-1)\frac{N}{r}+1}, \dots, m_{(i-1)\frac{N}{r}+\frac{N}{r}}$ to server S_i .

- *Broadcast Phase - Step 2:* Each server broadcasts, for $\ell = 1, \dots, N/r$, the values m_ℓ he holds to the users belonging to $P \subseteq \{1, \dots, n\}$ by means of the schemes $R(\ell, f_\ell(u))$, for every $u \in P$.

Along the same line, even the multi-level scheme can be distributed among a set of r servers. Each server still receives the description of some functions belonging to the perfect hash family and generates sets of smaller one-level schemes as proposed in [16].

Both the above constructions, the centralized one given in [16] and the distributed one here described, can be enhanced by adding fault tolerance. To this aim, we need the concept of a *generalized* perfect hash function family to set up a fault tolerant (m, n, t, ω) -DBES.

Definition 6 ([34]). An α - $\text{PHF}(N, n, m, \omega)$ perfect hash family is a $\text{PHF}(N, n, m, \omega)$ of functions \mathcal{H} such that, for every subset $X \subseteq \{1, \dots, n\}$ of size ω , there exists at least α functions $f \in \mathcal{H}$ whose restriction to X is one-to-one.

Notice that, if $\alpha = N - t + 1$, in any α -generalized perfect hash family, for every ω -subset $X \subseteq \{1, \dots, n\}$ and any t -subset $Y \subset \mathcal{H}$, there exist at least one function in Y whose restriction to X is one-to-one. This property is exactly what we need in order to set up a fault tolerant BES (resp. an (m, n, t, ω) -DBES). Indeed, if an α - PHF is used, during the broadcast phase - step 1, instead of splitting m by means of an (N, N) -SSS, m is “split” according to a (t, N) -SSS.

It is easy to see that an α - $\text{PHF}(\alpha N, n, m, \omega)$ can be constructed by taking α copies of a $\text{PHF}(N, n, m, \omega)$. However, more efficient constructions are possible. For example, by applying the probabilistic method [1], we can prove the following existential result about α -generalized perfect hash functions.

Theorem 1. Let $\alpha \leq \frac{Nq}{2} + 1$. Then, there exists an α -PHF(N, n, m, ω) if

$$N > 8 \frac{\omega \log n - \log(\omega!)}{q}.$$

Proof. Assume that the α -PHF(N, n, m, ω) is represented by means of a matrix A , where each row consists of a function of the family. Let $C \subseteq \{1, \dots, n\}$ be a subset of size ω of the n columns. It is not difficult to check that, assuming that A is filled in *uniformly at random*, the probability that one of the rows of $A[C]$ has *all different values* is given by

$$q = \frac{m(m-1)(m-2) \cdots (m-\omega-1)}{m^\omega}.$$

Therefore, the probability that i rows of $A[C]$ have different values is equal to

$$\binom{N}{i} q^i (1-q)^{N-i}.$$

Let the random variable $X(C)$ be defined as follows:

$$X(C) = \begin{cases} 0, & \text{if more than } \alpha \text{ rows of } A[C] \text{ have all different values} \\ 1, & \text{otherwise.} \end{cases}$$

Moreover, let

$$Y_i = \begin{cases} 0, & \text{if the } i\text{-th row of } A[C] \text{ has all different values} \\ 1, & \text{otherwise,} \end{cases}$$

and let $Y = Y_1 + \dots + Y_N$. It is not difficult to see that,

$$E[X(C)] = \sum_{i \leq \alpha-1} \binom{N}{i} q^i (1-q)^{N-i} = P(Y \leq \alpha-1).$$

A bound on the tail of the binomial probability distribution (see [23], p. 106) establishes that, for any $a \geq 0$,

$$P(Y \leq N(q-a)) \leq e^{-\frac{a^2 N}{2q}}.$$

If we want $N(q-a) = \alpha-1$ and we set $\alpha-1 = \frac{qN}{2}$, then $a = \frac{q}{2}$. Therefore, it holds that:

$$E[X(C)] = \sum_{i \leq \alpha-1} \binom{N}{i} q^i (1-q)^{N-i} \leq e^{-\frac{qN}{8}}.$$

Denoting $X = \sum_{C:|C|=\omega} X(C)$, it holds that

$$E(X) = \binom{n}{\omega} E(X(C)).$$

Hence, we get

$$E(X) \leq \binom{n}{\omega} e^{\frac{-Nq}{8}} \leq \frac{n^\omega}{\omega!} e^{\frac{-Nq}{8}}.$$

The above expected value $E(X)$ is less than 1 if

$$\omega \log n - \log \omega! - \frac{Nq}{8} < 0,$$

which is satisfied if

$$N > 8 \frac{[\omega \log n - \log (\omega!)])}{q}.$$

Therefore, the above condition guarantees the existence of an α -PHF(N, n, m, ω).

5.3 The KIO Construction

A general construction for BES schemes has been proposed in [38]. The idea is to use the Basic Fiat-Naor scheme in conjunction with an ideal secret sharing scheme (ISSS, for short). The goal in [38, 39, 42] was to obtain schemes where each user has to *store fewer pre-defined keys* and the broadcast messages are *shorter*, compared to other constructions. Using the KIO construction we obtain in our setting a Γ -DBES, where Γ is a specification of subsets of servers (i.e., access structure) that enable a subset of privileged users to receive the secret message. In other words, each participant i in a privileged set P uses the values received from a subset of servers belonging to Γ to recover the secret message. In this case the security of \mathcal{CP} with respect to the servers depends on the size of the smallest authorized subset in Γ .

Let $\mathcal{B} = \{B_1, \dots, B_\beta\}$ be a family of subsets of \mathcal{U} , and let ω be an integer. For each $1 \leq j \leq \beta$, suppose a Basic Fiat-Naor scheme ($\leq |B_j|, \leq \omega$) is constructed with respect to user set B_j . The secret values associated with the j -th scheme will be denoted s_{jC} , where $C \subseteq B_j$ and $|C| \leq \omega$. The value s_{jC} is given to every user in $B_j \setminus C$. Moreover, suppose that $\Gamma \subseteq 2^{\mathcal{B}}$ and there exists a Γ -ISSS defined on \mathcal{B} with values in $GF(q)$. Let $\mathcal{F} \subseteq 2^{\mathcal{U}}$, and suppose that

$$\{B_j : i \in B_j\} \in \Gamma \quad \forall i \in \mathcal{U} \quad \text{and} \quad \{B_j : |F \cap B_j| \geq \omega + 1\} \notin \Gamma \quad \forall F \in \mathcal{F}. \quad (1)$$

Then, we can construct a $(\leq n, \mathcal{F})$ -BES as follows: let $P \subset \mathcal{U}$. \mathcal{CP} can broadcast a message $m_P \in GF(q)$ to P using the following algorithm:

1. For each $B_j \in \mathcal{B}$, \mathcal{CP} computes a share $y_j \in GF(q)$ corresponding to the secret m_P .
2. For each $B_j \in \mathcal{B}$, \mathcal{CP} computes the key k_j corresponding to the set $P \cap B_j$ in the Basic Fiat-Naor scheme implemented on B_j :

$$k_j = \sum_{C \subseteq B_j : C \cap P = \emptyset, |C| \leq \omega} s_{jC}$$

3. For each $B_j \in \mathcal{B}$ \mathcal{CP} computes $b_j = y_j + k_j$.
4. The broadcast is $b_P = (b_j : B_j \in \mathcal{B})$.

The basic idea of the KIO construction can be explained as follows: first, consider a user $i \in P$ and define $A_i = \{j : i \in B_j\}$. User i can compute k_j for every $j \in A_i$. Then, for each $j \in A_i$, i can compute $y_j = b_j - k_j$. Finally, since $A_i \in \Gamma$, i can compute the message m_P from the shares y_j where $j \in A_i$. On the other hand, let $F \in \mathcal{F}$ be such that $F \cap P = \emptyset$. Define

$$A_F = \{j : |F \cap B_j| \geq \omega + 1\}.$$

The coalition F can compute k_j , and hence y_j for every $j \in A_F$. However, they can obtain no information about the shares y_j , where $j \notin A_F$. Since $A_F \notin \Gamma$, F has no information about the value of m_P .

A straightforward application of the KIO construction to the distributed setting can be done as follows: \mathcal{CP} gives the “responsability” of a subset $B_j \in \mathcal{B}$ to every server. Then, every server sets up a $(\leq |B_j|, \leq \omega)$ Basic Fiat-Naor scheme on its subset, and performs the set up phase of such a scheme with the users. Finally, during the broadcast phase, \mathcal{CP} shares the message m_P he wishes to send to the subset P and sends a share to each server; the servers broadcast, according to the same logic of the non-distributed KIO, the values related to users in P .

The KIO construction can be improved by means of a specific set system. Using a suitable design it is possible to achieve *fault tolerance* for both the original KIO construction and the distributed one. To this aim, we introduce the following definition [39]:

Definition 7. An (n, β, r, λ) -design is a pair (X, \mathcal{B}) such that the following properties are satisfied

1. $|X| = n$
2. \mathcal{B} is a set of β subsets of X called blocks
3. every point occurs in exactly r blocks
4. every pair of points occurs in at most λ blocks.

If a further property is satisfied we get the following:

Definition 8. An (n, β, r, λ) -design is called an ω -threshold design provided that, for all $F \subseteq X$ such that $|F| \leq \omega$, we have that

$$|\{B \in \mathcal{B} : |F \cap B| \geq 2\}| \leq r - 1.$$

Finally, it is possible to show that:

Lemma 1 ([39]). An (n, β, r, λ) -design is an ω -threshold design if $r > \lambda(\omega)_2$.

Using an ω -threshold design (X, \mathcal{B}) , and in particular the collection of subsets \mathcal{B} , we can set up a fault tolerant BES (resp. (m, n, t, ω) -DBES) scheme. More precisely, at each subset $B_i \in \mathcal{B}$ is associated a $(\leq |B_i|, \leq 1)$ Basic Fiat-Naor scheme, and the secret k is split among the β subsets by means of a (t, β) threshold secret sharing scheme, where $t = \lambda(\omega)_2 + 1 \leq r$.

Notice that t shares allow the secret to be reconstructed, while any coalition of at most ω participants does not possess enough shares (i.e., condition (1) of the KIO construction is satisfied due to Definition 8 and the choice of t). Moreover, if less than $r - t$ shares broadcasted by the servers are lost, then each privileged participant gets at least t shares that he can decrypt (i.e., $r - t$ is the fault tolerance factor). Finally, it is interesting to point out that, by choosing in an appropriate way the values of the parameters $r \geq t > \lambda(\omega)_2$, it is possible to achieve a *tradeoff* between security and fault tolerance.

The above instance of the basic KIO construction was given in [39], and an explicit construction of a suitable ω -threshold design by means of universal hash families can be found in subsections 5.3 and 5.4 of that paper. Notice that ω -threshold designs were therein used in order to obtain BES schemes with better information rates compared to the KIO schemes based on other designs (i.e., Balanced Incomplete Block Design).

5.4 Key Distribution Patterns

Another DBES construction can be set up using Key Distribution Patterns. Key Distribution Patterns were introduced by Mitchell and Piper [28]. We briefly recall this concept by following the exposition given in [38].

Let $\mathcal{B} = \{B_1, \dots, B_\beta\}$ be a set of subsets of \mathcal{U} . We say that $(\mathcal{U}, \mathcal{B})$ is a $(\mathcal{P}, \mathcal{F})$ -Key Distribution Pattern ($(\mathcal{P}, \mathcal{F})$ -KDP, for short) if

$$\{B_j : P \subseteq B_j \text{ and } F \cap B_j = \emptyset\} \neq \emptyset$$

for all $P \in \mathcal{P}$ and $F \in \mathcal{F}$ such that $P \cap F = \emptyset$.

Along the same lines of the Basic Fiat-Naor Scheme, a BES scheme for $(\mathcal{P}, \mathcal{F})$ can be set up as follows:

KDP-based BES.

- *Set up Phase.* For every subset $B_j \in \mathcal{B}$, \mathcal{CP} chooses a random value $s_j \in GF(q)$ and gives s_j to every user in B_j . The key associated with a privileged set P is defined to be

$$\kappa_P = \sum_{j:P \in B_j} s_j.$$

- *Broadcast Phase.* \mathcal{CP} broadcasts the value

$$b_P = \kappa_P + m_P \bmod q.$$

Notice that each user in P can compute m_P . On the other hand, if F is a coalition of users disjoint from P , then there is at least one block B_j such that $P \subseteq B_j$ and $F \cap B_j = \emptyset$. Therefore, F does not hold s_j and cannot compute κ_P .

The above scheme can be easily distributed to set up a (m, n, t, \mathcal{F}) -DBES, if \mathcal{CP} , during the set up phase, gives, for $j = 1, \dots, \beta$, to server S_j the value s_j , corresponding to the subset B_j , and each server sends this value to all the users in B_j . Then, during the broadcast phase, \mathcal{CP} shares the message m_P among the servers S_j such that $P \subseteq B_j$. Each server, broadcasts $m_P^j + s_j$ where m_P^j is his share for m_P . If t represents the minimum number of supersets $B_j \in \mathcal{B}$ for the subset $P \in \mathcal{P}$, then any $t - 1$ servers do not get any information about the message m_P . On the other hand, the number of servers that enable the recovering of the message is in general $\geq t$ and depends on P . Notice that, if for each $P \in \mathcal{P}$, the number of supersets B_j of P is exactly t , then we get an (m, n, t, \mathcal{F}) -DBES (i.e., $t - 1$ servers do not get any information, but t do/reconstruct the message). Fault tolerance depends on the choices of the parameters of the secret sharing scheme (or the ramp scheme) used by \mathcal{CP} to share the message m . Resilient functions also can be used here in order to reduce the length of the broadcast message.

6 Conclusions

In this paper we have considered two aspects of secure broadcast transmission: *reliability* and *trust* in the broadcaster. We have extended some existing broadcast encryption schemes, in order to gain fault tolerance and to remove the need for trust in the broadcaster. These schemes permit a tradeoff between fault tolerance and trust, and can be applied in a variety of broadcast scenarios.

Acknowledgements

D. R. Stinson's research is supported by NSERC grants IRC # 216431-96 and RGPIN # 203114-02.

References

- [1] N. Alon and J. Spencer, *The Probabilistic Method*, John Wiley, (2nd Edition), 2000. [272](#)
- [2] J. Anzai, N. Matsuzaki, and T. Matsumoto, *A Quick Group Key Distribution Scheme with Entity Revocation*, Advances in Cryptology - Asiacrypt '99, Lecture Notes in Computer Science, Vol. 1716, pp. 333-347. [266](#)
- [3] O. Berkman, M. Parnas, and J. Sgall, *Efficient Dynamic Traitor Tracing*, Proc. of the 11-th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2000), pp. 586–595, 2000. [266](#)
- [4] S. Berkovits, *How to Broadcast a Secret*, Advances in Cryptology - Eurocrypt '91, Lecture Notes in Computer Science, vol. 547, pp. 536–541, 1991. [266](#)
- [5] C. Blundo and A. Cresti, *Space Requirements for Broadcast Encryption*, Advances in Cryptology - Eurocrypt '94, Lecture Notes in Computer Science, vol. 950, pp. 287–298, 1995. [266](#)
- [6] C. Blundo, Luiz A. Frota Mattos, and D. R. Stinson, *Generalized Beimel-Chor Schemes for Broadcast Encryption and Interactive Key Distribution*, Theoretical Computer Science, vol. 200, pp. 313–334, 1998. [266](#)
- [7] G. R. Blakley and C. Meadows, *Security of Ramp Schemes*, Advances in Cryptology - Crypto '84, Lecture Notes in Computer Science, vol. 196, pp. 242-268, 1984. [280](#)
- [8] D. Boneh and M. Franklin, *An Efficient Public Key Traitor Scheme*, Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science, vol. 1666, pp. 338–353, 1999. [266](#)
- [9] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, *Issue in Multicast Security: A Taxonomy and Efficient Constructions*, Infocom '99, pp. 708–716, 1999. [266](#)
- [10] R. Canetti, T. Malkin, and K. Nissim, *Efficient Communication-Storage Tradeoffs for Multicast Encryption*, Advances in Cryptology - Eurocrypt '99, Lecture Notes in Computer Science, vol. 1592, pp. 459–474, 1999. [266](#)
- [11] B. Chor, A. Fiat, M. Naor and B. Pinkas, *Traitor Tracing*, IEEE Transactions on Information Theory, vol. 46, No. 3, pp. 893–910, May 2000. [266](#)
- [12] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, John Wiley & Sons, 1991. [280](#)
- [13] G. Di Crescenzo and O. Kornievskaja, *Efficient Multicast Encryption Schemes*, Security in Communication Network (SCN02), Lecture Notes in Computer Science, 2002. [266](#)
- [14] C. Dwork, J. Lotspiech, and M. Naor, *Digital Signets: Self-Enforcing Protection of Digital Information*, Proceedings of the 28-th Symposium on the Theory of Computation, pp. 489–498, 1996. [266](#)
- [15] P. Erdos, P. Frankl, and Z. Furedi, *Families of finite subsets in which no set is covered by the union of r others*, Israel Journal of Mathematics, N. 51, pp. 75–89, 1985. [268](#)
- [16] A. Fiat and M. Naor, *Broadcast Encryption*, Proceedings of Crypto '93, Lecture Notes in Computer Science, vol. 773, pp. 480-491, 1994. [266](#), [270](#), [271](#), [272](#)
- [17] A. Fiat and T. Tessa, *Dynamic Traitor Tracing*, Journal of Cryptology, Vol. 14, pp. 211–223, 2001. [266](#)
- [18] E. Gafni, J. Staddon, and Y. L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*, Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science, vol. 1666, p. 372–387, 1999. [266](#)

- [19] J. Garay, J. Staddon, and A. Wool, *Long-Lived Broadcast Encryption*, Advances in Cryptology - Crypto 2000, Lecture Notes in Computer Science, vol. 1880, pp. 333–352, 2000. [266](#)
- [20] D. Halevy and A. Shamir, *The LSD Broadcast Encryption Scheme*, Advances in Cryptology - Crypto '02, Lecture Notes in Computer Science, vol. 2442, pp. 47-60, 2002. [266](#)
- [21] A. Kiayias and M. Yung, *Traitor Tracing with Constant Transmission Rate*, Advances in Cryptology - Eurocrypt '02, Lecture Notes in Computer Science, vol. 2332, pp. 450-465, 2002. [266](#)
- [22] A. Kiayias and M. Yung, *Self Protecting Pirates and Black-Box Traitor Tracing*, Advances in Cryptology - Crypto '01, Lecture Notes in Computer Science, vol. 2139, pp. 63-79, 2001. [266](#)
- [23] D. E. Knuth, *The Art of Computer Programming*, Addison Wesley, (3rd Edition), 1997. [273](#)
- [24] R. Kumar, S. Rajagopalan, and A. Sahai, *Coding Constructions for Blacklisting Problems without Computational Assumptions*, Advances in Cryptology - Crypto '99, Lecture Notes in Computer Science, Vol. 1666, pp. 609–623, 1999. [266](#), [267](#), [268](#)
- [25] H. Kurnio, R. Safani-Naini, and H. Wang, *A Group Key Distribution Scheme with Decentralised User Join*, Security in Communication Network (SCN02), Lecture Notes in Computer Science, 2002. [266](#)
- [26] H. Kurnio, R. Safani-Naini, and H. Wang, *A Secure Re-keying Scheme with Key Recovery Property*, ACISP 2002, Lecture Notes in Computer Science, Vol. 2384, pp. 40–55, 2002. [266](#)
- [27] M. Luby and J. Staddon, *Combinatorial Bounds for Broadcast Encryption*, Advances in Cryptology - Eurocrypt '98, Lecture Notes in Computer Science, vol. 1403, pp. 512–526, 1998. [266](#)
- [28] C. J. Mitchell and F. C. Piper, *Key Storage in Secure Networks*, Discrete Applied Mathematics, vol. 21, pp. 215–228, 1988. [276](#)
- [29] D. Naor, M. Naor, and J. Lotspiech, *Revocation and Tracing Schemes for Stateless Receivers* Advances in Cryptology - Crypto '01, Lecture Notes in Computer Science, vol. 2139, pp. 41–62, 2001. [266](#)
- [30] M. Naor and B. Pinkas, *Efficient Trace and Revoke Schemes*, Financial Cryptography 2000, Lecture Notes in Computer Science, vol. 1962, pp. 1–21, 2000. [266](#)
- [31] A. Perrig, D. Song, and J. D. Tygar, *ELK, a new Protocol for Efficient Large-Group Key Distribution*, in IEEE Symposium on Security and Privacy (2000). [266](#)
- [32] B. Pfitzmann, *Trials of Traced Traitors*, Information Hiding, Lecture Notes in Computer Science, vol. 1174, pp. 49–64, 1996. [266](#)
- [33] R. Poovendran and J. S. Baras, *An Information Theoretic Analysis of Rooted-Tree Based Secure Multicast Key Distribution Schemes*, Advances in Cryptology, Crypto '99, vol. 1666, pp. 624–638, 1999. [266](#)
- [34] R. Safavi-Naini and H. Wang, *New Constructions for Multicast Re-Keying Schemes Using Perfect Hash Families*, 7th ACM Conference on Computer and Communication Security, ACM Press, pp. 228–234, 2000. [266](#), [272](#)
- [35] R. Safavi-Naini and Y. Wang, *Sequential Traitor Tracing*, Lecture Notes in Computer Science, vol. 1880, p. 316–332, 2000. [266](#)
- [36] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin and D. Dean, *Self-Healing Key Distribution with Revocation*, IEEE Symposium on Security and Privacy, May 12-15, 2002, Berkeley, California. [266](#)

- [37] J. N. Staddon, D. R. Stinson and R. Wei, *Combinatorial properties of frameproof and traceability codes*, IEEE Transactions on Information Theory vol. 47, pp. 1042–1049, 2001. [266](#)
- [38] D. R. Stinson, *On Some Methods for Unconditionally Secure Key Distribution and Broadcast Encryption*, Designs, Codes and Cryptography, vol. 12, pp. 215–243, 1997. [266](#), [269](#), [274](#), [276](#)
- [39] D. R. Stinson and T. van Trung, *Some New Results on Key Distribution Patterns and Broadcast Encryption*, Designs, Codes and Cryptography, vol. 15, pp. 261–279, 1998. [274](#), [275](#), [276](#)
- [40] D. R. Stinson and R. Wei, *Key preassigned traceability schemes for broadcast encryption*, Proceedings of SAC'98, Lecture Notes in Computer Science, vol. 1556, pp. 144–156, 1999. [266](#)
- [41] D. R. Stinson and R. Wei, *Combinatorial properties and constructions of traceability schemes and frameproof codes*, SIAM Journal on Discrete Mathematics, vol. 11, pp. 41–53, 1998. [266](#)
- [42] D. R. Stinson and R. Wei, *An Application of Ramp Schemes to Broadcast Encryption*, Information Processing Letters, Vol. 69, pp. 131–135, 1999. [274](#)
- [43] D. R. Stinson and R. Wei, *Generalized Cover-Free Families*, preprint. [269](#)
- [44] D. M. Wallner, E. J. Harder, and R. C. Agee, *Key Management for Multicast: Issues and Architectures*, Internet Draft (draft-wallner-key-arch-01.txt), <ftp://ftp.ietf.org/internet-drafts/draft-wallner-key-arch-01.txt>. [266](#)
- [45] C. Wong, and S. Lam, *Keystone: A Group Key Management Service*, in International Conference on Telecommunications, ICT 2000. [266](#)

A Ramp Secret Sharing Schemes

The idea of a ramp secret sharing scheme has been introduced in [7]. More precisely, a ramp secret sharing scheme $((t_1, t_2, n)\text{-RS}$, for short) is a protocol by means of which a dealer distributes a secret s among a set of n participants \mathcal{P} in such a way that subsets of \mathcal{P} of size greater than or equal to t_2 can reconstruct the value of s ; no subset of \mathcal{P} of size less than or equal to t_1 can determine anything about the value of the secret; and a subset of size $t_1 < |P| < t_2$ can recover *some* information about the secret [7]. Using the entropy function [12], the three properties of a (linear) $(t_1, t_2, n)\text{-RS}$ can be stated as follows. Assuming that P denotes both a subset of participants and the set of shares these participants receive from the dealer to share a secret $s \in S$, and denoting the corresponding random variables in bold, it holds that

- *Any subset of participants of size less than or equal to t_1 has no information on the secret value:* Formally, for each subset $P \in \mathcal{P}$ of size $|P| \leq t_1$, $H(\mathbf{S}|\mathbf{P}) = H(\mathbf{S})$.
- *Any subset of participants of size $t_1 < |P| < t_2$ has some information on the secret value:* Formally, for each subset $P \in \mathcal{P}$ of size $t_1 < |P| < t_2$, $H(\mathbf{S}|\mathbf{P}) = \frac{|P|-t_1}{t_2-t_1}H(\mathbf{S})$.
- *Any subset of participants of size greater than t_2 can compute the whole secret:* Formally, for each subset $P \in \mathcal{P}$ of size $|P| \geq t_2$, $H(\mathbf{S}|\mathbf{P}) = 0$.

Shared Generation of Pseudo-Random Functions with Cumulative Maps

Huaxiong Wang and Josef Pieprzyk

Center for Advanced Computing – Algorithms and Cryptography
Department of Computing, Macquarie University, Australia
`{hwang,josef}@ics.mq.edu.au`

Abstract. In Crypto'95, Micali and Sidney proposed a method for shared generation of a pseudo-random function $f(\cdot)$ among n players in such a way that for all the inputs x , any u players can compute $f(x)$ while t or fewer players fail to do so, where $0 \leq t < u \leq n$. The idea behind the Micali-Sidney scheme is to generate and distribute secret seeds $S = \{s_1, \dots, s_d\}$ of a poly-random collection of functions, among the n players, each player gets a subset of S , in such a way that any u players together hold all the secret seeds in S while any t or fewer players will lack at least one element from S . The pseudo-random function is then computed as

$$f(x) = \bigoplus_{i=1}^d f_{s_i}(x),$$

where $f_{s_i}(\cdot)$'s are poly-random functions. One question raised by Micali and Sidney is how to distribute the secret seeds satisfying the above condition such that the number of seeds, d , is as small as possible.

In this paper, we continue the work of Micali and Sidney. We first provide a general framework for shared generation of pseudo-random function using cumulative maps. We demonstrate that the Micali-Sidney scheme is a special case of this general construction. We then derive an upper and a lower bound for d . Finally we give a simple, yet efficient, approximation greedy algorithm for generating the secret seeds S in which d is close to the optimum by a factor of at most $u \ln 2$.

1 Introduction

The power of sharing information and cryptographic capabilities is crucial in many real-life applications of cryptography. The field of *threshold cryptography* has been extensively studied by many researchers (see, for example, [5, 7, 8, 10, 12, 13, 14, 29]). The main goal of threshold cryptography is to replace a system entity in a classical cryptosystem with a group of entities sharing the same power. Cryptographic primitives and tasks to which threshold cryptosystems have been applied include digital signature, public-key encryption, identification, and block ciphers etc. Most threshold cryptosystems follow the tracks of usage of algebraic structures and their resulting homomorphic properties of the underlying cryptographic primitives. A typical approach for constructing threshold cryptosystems

is a combination of certain secret sharing scheme and cryptographic primitives to be shared, although in general a simple combination of these two cryptographic primitives could result in a completely insecure system.

There are cryptographic primitives whose algorithms rely on a lack of algebraic structure in order to guarantee their security. Examples of such primitives include block ciphers, pseudo-random functions and MACs etc. Sharing non-homomorphic cryptographic primitives has been less studied in the context of threshold cryptography. To our best knowledge, the first paper dealing with non-homomorphic threshold cryptography is [25] in which Micali and Sidney suggested a method for shared generation of a pseudo-random function. Later, Naor, Pinkas and Reingold [27] studied distributed pseudo-random functions and their applications for key distribution. Brickell, Di Crescenzo and Frankel [5] discussed techniques for sharing the computation of a block cipher. The techniques in [25, 27] and [5] are combinatorial in nature, relying on the combinatorial concepts of *resilient collection of set* and *sequence sharing*.

Cumulative maps, also called *cumulative arrays*, were first introduced by Jackson and Martin [18], although constructions for cumulative maps were implicitly used by Ito, Saito and Nishizeki [17] to prove the existence of perfect secret sharing schemes in *any* monotone access structures. Interestingly, cumulative maps have not been subject to much study in the context of secret sharing. One of the reasons is perhaps due to the fact that secret sharing scheme using the cumulative maps generally results in large size of shares. However, it seems that they are particularly suitable as building blocks in non-homomorphic threshold cryptosystems because of their combinatorial natures. This is indeed the case for the work in [25] and [5].

In this paper, we will follow the work [25] and continue the investigation of how to share generation of pseudo-random functions. Note, however, that the techniques we use, can be also effectively applied to share block ciphers, MACs etc. We extend the notion of cumulative maps for access structures to more general ramp access structures. This generalisation allows us to provides a framework of sharing pseudo-random functions with cumulative maps as basic building blocks and show that the Micali-Sidney scheme is a special case under this general construction. We study a special class of cumulative maps, called (t, u, n) cumulative maps, and derive bounds on their relevant parameters. We present a simple, yet efficient approximation greedy algorithm for constructing efficient (t, u, n) cumulative maps, which in turn partially answers an open question proposed in [25].

The paper is organised as follows. In Section 2 we briefly review the concept of access structure and cumulative maps, and introduce the notion of cumulative for ramp access structures. In Section 3 we present a general scheme to share generation of a pseudo-random function using a cumulative maps for ramp access structures. In Section 4, we show that the Micali-Sidney scheme is a special case of our general construction. In Section 5 we derive bounds on minimal (t, u, n) cumulative maps. In Section 6 we give an approximation greedy algorithm to construct efficient (t, u, n) cumulative maps. Finally, we conclude the paper in Section 7.

2 Access Structures and Cumulative Maps

Access Structures Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a group of n players and let $2^{\mathcal{P}}$ denote the family of all subsets of \mathcal{P} . A subset Γ of $2^{\mathcal{P}}$ with the property that if $A \in \Gamma$ and $A \subseteq A'$ then $A' \in \Gamma$ is called *monotone increasing*. An *access structure* Γ is a monotone increasing subset of $2^{\mathcal{P}}$. Elements in Γ are called *authorised subsets* (on Γ) and elements not in Γ are called *unauthorised subset*. The notion of access structures plays an important role in the theory of secret sharing. Informally, (perfect) secret sharing with a given access structure is a method of collective ownership of the secret by distribution of shares in such a way that any subgroup of players from the access structure is able to recover jointly the secret. If, however, a group of players does not belong to the access structure then the players get no information about the secret (in the information-theoretic sense). For example, the first secret sharing scheme invented by Blakley [2] and Shamir [30] was the (u, n) threshold secret sharing scheme in which the access structure is defined by

$$\Gamma = \{A \subseteq \mathcal{P} \mid |A| \geq u\},$$

called the (u, n) *threshold access structure*.

A subset Σ of $2^{\mathcal{P}}$ is called *monotone decreasing* if whenever $A \in \Sigma$ and $B \subseteq A$ then $B \in \Sigma$ ($A, B \subseteq \mathcal{P}$). We will call (Σ, Γ) a *ramp access structure* on \mathcal{P} if Γ is monotone increasing, Σ is monotone decreasing and $\Gamma \cap \Sigma = \emptyset$. Note that an access structure is a ramp access structure with $\Sigma = 2^{\mathcal{P}} \setminus \Gamma$. A secret sharing scheme with a ramp access structure (Γ, Σ) , sometimes called *non-perfect* secret sharing scheme allows the players in $C \in 2^{\mathcal{P}} \setminus (\Gamma \cup \Sigma)$ to recover some partial information about the secret. Secret sharing schemes with access structure and ramp access structure have been extensively studied by many researchers (see, for example [1, 3, 17, 18, 19, 20, 21, 28, 31, 33]).

Cumulative Maps *Cumulative maps* (also called *cumulative arrays*) for access structures were formally introduced in [31] but the idea behind them was implicitly used in the constructions given in [17].

Definition 1. Suppose $\mathcal{P} = \{P_1, \dots, P_n\}$ is the set of n players and (Σ, Γ) is a ramp access structure on \mathcal{P} . Let $X = \{x_1, \dots, x_d\}$ and $\tau : \mathcal{P} \rightarrow 2^X$ a mapping from \mathcal{P} to the family of subsets of X . We call (X, τ) a *cumulative map* for (Γ, Σ) , if the following conditions are satisfied:

1. $\bigcup_{P \in A} \tau(P) = X, \quad \forall A \in \Gamma;$
2. $\bigcup_{P \in B} \tau(P) \subset X, \quad \forall B \in \Sigma.$

Moreover, a cumulative map (X, τ) for (Σ, Γ) is called *minimal* if for any cumulative map (X', τ') for (Σ, Γ) , $|X'| \geq |X|$.

The above definition generalises the one for access structure given in [31] in that if the access structure Γ is viewed as a $(2^{\mathcal{P}} \setminus \Gamma, \Gamma)$ ramp access structure then the two definitions are exactly the same.

Cumulative maps were implicitly used in [17] to show the existence of perfect secret sharing scheme for *any* access structure as follows. Let (X, τ) be a cumulative map for an access structure Γ with $X = \{x_1, \dots, x_d\}$. We may, without loss of generality, assume that the secret set, \mathcal{K} , is an Abelian group. To share a secret $k \in \mathcal{K}$, the dealer randomly chooses d elements $k_1, \dots, k_d \in \mathcal{K}$ such that $k = k_1 + k_2 + \dots + k_d$ then distributes the share to each participant according to (X, τ) . That is, the share of participant P_i is $\{k_j \mid \text{if } x_j \in \tau(P_i)\}$. Then, obviously, the shares from players $A \in 2^{\mathcal{P}}$ consist of all the d components k_1, \dots, k_d if and only if $A \in \Gamma$. It is not difficult to see that the above construction also gives rise to a secret share scheme for the ramp access structure (Σ, Γ) if (X, τ) is a cumulative map for (Σ, Γ) instead.

In [18], Jackson and Martin gave an algorithm to construct the *minimal* cumulative maps for any access structures. Let Γ be an access structure on \mathcal{P} . We denote $\Gamma^+ = \{U_1, \dots, U_d\}$ the set of the maximal unauthorised sets with respect to Γ , we define the mapping τ by

$$\forall P \in \mathcal{P}, \tau(P) = \{U_i \mid P \notin U_i, 1 \leq i \leq d\}.$$

Then (Γ^+, τ) is a cumulative map for Γ . For example, in a (u, n) threshold access structure Γ the set of maximal unauthorised sets is $\Gamma^+ = \{A \subseteq \mathcal{P} \mid |A| = u - 1\}$, so $|\Gamma^+| = \binom{n}{u-1}$ and the share of each participant consists of $\binom{n-1}{u-1}$ components k_i 's.

However, it is not known how to construct minimal cumulative maps for the general ramp access structures.

3 Sharing Generations of Pseudo-Random Function

As we have mentioned in the introduction, threshold cryptosystems in which the underlying cryptosystems (to share) lack of certain algebraically homomorphic properties have been less studied in the literature. To our best knowledge, Micali and Sidney [25] was the first attempt to share the non-homomorphic cryptographic primitive, i.e., pseudo-random functions. In this section, we will generalise the Micali-Sidney scheme to general ramp access structures.

Definition 2. *Let f be a pseudo-random function. We say that players \mathcal{P} share generation of f with a ramp access structure (Σ, Γ) , if the following conditions are satisfied.*

- for all inputs x , any set of players $A \subseteq \mathcal{P}$ can compute and reveal $f(x)$ if $A \in \Gamma$;
- for all inputs x such that $f(x)$ has never been revealed, no set of participant $B \subseteq \mathcal{P}$ can feasibly compute $f(x)$ if $B \in \Sigma$.

The Micali-Sidney scheme to share generation of pseudo-random functions relies on the notion of a poly-random collection of functions introduced by Goldreich *et al*

Let $\mathcal{F} = \{F_\ell \mid \ell = 1, 2, \dots\}$, where $F_\ell = \{f_x : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell \mid x \in \{0, 1\}^\ell\}$ is a collection of 2^ℓ functions. We call \mathcal{F} a poly-random collection of functions if the following properties are satisfied:

- There is a polynomial-time algorithm \mathcal{A} which, given inputs $x, y \in \{0, 1\}^\ell$, computes $f_x(y)$.
- If we choose $x \in \{0, 1\}^\ell$ at random, then the function $f_x : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ behaves exactly like a random function mapping $\{0, 1\}^\ell$ to itself as far as a tester limited to polynomial computational time. *i.e.*, it passes all polynomially bounded statistical tests.

As already pointed out in [25], poly-random collections of functions can be constructed from one-way functions (see, [22, 16]). Now we are ready to generalise the Micali-Sidney scheme for the general ramp access structures.

The Scheme. Let \mathcal{F} be a poly-random collections. Let (Σ, Γ) be a ramp access structure on P and let (X, τ) be a cumulative map for (Σ, Γ) . For an integer ℓ , we share generation of a pseudo-random function from $\{0, 1\}^\ell$ to $\{0, 1\}^\ell$ among \mathcal{P} for the ramp access structure (Σ, Γ) as follows.

Seed Generation and Distribution. Denote $X = \{x_1, \dots, x_d\}$.

For each $x_i \in X$, we associate a subset C_i of \mathcal{P} defined by

$$C_i = \{P_j \mid \text{if } x_i \in \tau(P_j)\}.$$

Each subset C_i of players jointly runs a protocol **GEN-SEED**(i) which chooses at random an ℓ -bit secret seed, s_i , and makes it known to every player in C_i , but not to other players (for practical implementations of the protocol **GEN-SEED**(i), we refer readers to Page 190 in [25]).

Function Evaluation. The shared random function is computed as

$$f(y) = \bigoplus_{i=1}^d f_{s_i}(y),$$

where \bigoplus is the exclusive-OR.

Theorem 1. *The function $f(\cdot)$ defined above is a pseudo-random function from $\{0, 1\}^\ell$ to $\{0, 1\}^\ell$ with the following properties:*

- *For any input $y \in \{0, 1\}^\ell$, and subset of players $A \subseteq \mathcal{P}$ can compute $f(y)$ if $A \in \Gamma$;*
- *For all inputs y such that $f(y)$ has never been revealed, any subset of players $B \subseteq \mathcal{P}$ cannot feasibly compute $f(y)$ if $B \in \Sigma$.*

In other words, the function $f(\cdot)$ is shared among \mathcal{P} with the ramp access structure (Σ, Γ) .

The proof of Theorem 1 follows directly from an identical argument in [25] and the properties of cumulative maps. Indeed, the properties of the cumulative map guarantee that any subset of players $A \in \Gamma$ together is able to find all the secret seeds, so can compute the value of function $f(y)$ as defined; while any subset of players $B \in \Sigma$ will lack at least one secret seed and so is unable to compute $f(y)$. Moreover, the properties of poly-random functions guarantee that even if the information from the players $B \in \Sigma$ is combined with knowledge of previously evaluated values of $f(\cdot)$, they have no useful information about the new value of $f(\cdot)$.

4 (t, u, n) Cumulative Maps

Let $0 \leq t < u \leq n$. We consider ramp access structures in which $\Sigma = \{B \mid |B| \leq t\}$ and $\Gamma = \{A \mid |A| \geq u\}$. In this case, we call (Σ, Γ) a (t, u, n) ramp access structure with (t, u, n) cumulative maps.

Since the value $|X|$ corresponds to the numbers of elements (secret seeds) in the shared generation pseudo-random functions scheme, we wish the value $|X|$ to be as small as possible. We denote the value $|X|$ of a minimal (t, u, n) cumulative map by $d(t, u, n)$. It is obvious that if $t = u - 1$, then a (t, u, n) ramp access structure is a (u, n) threshold access structure. Thus, from [18, 27] we know that $d(u - 1, u, n) = \binom{n}{u-1}$ and there exists a construction that meets the bound. When $t < u - 1$, finding the minimal cumulative map seems to be a hard problem.

Lemma 1. *Let $\mathcal{P} = \{P_1, \dots, P_n\}$. Then (X, τ) is a (t, u, n) cumulative map on \mathcal{P} if and only if (X, τ) is a cumulative map for an access structure Δ such that*

1. *for any $B \subseteq \mathcal{P}$ with $|B| = t$, there exists a $C \notin \Delta$ such that $B \subseteq C$;*
2. *for any $A \subseteq \mathcal{P}$ with $|A| = u$, $A \in \Delta$.*

Proof. Sufficiency. If there is a cumulative map (X, τ) for an access structure Δ satisfying the two conditions in the lemma, it is straightforward to verify that (X, τ) is also a (t, u, n) cumulative map.

Necessity. Let (X, τ) be a (t, u, n) cumulative map. We construct Δ as follows.

$$\Delta = \{A \subseteq \mathcal{P} \mid \bigcup_{P \in A} f(P) = X\}.$$

Then it is straightforward to verify that (X, τ) is a cumulative map for Δ and Δ satisfies the two conditions in the lemma.

At the first glance, the above lemma seems to tell us that cumulative maps for ramp access structures are not interesting by themselves, since, according to Lemma 1, cumulative maps for ramp access structures are *equivalent* to cumulative maps for certain access structures which are already known from [18], i.e., there exists an efficient algorithm to construct minimal cumulative map for any access structure. However, there is no unique access structure that corresponds

to a (t, u, n) ramp access structure under this *equivalence*. More precisely, there are many access structures $\Delta_1, \dots, \Delta_m$ together with their corresponding (minimal) cumulative maps $(X_1, \tau_1), \dots, (X_m, \tau_m)$, all of them *equivalent* to a common (t, u, n) ramp access structure in the sense of Lemma 1. For example, any (k, n) cumulative map, for $k = t + 1, \dots, u$, is also (t, u, n) cumulative map. The challenge is to find such a cumulative map (X_i, τ_i) in which $|X_i|$ is minimal or close to minimal.

The Micali-Sidney scheme is based on a combinatorial object called *resilient collection* introduced in [25]. We will show that each resilient collection gives rise to a cumulative map in a natural way, and consequently our construction is a generalisation of the Micali-Sidney scheme [25].

Definition 3 ([25]). *A (t, u, n) resilient collection of sets is a collection $\mathcal{S} = \{S_1, \dots, S_d\}$ of subset of $\mathcal{P} = \{P_1, \dots, P_n\}$ such that:*

- $|S_i| = n - u + 1$ for $i = 1, \dots, n$.
- If $\mathcal{S} \subseteq \mathcal{P}$ and $|\mathcal{S}| = t$, then there is an S_i such that $S_i \cap S = \emptyset$.

Theorem 2. *If there exists a (t, u, n) -resilient collection of sets $\mathcal{S} = \{S_1, \dots, S_d\}$, then there exists a (t, u, n) cumulative map (X, τ) on \mathcal{P} such that $|X| = d$.*

Proof. Suppose $\mathcal{S} = \{S_1, \dots, S_d\}$ is a (t, u, n) -resilient collection. Set $X = \mathcal{S} = \{S_1, \dots, S_d\}$ and let τ be a mapping from \mathcal{P} to 2^X defined by

$$\tau(P) = \{S_i \mid \text{if } P \in S_i\}.$$

We show that (X, τ) is a (t, u, n) cumulative map. For any u -subset of \mathcal{P} , A , we have $\cup_{P \in A} \tau(P) = X$. Indeed, otherwise if there exists an S_i such that $S_i \notin \tau(P)$ for any $P \in A$, then $P \notin S_i$ for all $P \in A$ and so $A \cap S_i = \emptyset$. But $|A| = u$, it yields that $|S_i| \leq n - u$ which contradicts with the assumption that $|S_i| = n - u + 1$ for $i = 1, \dots, n$. On the other hand, for any t -subset B of \mathcal{P} , there exists an S_i such that $S_i \cap B = \emptyset$ as $\mathcal{S} = \{S_1, \dots, S_d\}$ is a (t, u, n) -resilient collection. It follows that for any $P \in B$, $P \notin S_i$ and so $S_i \notin \tau(P)$. Therefore $S_i \notin \cup_{P \in B} \tau(P)$, which means that $\cup_{P \in B} \tau(P) \neq X$. The claim follows.

Note that if the (t, u, n) cumulative map on \mathcal{P} is taken from a (t, u, n) resilient collection, then our shared generation of pseudo-random function scheme is exactly the same as the original Micali-Sidney scheme. However, the converse of Theorem 2 is not true in general. Cumulative maps for ramp access structure are of independent interest.

It is also interesting to note that Desmedt and Kurosawa [9] introduces verifiers set systems in their designs for the MIX network. Their verifier set systems are (t, u, n) resilient collection in which $u = n - t$. For some constructions of (t, u, n) resilient collections we refer readers to [25, 27], and [9].

5 Bounds on (t, u, n) Cumulative Maps

In this section we are interested in the bound $d(t, u, n)$ for the minimal (t, u, n) cumulative map, we will derive a lower and an upper bound on $d(t, u, n)$.

We start with two lemmas. Let (X, τ) be a (t, u, n) cumulative map on \mathcal{P} . Assume $X = \{x_1, \dots, x_d\}$. To each $x_i \in X$, we associate a subset $C_i \subseteq \mathcal{P}$ defined by

$$C_i = \{P \in \mathcal{P} \mid x_i \in \tau(P)\}.$$

Lemma 2. *For any $A \subset \mathcal{P}$ and any $x_i \in X$, $x_i \in \cup_{P \in A} \tau(P)$ if and only if $A \cap C_i \neq \emptyset$.*

Proof. Necessity. If $x_i \in \cup_{P \in A} \tau(P)$ then there exist at least a $P \in A$ such that $x_i \in \tau(P)$. It follows $P \in C_i$ and so $P \in A \cap C_i \neq \emptyset$.

Sufficiency. If $A \cap C_i \neq \emptyset$, then there exist a $P \in A$ and $P \in C_i$. It follows that $x_i \in \tau(P) \subseteq \cup_{P \in A} \tau(P)$.

Lemma 3. *For $1 \leq i \leq d$, $|C_i| \geq n - u + 1$.*

Proof. Since (X, τ) is a (t, u, n) , we know that any u -subset $A \subseteq \mathcal{P}$ and any $x_i \in X$, $x_i \in \cup_{P \in A} \tau(P)$. From Lemma 2, it follows that $A \cap C_i \neq \emptyset$ for all u -subset $A \subseteq \mathcal{P}$. Therefore $|C_i| \geq n - u + 1$.

Theorem 3. *For any (t, u, n) cumulative map (X, τ) , $|X| \geq \frac{\binom{n}{t}}{\binom{u-1}{t}}$. In other words, we have $d(t, u, n) \geq \binom{n}{t}/\binom{u-1}{t}$.*

Proof. For each C_i , there are $\binom{n-|C_i|}{t}$ subsets of \mathcal{P} of size which are disjoint from C_i . Since there are $\binom{n}{t}$ subsets of \mathcal{P} of size t , from Lemma 2 each of which must be disjoint from some C_i . It follows

$$\binom{n-|C_1|}{t} + \binom{n-|C_2|}{t} + \cdots + \binom{n-|C_d|}{t} \geq \binom{n}{t}.$$

By Lemma 3, we know that for each $1 \leq i \leq d$, $n - |C_i| \leq n - (n - u + 1) = u - 1$. We obtain that

$$d \binom{u-1}{t} \geq \sum_{i=1}^d \binom{n-|C_i|}{t} \geq \binom{n}{t}.$$

Therefore,

$$d \geq \frac{\binom{n}{t}}{\binom{u-1}{t}},$$

which completes the proof of the theorem.

We remark that the lower bound of Theorem 3 for (t, u, n) cumulative map generalises the bound for (t, u, n) resilient collection given in [25]. Next, we give an upper bound on $d(t, u, n)$ which is essentially derived from the bound on (t, u, n) -resilient collection in [25].

Theorem 4. *There exists a (t, u, n) cumulative map (X, τ) , provided*

$$|X| \geq \left\lceil \frac{\binom{n}{t}}{\binom{u-1}{t}} \ln \binom{n}{t} \right\rceil.$$

$$\text{In other words, we have } d(t, u, n) \leq \left\lceil \frac{\binom{n}{t}}{\binom{u-1}{t}} \ln \binom{n}{t} \right\rceil.$$

Proof. It has been shown (Theorem 2 in [25]) that there exists a (t, u, n) -resilient collection \mathcal{S} of d sets, $\mathcal{S} = \{S_1, \dots, S_d\}$ if

$$d \geq \left\lceil \frac{\binom{n}{t}}{\binom{u-1}{t}} \ln \binom{n}{t} \right\rceil.$$

By Theorem 2, our upper bound on $d(t, u, n)$ for (t, u, n) cumulative map follows directly.

Since $\binom{n}{t} \leq 2^n$, the lower bound and upper bound given above differ from each other by at most a factor of $(n \ln 2)$. Thus, these bounds on $d(t, u, n)$ might not be too far away from the optimal value. However, the upper bound for the resilient collection (and so our upper bound on cumulative map) is obtained through a probabilistic construction, so it only provides the existence result. It is desirable to have explicit constructions with the size of X as small as possible.

6 An Approximation Algorithm

The bound given in Theorem 4 is only an existence result, the explicit constructions for (t, u, n) cumulative maps with small size of d are thus highly desired, they are believed to be a difficult problem. We quote from [25]:

...it would be nice if, given legal values for n, t , and u , we could easily come up with an (t, u, n) -resilient collection of as few sets as possible. Unfortunately, we don't know how to do this, and it seems to be a difficult problem to solve in general.

In this section, we will (partially) answer the above question by giving a simple, yet reasonable approximation greedy algorithm that achieves optimum bound by at most a factor of $(u \ln 2)$, such a construction is pretty close to the probabilistic construction given in [25]. The basic idea behind our algorithm is to show that finding a (t, u, n) cumulative map is equivalent to a set-cover problem and then we design a simple greedy algorithm to solve this special set-cover problem.

The *set cover problem* [6] is defined as follows. Let (V, \mathcal{B}) where $V = \{a_1, \dots, a_v\}$ is a finite set and $\mathcal{B} = \{B_1, \dots, B_n\}$ is a family of subsets of V , such that every element of V belongs to at least one set of \mathcal{B} . Consider a subset $\mathcal{C} \subseteq \mathcal{B}$. We say that \mathcal{C} covers V if every element of V is in some set of \mathcal{C} , that is

$$V = \bigcup_{B_i \in \mathcal{C}} B_i.$$

The *set cover problem* finds the minimum-sized subset \mathcal{C} of \mathcal{B} that covers X .

Set cover is an important optimisation problem and has been applied to a number of applications. For example, suppose we want to set up security cameras to cover a large art gallery. From each possible camera position, we can see a certain subset of the paintings. Each subset of paintings is a set in the system. We want to put up the fewest cameras to see all the paintings. It is a well-known fact [6, 26] that the set cover problem is **NP**-complete. In the following we will show that finding a (t, u, n) cumulative map is a special case of the set-cover problem.

Consider a (t, u, n) cumulative map (X, τ) . From Lemma 1 we know that (X, τ) is uniquely determined by an access structure Δ with properties (1) and (2) given in Lemma 1. Furthermore, in this case (X, τ) is a minimal cumulative map for Δ . As before, let Δ^+ denote the collection of maximal unauthorised access sets with respect to Δ , then it is known that Δ is uniquely determined by Δ^+ . We denote the above access structure by $\Delta_{(X, \tau)}$ indexed by its corresponding cumulative map and a similar notation applies to $\Delta_{(X, \tau)}^+$ as well. From [18, 27] we know that $|X| = \Delta^+$. Thus, to find a minimal (t, u, n) cumulative map (X, τ) is the same as to find an access structure Δ such that $\Delta_{(X, \tau)}^+$ is minimal.

Lemma 4. *Let \mathcal{L} be a collection of subsets of \mathcal{P} . There exists a (t, u, n) cumulative map (X, τ) such that $\Delta_{(X, \tau)}^+ = \mathcal{L}$ if and only if the following conditions are satisfied:*

1. *For any t -subset $B \subseteq \mathcal{P}$, there exists a $L \in \mathcal{L}$ such that $B \subseteq L$.*
2. *For any $L \in \mathcal{L}$, $|L| \leq u - 1$.*
3. *Subsets in \mathcal{L} are incomparable. That is, for any $L_1, L_2 \in \mathcal{L}$, $L_1 \not\subseteq L_2$.*

Proof. The proof of the lemma is straightforward by verifying the following facts:

- Condition (1) \Leftrightarrow less than t players can not find all the elements of X .
- Condition (2) \Leftrightarrow more than u players can recover all the elements of X .
- Condition (3) \Leftrightarrow \mathcal{L} consists of maximal unauthorised sets.

From Lemma 4, our problem is reduced to finding \mathcal{L} which satisfies the three conditions with the size $|\mathcal{L}|$ as small as possible.

Set

$$\mathcal{E} = \{E \subseteq \mathcal{P} \mid t \leq |E| \leq u - 1\}.$$

Then, it is clear that if \mathcal{L} satisfies the three conditions in Lemma 4, then $\mathcal{L} \subseteq \mathcal{E}$.

Now we construct a set system (V, \mathcal{B}) as follows. Let $V = \{D \subseteq \mathcal{P} \mid |D| = t\}$ and $\mathcal{B} = \{B_E \mid E \in \mathcal{E}\}$, where B_E is defined as $B_E = \{D \subseteq E \mid D \in V\} \subseteq V$.

Lemma 5. *Let $\mathcal{C} = \{B_{E_1}, \dots, B_{E_d}\} \subseteq \mathcal{B}$ and $\mathcal{L} = \{E_1, \dots, E_d\}$. Then there exists a (t, u, n) cumulative map (X, τ) such that $\mathcal{L} = \Delta_{(X, \tau)}^+$ if and only if the following conditions are satisfied:*

1. \mathcal{C} covers V ;
2. *for any $1 \leq i, j \leq d$, $E_i \not\subseteq E_j$.*

Proof. Condition (1) is equivalent to Condition (1) and Condition (2) of Lemma 4; Condition (2) is equivalent to Condition (3) of Lemma 4.

Obviously, not all the covers \mathcal{C} satisfies condition (2) of Lemma 5. However, if \mathcal{L} is a minimal cover, then condition (2) is guaranteed. Indeed, otherwise assume that there are $E_i, E_j \in \mathcal{L}$ such that $E_i \subseteq E_j$ for some i and j in a minimal cover \mathcal{C} . Then $B_{E_i} \subseteq B_{E_j}$ since any D contained in E_i is also a subset of E_j . Thus if we delete B_{E_i} from \mathcal{C} , \mathcal{C} will still covers V which contradicts with the assumption that \mathcal{C} is minimal.

Now we give a simple, but reasonable approximation greedy algorithm to output a cover that satisfies the conditions in Lemma 5. The algorithm is as follows.

```

Input:  $(V, \mathcal{B})$ 
 $W = V;$ 
 $\mathcal{C} = \text{empty};$ 
while (  $W$  is nonempty ) {
    select  $B$  in  $\mathcal{B}$  that covers the most elements of  $W$ ;
    add  $B$  to  $\mathcal{C}$ ;
     $W = W - B;$ 
}
return  $\mathcal{C};$ 

```

The algorithm doesn't produce a *minimal* cover in general. Fortunately, the output \mathcal{C} does satisfy the condition (2) in Lemma 5 as shown in the following result.

Lemma 6. *Let $\mathcal{C} = \{B_{E_1}, \dots, B_{E_d}\}$ be the output of the above algorithm and let $\mathcal{L} = \{E_1, \dots, E_d\}$. Then \mathcal{C} and \mathcal{L} satisfy the condition (1) and (2) in Lemma 5.*

Proof. It is clear that \mathcal{C} satisfies the condition (1). We need to show the condition (2). Without loss of generality, we assume that the while-loop outputs \mathcal{C} in order B_{E_1}, \dots, B_{E_d} . If Condition (2) is not true, then there exists $i \neq j$ such that $E_i \subseteq E_j$, then we have $B_{E_i} \subseteq B_{E_j}$. We need to consider two cases $i > j$ and $i < j$. For case $i > j$, since the elements covered by B_{E_i} are also covered by B_{E_j} , we know that after B_{E_j} has been selected, the remaining elements are at most $V - B_{E_j}$, and so B_{E_i} is selected to cover the elements from $W - B_{E_{i-1}} \subseteq V - B_{E_j}$. There is no element in $V - B_{E_j}$ that can be covered by B_{E_i} , since these elements have been covered by B_{E_j} and taken away from W . We conclude that $B_i = \emptyset$, which is a contradiction. A similar argument applies to case $i < j$. Therefore Condition (2) for \mathcal{L} is satisfied.

Theorem 5. *The above greedy algorithm has the optimum ratio bound of at most $u \ln 2$.*

Proof. Let c denote the size of the optimum set cover and let d denote the size of the output of the greedy algorithm. We will show that $(d - 1)/c \leq \ln \binom{n}{t}$.

Initially, there are $\alpha_0 = \binom{n}{t}$ elements left to be covered. We know that there is a cover of size c (the optimal cover) and therefore by the pigeonhole principle, there must be at least one set that covers at least α_0/c elements. Otherwise, if every set covers less than α_0/c elements, then no collection of c sets could cover all α_0 elements. Since the greedy algorithm selects the largest set, it will select a set that covers at least these many elements. Thus the number of elements that remain to be covered is at most $\alpha_1 = \alpha_0 - \alpha_0/c = \alpha_0(1 - 1/c)$.

Applying the argument again, we know that we can cover these α_1 elements with a cover of size c (the optimal cover), and hence there exists a subset that covers at least α_1/c elements, leaving at most $\alpha_2 = \alpha_1(1 - 1/c) = \alpha_0(1 - 1/c)^2$ elements remaining.

If we apply this argument $d - 1$ times, each time we succeed in covering at least a fraction of $(1 - 1/c)$ of the remaining elements, then the number of elements that remain uncovered after $d - 1$ sets have been chosen by the greedy algorithm is at most $\alpha_{d-1} = \alpha_0(1 - 1/c)^{d-1}$.

Now consider the largest value of d such that after removing all but the last set of the greedy cover, we will have some elements remaining to be covered. Thus, we are interested in the largest value of d such that

$$1 \leq \alpha_0 \left(1 - \frac{1}{c}\right)^{d-1}.$$

We can rewrite this as

$$1 \leq \alpha_0 \left[\left(1 - \frac{1}{c}\right)^c\right]^{(d-1)/c}.$$

Let where e be the base of the natural logarithm. Since $1 + x \leq e^x$ for all x , we substitute $-1/c$ for x we have $(1 - 1/c) \leq e^{-1/c}$, raising both sides to the c th power, we obtain that for all $c > 0$,

$$\left(1 - \frac{1}{c}\right)^c \leq \frac{1}{e}.$$

It follows that

$$1 \leq \alpha_0(1/e)^{(d-1)/c}.$$

Now if we multiply by $e^{(d-1)/c}$ and take natural logs we get that d satisfies:

$$e^{(d-1)/c} \leq \alpha_0 \implies \frac{d-1}{c} \leq \ln \alpha_0.$$

Each B_D can cover at most $\binom{u}{t}$ elements, we have $\alpha_0 \leq \binom{u}{t} \leq 2^u$, the result follows.

7 Conclusions

In this paper, we show how to share a pseudo-random function using a cumulative map. Our techniques can be directly applied to share other non-homomorphic

cryptographic primitives such as block ciphers and MACs. We believe that cumulative maps for the ramp access structure are of independent interest, as we have mentioned, they have been also applied to the design of the MIX networks.

On the other hand, from the bounds in Section 5 we know that in the worst case the size of the minimal (t, u, n) cumulative map, $d(t, u, n)$, can be $2^{O(n)}$, for example, $d(u - 1, u, n) = \binom{n}{u-1}$. This means that it requires the system to generate many secret keys (i.e., secret seeds in this paper). Recently, Martin *et al* [23, 24] introduced *generalised cumulative array* (GCA) to generalise the concept of cumulative maps for sharing block ciphers and MACs that reduces the number of keys from $2^{O(n)}$ to $O(\log n)$. The GCA approach for sharing block ciphers and MACs seems not suitable for sharing pseudo-functions, as it is not *consistent* in the sense that for the given same input x , different authorised subsets will compute the different value of $f(x)$. One of the interesting problems thus is: can we have other combinatorial approaches for sharing generation of pseudo-random functions (or block ciphers, MACs etc) that exceeds our derived bound? In other words, how can we generalise the cumulative maps for threshold cryptography in other meaningful ways?

References

- [1] J. Benaloh and J. Leichter. Generalised secret sharing and monotone functions. *Adv. in Cryptology—CRYPTO '88*, LNCS, 403(1988), 27–35. [283](#)
- [2] G.R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48(1979), 313–317. [283](#)
- [3] G. R. Blakley and C. Meadows. Security of ramp schemes. *Advances in Cryptology – Proceedings of CRYPTO '84, Lecture Notes in Comput. Sci.*, 196(1985), 242–268. [283](#)
- [4] C. Blundo, A. Cresti, A. De Santis and U. Vaccaro. Fully dynamic secret sharing schemes. *Advances in Cryptology – CRYPTO '93, Lecture Notes in Comput. Sci.*, 773(1993), 110–125.
- [5] E. Brickell, G. Di Crescenzo and Y. Frankel. Sharing Block Ciphers. *Information Security and Privacy, Lecture Notes in Computer Science*, 1841(2000) 457–470. [281](#), [282](#)
- [6] T. Cormen, C Leiserson and R. Rivest, *Introduction to Algorithms*, The MIT Press, 1989. [289](#), [290](#)
- [7] A. De Santis, Y. Desmedt, Y. Frankel and M. Yung. How to Share a Function Securely. *Proceedings of ACM Symp. Theory of Computing (STOC) '94* (1994) 522–533. [281](#)
- [8] Y. Desmedt, Y. Frankel. Threshold Cryptosystems. *Advances in Cryptology – CRYPTO '89, Lecture Notes in Computer Science*, 435(1989), 307–315. [281](#)
- [9] Y. Desmedt and K. Kurosawa. How to Break a Practical MIX and Design a New One. *Eurocrypt'00, Lecture Notes in Computer Science*, 2000, 556–272. [287](#)
- [10] P.-A. Fouque, G. Poupard and J. Stern. Sharing Decryption in the Context of Voting or Lotteries. *Financial Cryptography 2000, Lecture Notes in Computer Science*, 1962 (2001) 90–104. [281](#)
- [11] Y. Frankel, P. Gemmell and M. Yung, Witness-based Cryptographic Program Checking and Robust Function Sharing. *Proc. 28th STOC*, 499–508, ACM, 1996.

- [12] Y. Frankel, P. MacKenzie and M. Yung. Robust efficient distributed RSA-key generation. *Proc. 30th STOC*, 663-672, ACM, 1998. [281](#)
- [13] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust Threshold DSS Signatures. *Advances in Cryptology: Eurocrypt '96, Lecture Notes in Computer Science*, 1070 (1996) 354-371. [281](#)
- [14] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust and efficient sharing of RSA functions, *J. of Cryptology*, 13(2) (2000) 273-300. [281](#)
- [15] O. Goldreich, S. Goldwasser and S. Micali. How to construct random functions. *Journal of the Association for Computing Machinery*, 33(4) (1986), 792-804.
- [16] R. Impallizzo, L. Levin and M. Luby. Pseudo-random generation from one-way functions. *Proceedings of the 21th Annual ACM Symposium on Theory of Computing*, 1989, 12-24. [285](#)
- [17] M. Ito, A. Saito and T. Nishizeki. Secret Sharing Scheme Realizing General Access Structure. *J. Cryptology*, 6 (1993) 15-20. [282](#), [283](#), [284](#)
- [18] W.-A. Jackson and K. M. Martin. Cumulative Arrays and Geometric Secret Sharing Schemes, *Advances in Cryptology: Auscrypt '92, Lecture Notes in Computer Science*, 718 (1993) 48-55. [282](#), [283](#), [284](#), [286](#), [290](#)
- [19] W.-A. Jackson and K. M. Martin. Geometric secret sharing schemes and their duals. *Des. Codes Cryptogr.*, 4(1994), 83-95. [283](#)
- [20] W.-A. Jackson and K. M. Martin. A combinatorial interpretation of ramp schemes. *Australasian Journal of Combinatorics*, 14(1996), 51-60. [283](#)
- [21] K. Kurosawa, K. Okada, K. Sakano, W. Ogata and S. Tsujii. Non-perfect secret sharing schemes and matroids. *Advances in Cryptology: Eurocrypt '93, Lecture Notes in Computer Science*, 765 (1993) 126-141. [283](#)
- [22] L. Levin. One-way functions and pseudorandom generators. *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, 1985, 363-365. [285](#)
- [23] K. Martin, R. Safavi-Naini, H. Wang and P. Wild. Distributing the Encryption and Decryption of a Block Cipher. *Preprint*, 2002. [293](#)
- [24] K. Martin, J. Pieprzyk, R. Safavi-Naini, H. Wang and P. Wild. Threshold MACs. ICISC02, the 5th international conference on information security and Cryptology, Lecture Notes in Computer Science, 2002, to appear. [293](#)
- [25] S. Micali and R. Sidney. A Simple Method for Generating and Sharing Pseudo-Random Functions, with Applications to Clipper-like Escrow Systems. *Advances in Cryptology: CRYPTO '95, Lecture Notes in Computer Science*, 963(1995), 185-195. [282](#), [284](#), [285](#), [286](#), [287](#), [288](#), [289](#)
- [26] D. Mount. *Design and Analysis of Computer Algorithms*. Lecture Notes of the University of Maryland, College Park, 1999. [290](#)
- [27] M. Naor, N. Pinks and O. Reingold, Distributited Pseudo-random Functions and KDCs. *Eurocrypt'99*. [282](#), [286](#), [287](#), [290](#)
- [28] W. Ogata and K. Kurosawa, Some basic properties of general nonperfect secret sharing schemes, *Journal of Universal Computer Science*, 4(8), 1998, 690-704. [283](#)
- [29] T. Rabin, A simplified Approach to Threshold and Proactive RSA, In *Crypto '98*, pages 89-104, 1998. Springer-Verlag, LNCS 1109. [281](#)
- [30] A. Shamir. How to share a secret. *Communications of the ACM*, 22:612-613, 1979. [283](#)
- [31] G. J. Simmons, W.-A. Jackson and K. Martin. The Geometry of Shared Secret Schemes, *Bulletin of the ICA*, 1 (1991), 71-88. [283](#)
- [32] V. Shoup, Practical Threshold Signature, *Advances in Cryptology - Eurocrypt'99*, LNCS, **1807**(2000), 207-222.

- [33] D. R. Stinson. An explication of secret sharing schemes. *Des. Codes Cryptogr.*, 2:357–390, 1992. [283](#)

Authenticated Data Structures for Graph and Geometric Searching*

Michael T. Goodrich¹, Roberto Tamassia², Nikos Triandopoulos², and Robert Cohen³

¹ Dept. Computer Science, Univ. of California, Irvine
`goodrich@acm.org`

² Dept. Computer Science, Brown University
`{rt,nikos}@cs.brown.edu`

³ Dept. Computer Science, Univ. of Massachusetts, Boston
`rfc@cs.umb.edu`

Abstract. Authenticated data structures provide cryptographic proofs that their answers are as accurate as the author intended, even if the data structure is maintained by a remote host. We present techniques for authenticating data structures that represent graphs and collections of geometric objects. In our model, a data structure maintained by a trusted source is mirrored at distributed directories that answer queries and provide proof of correctness. Our work has applications to the authentication of network management systems and geographic information systems.

1 Introduction

Verifying information that at first appears authentic is an often neglected task in data structure and algorithm usage. Fortunately, there is a growing literature on correctness checking that aims to rectify this omission. Following early work on program checking (see, e.g., [4, 37, 38]), efficient schemes have been developed for checking the results of various data structures and algorithms (see, e.g., [2, 5, 6, 7, 15, 16, 19, 24, 29, 30]). The original motivation for this work was mainly to defend the user against an inadvertent error made during implementation.

With the advent of Web services and content delivery networks (e.g., Akamai), data structures and algorithms are no longer being used just by a single user on an individual machine. The computer responding to a user’s query could be unknown to both the data structure author and its user. Thus, we want to guard against the possibility that an agent hosting a data structure could deliberately falsify query responses to users.

In this paper we are interested in studying a new dimension in data structure and algorithm checking—how can we design sophisticated data structures and algorithms so that their responses can be verified as accurately as if they were coming from their author, even when the response is coming from an untrusted

* Research supported in part by DARPA Grant F30602-00-2-0509 and NSF Grant CCR-0098068.

host? Digital signatures can be used to verify simple static documents, but are inefficient for dynamic data structures. We therefore need new techniques for authenticating data structures. In particular, we are interested in efficiently verifying paths and connectivity information in transportation and computer networks and complex geometric queries in spatial databases, such as ray shooting queries, point location queries, and range searching queries.

There are various challenges in providing an integrity service in such contexts. First of all, the space of possible answers is much larger than the data size itself. For example, even in a tree there is a quadratic number of distinct paths between pairs of nodes and requiring an authenticator to digitally sign every possible path query is therefore prohibitive. Also, unlike dictionaries, graph and geometric structures do not have a natural linear ordering of the data upon which one can build a hash tree [31, 32].

Our data structure authentication model involves three parties: a trusted *source*, an untrusted *directory*, and a *user*. The *source* holds a structured collection S of objects, where we assume that a repertoire of *query operations* are defined over S . If S is fixed over time, we say that it is *static*. Otherwise, we say that S is *dynamic* and assume that a repertoire of *update operations* are defined that modify S . The *directory* maintains a copy of the collection S together with *structure authentication information*, which consists of time-stamped statements about S signed by the source. The *user* performs queries on S but instead of contacting the source directly, it queries a directory. The directory provides the user with an answer to the query together with *answer authentication information*, which yields a cryptographic proof of the answer when used in conjunction with the structure authentication information. The user then verifies this proof. If S is dynamic, the directory receives together with each update *update authentication information*, which consists of signed time-stamped statements about the update and the current state of S .

The data structures used by the source and the directory to store collection S , together with the protocols and algorithms for queries, updates, and verifications executed by the various parties, form what we call an *authenticated data structure* [21, 28, 33]. In a practical deployment of an authenticated data structure, there would be various instances of geographically distributed directories. Such a distribution scheme reduces latency, allows for load balancing, and reduces the risk of denial-of-service attacks.

Previous work related to authenticated data structures (initially motivated by its applications to the *certificate revocation* problem in public key infrastructure) is mostly concerned with *authenticated dictionaries*, which are authenticated structures for data sets on which membership queries are performed.

The *Merkle's hash tree* scheme introduced in [31, 32] can be used to implement a static authenticated dictionary. The hashes of the elements of a set are stored at the leaves of a balanced binary tree, each internal node stores the hash of its children and the hash of the root is signed. An element of the set is proven to belong in it by reporting the hashes stored at the sibling nodes of the nodes in the path from the root to the leaf that corresponds to the element. In [25]

the same approach is used, but the verification of negative membership answers is simplified, by storing intervals instead of individual elements. A dynamic authenticated dictionary based on hash trees is described in [33]. Other schemes based on variations of hash trees have been proposed in [8, 20].

A data structure for an authenticated dictionary based on skip lists [35] and its software architecture and implementation are presented in [21, 23]. The notion of commutative hashing is also introduced to simplify the verification process. In [1], the notion of *persistent authenticated dictionaries* is introduced, where the user can issue historical queries of the type “was element e in set S at time t ”. Related work appears in [26, 27]. Finally, in [9, 22] the RSA one-way accumulator is used to realize a dynamic authenticated dictionary with constant verification time.

A first step towards the design of more general authenticated data structures (beyond dictionaries) is made in [14] with the authentication of operations *select*, *project* and *join* in a relational database and multidimensional orthogonal range queries. A study of authenticated queries beyond tree structures and skip lists is presented in [28]. It is shown that a class of data structures such that (i) the links of the structure form a directed acyclic graph of bounded degree and with a single source node and (ii) queries correspond to a traversal of a subdigraph of G , can be authenticated by means of a hashing scheme that digests the entire G into a hash value at its source. This technique is applied to the design of static authenticated data structures for pattern matching in tries and for orthogonal range searching in a multidimensional set of points. This paper also contains an initial treatment of authenticating fractional cascading structures, but only for a special and simple case, where catalogs are arranged as unions in a tree. Related work on the authentication of XML documents appears in [13].

In this paper, we present general techniques for building authenticated data structures for a number of query problems on a general graph G , including *connectivity* queries (i.e., whether two nodes of G are in the same connected component), *biconnectivity* queries, *triconnectivity* queries, and *path queries* (i.e., return the path that connects two nodes of G or return the length of this path). We also support efficient update operations that involve inserting vertices and edges in G . Our data structure uses linear space and supports connectivity queries and update operations in $O(\log n)$ time and path queries in $O(\log n + k)$ time, where k is the length of the path reported. The update authentication information has $O(1)$ size. The size of the answer authentication information and the verification time are each $O(\log n)$ for connectivity, biconnectivity and triconnectivity queries and $O(\log n + k)$ for path queries. These results have applications to the authentication of network management systems.

In addition, we address several geometric search problems, showing how to authenticate the full, general version of the powerful *fractional cascading* technique [10]. Namely, we can authenticate any query efficiently answered in a fractional-cascading structure via *iterative search*, where we have a collection of k dictionaries of total size n stored at the nodes of a connected graph and we want to search for an element in each dictionary in a subgraph of this graph.

A number of fundamental two-dimensional geometric searching problems arising in the implementation of geographic information systems can be solved with data structures based on fractional cascading [11]. These problems include: *line intersection*, *ray shooting*, *point location*, *orthogonal range search*, *orthogonal point enclosure* and *orthogonal intersection* queries (see [34]). Our authenticated fractional cascading data structure can be extended to yield efficient authenticated data structures for all the above problems with applications to the authentication of geographic information systems.

The security of our schemes is based on standard cryptographic primitives, such as collision-resistant hashing and digital signatures; hence, it is practical and does not need any new cryptographic assumptions.

In Section 2, we state our cryptographic assumptions and present the general authentication scheme used. In Section 3, we show how queries on sequences can be authenticated and we present the *path hash accumulator*. In Section 4, we present our authenticated data structures for various *path* and *connectivity* queries on graphs. In Section 5, we present the authentication of the *fractional cascading* algorithmic paradigm, which leads to the authentication of various geometric data structures. We leave open the problem of dynamizing our authenticated data structures based on fractional cascading.

Symbol [A] indicates that the proof (or proof sketch) of the preceding statement appears in the Appendix.

2 Cryptographic Preliminaries

The basis of trust in our authentication model is the assumption that the user trusts the source, i.e., the user has the public key of the source and trusts that anything signed under this key is authentic. Moreover, all the desired security results are achieved by means of the use of a *cryptographic hash function*. A cryptographic hash function h typically operates on a variable-length message M producing a fixed-length hash value $h(M)$. We assume some well-defined binary representation for any data element e , so that h can operate on e . Also, we assume that rules have been defined so that h can operate over any finite sequence of elements. A cryptographic hash function h is called *collision-resistant* if (1) it takes as input a string of arbitrary length and outputs a short string; and (2) it is infeasible to find two different strings $x \neq y$ that hash to the same value, i.e., form a *collision* $h(x) = h(y)$.

Let S be a data set owned by the source. In our authentication schemes, a collision-resistant hash function is used to produce a *digest*, i.e., a cryptographic hash value over the data elements of S . The digest is computed in a systematic way that can be expressed by means of directed acyclic graph (DAG) defined over S (a similar technique is presented in [28]). Once a single-sink DAG G is defined, it is associated with S as follows. Each node u of G stores a *label* $L(u)$ such that if u is a source of G , then $L(u) = h(e_1, \dots, e_p)$, where e_1, \dots, e_p are elements of S , else (u is not a source of G) $L(u) = h(e_1, \dots, e_q, L(z_1), \dots, L(z_k))$, where $(z_1, u), \dots, (z_k, u)$ are edges of G and e_1, \dots, e_q are elements of S (here,

both p and q are some constant integers). We view the label $L(t)$ of the sink t of G as the digest of S . We call the above scheme a *hashing scheme* for S using G .

The authentication techniques presented in this paper are based on the following general scheme. The source and the directory store identical copies of the data structure representing S and maintain the same hashing scheme on S . The source periodically signs the digest of S together with a timestamp and sends the signed timestamped digest to the directory. When updates occur on S , they are sent to the directory together with the new signed time-stamped digest. Note that, in this setting, the update authentication information has $O(1)$ size and the structure authentication information consists only of the digest. When the user poses a query, the directory returns to the user (1) the signed timestamped digest of S , (2) the answer to the query and (3) a proof consisting of a small collection of labels from the hashing scheme that allows the recomputation of the digest. The user validates the answer by recomputing the digest, checking that it is equal to the signed one and verifying the signature of the digest; the total time spent for this process is called the *answer verification time*.

3 Path Properties and the Path Hash Accumulator

In this section, we present an authenticated scheme for various types of queries on a sequence S . An abstract notion of a *path* is used to represent S . We use and extend the notation used in [12]. A *path* is an ordered sequence of one or more nodes, where each node is connected to its successor by a directed edge. By $head(p)$ and $tail(p)$ we denote the first and last nodes of path p . If p' and p'' are paths, the *concatenation* $p = p'|p''$ is a path formed by adding a directed edge from $tail(p')$ to $head(p'')$. A *subpath* $\bar{p}(v, u) = \bar{p}$ of path p is the path consisting of the collection of consecutive nodes of p , with $head(\bar{p}) = v$ and $tail(\bar{p}) = u$.

A path stores a data set through *node attributes*, which are values stored at nodes, and *node properties* which are collections of node attributes. A node attribute $N(v)$ of node v can assume arbitrary values and occupies $O(1)$ storage. A node property $\mathcal{N}(v)$ is a sequence $N_1(v), \dots, N_r(v)$ of node attributes, where r is a constant. Similarly, *path attribute* and *path property* are defined to extend the notion of node attribute and node property. With $P(p)$ we denote a path attribute of p ; it occupies $O(1)$ storage and depends on the node attributes of some of the nodes of p . A path property $\mathcal{P}(p)$ of p is a sequence of path attributes $P_1(p), \dots, P_s(p)$, where s is a constant. We require that $\mathcal{P}(p)$ includes path attributes $head(p)$, $tail(p)$. The definition of path attribute and path property are naturally extended when subpaths of paths are considered.

Let $p = p'|p''$ be a path that is the concatenation of paths p' and p'' . A path property \mathcal{P} satisfies the *concatenation criterion* if $\mathcal{P}(p) = \mathcal{F}(\mathcal{P}(p'), \mathcal{P}(p''))$, where \mathcal{F} is a function that can be computed in $O(1)$ time that is called the *concatenation function* of \mathcal{P} .

Given a path p and a query argument q , a *node selection query* Q_N maps p into a node $v = Q_N(p, q)$ of p . A node selection query is always associated with

some path selection function. Given that $p = p'|p''$, a *path selection function* $\sigma(p, q)$ for Q_N determines in $O(1)$ time whether v is in p' or p'' using q and values $\mathcal{P}(p')$ and $\mathcal{P}(p'')$. A *path selection query* extends a node selection query. Given a path p and some query argument q , a path selection query Q_P maps p into a subpath $\bar{p} = Q_P(p, q)$ of p and is characterized by a *path advance function* $\alpha(p, q)$ that, given that $p = p'|p''$, returns in $O(1)$ time the subpath(s) among p' , p'' (possibly none) for which the query argument q holds (values $\mathcal{P}(p')$ and $\mathcal{P}(p'')$ are used by the path advance function).

Let p be a path and \mathcal{P} be a path property that satisfies the concatenation criterion. We are interested in authenticating the following operations:

- **property**(subpath $\bar{p}(v, u)$): report the value of path property \mathcal{P} for subpath $\bar{p}(v, u)$ of p (\bar{p} may be equal to p).
- **property**(node v): report the value of node property \mathcal{N} for node v .
- **locate**(path p , path selection function σ , argument q): find node v of p returned by the node selection query expressed by σ .
- **subpath**(path p , path advance function α , argument q): find the subpath of p returned by the path selection query expressed by α .

We represent a path p with a balanced binary tree $T(p)$ as follows. A leaf of $T(p)$ represents a node of p . An internal node v of $T(p)$ represents the subpath $p(v)$ of p associated with the leaves in the subtree of v . Each leaf stores the corresponding node property and each non leaf node stores the corresponding path property.

Let h be a collision-resistant hash function. The *path hash accumulator* for a path p is the hashing scheme over the node and path properties of p defined as follows. Consider the data set consisting of: (1) for each leaf node v of $T(p)$, the node property $\mathcal{N}(v)$ and (2) for each internal node u of $T(p)$, the path property $\mathcal{P}(p(u))$. Let G be the DAG obtained from $T(p)$ by directing each edge toward the parent node. For a node v of p , let $\text{pred}(v)$ and $\text{succ}(v)$ denote the predecessor and the successor of v in p , respectively. In particular, $\text{pred}(\text{head}(p))$ and $\text{succ}(\text{tail}(p))$ are some special (nil) values. Using G and h , we compute a label L for each node of $T(p)$ as follows: (1) if u is a source vertex of G , i.e., a leaf of $T(p)$, then $L(u) = h(\mathcal{N}(\text{pred}(u)), \mathcal{N}(u), \mathcal{N}(\text{succ}(u)))$; (2) if w is a non source vertex of G and (z_1, w) and (z_2, w) are edges of G , then $L(w) = h(\mathcal{P}(w), L(z_1), L(z_2))$. The digest of the above data set is the label $L(r)$ of the sink r of G (r is the root of $T(p)$). This digest is called the *path hash accumulation* of path p .

Lemma 1. *Let p be a path of length n . There exists an authenticated data structure of size $O(n)$ for p based on the path hash accumulator scheme such that: (i) query operations **property**, **property**, **locate** and **subpath** each take $O(\log n)$ time; (ii) the update authentication information has size $O(1)$; (iii) the answer authentication information size and the answer verification time each are $O(\log n)$. [A]*

4 Authenticated Graph Searching

In this section, we consider authenticated data structures for graph searching problems. We develop an efficient and fully dynamic authenticated data structure that supports path property queries in a forest (collection of trees). We use the path hash accumulator authentication scheme over a collection Π of paths that are maintained through the update operations on paths `split` and `concatenate`. At a high level point of view, Π is organized by means of a rooted tree \mathcal{T} of paths, meaning that each node of \mathcal{T} corresponds to a path $p \in \Pi$. Neighboring paths in \mathcal{T} are generally interconnected and share information. This is achieved by the definition of suitable node attributes and properties.

A tree of paths \mathcal{T} is considered to be directed; the direction of an edge is from a child to a parent. Let μ be a node of \mathcal{T} , let μ_1, \dots, μ_k be its children in \mathcal{T} and let p be the path that corresponds to μ . A node attribute $N(v)$ of a node v of p is extended so that it depends not only on v but possibly also on some path properties of the paths that correspond to nodes μ_1, \dots, μ_k of \mathcal{T} . We say that path p is the *parent path* of paths μ_1, \dots, μ_k and these paths are the *children paths* of p . Clearly, this extension makes the path property $\mathcal{P}(p)$ of path p that correspond to node μ to include information about paths in the subtree of \mathcal{T} having as root node μ . We again consider path properties that satisfy the concatenation criterion. Note that \mathcal{T} introduces a *hierarchy* over paths in Π .

Let F be a forest. F is associated with a data set by storing at each tree node a node attribute. Using the framework presented in Section 3, any path in F is associated with some path property \mathcal{P} . We assume that \mathcal{P} satisfies the concatenation criterion. We study the implementation of the authenticated query operation $\text{property}(u, v)$ —return the path property of the path from u, v , if such a path exists, while update operations `destroyTree`, `newTree`, `link` and `cut` are performed on the trees of F .

Our data structure is based on dynamic trees [36]. A dynamic tree T is a rooted tree whose edges are classified as being either *solid* or *dashed*, with the property that any internal node has at most one child connected by a solid edge. This edge classification partitions the nodes of the tree into *solid paths* connected with each other by dashed edges (see Appendix, Figure 1(a)). We view a solid path as directed toward the root of T .

Every non-leaf node v of T has at most one child u_0 such that a solid edge connects them. Assume that v has more children and consider all these children, say nodes u_1, \dots, u_k , in T (connected with v through dashed edges). The *dashed path* $d(v)$ is a path of length k such that there is a one-to-one correspondence between edges (u_i, v) in T and nodes of $d(v)$.

Let T_1, \dots, T_m be the trees in F . Let $\Pi(T_i)$ be the collection of all solid and dashed paths defined for tree T_i of F as explained above. Using the hierarchical path scheme discussed at the beginning of the section, we can associate $\Pi(T_i)$ with a directed tree $\mathcal{T}(i)$ of paths. This is performed as follows:

- each path p (solid or dashed) in $\Pi(T_i)$ corresponds to a vertex μ_p of $\mathcal{T}(i)$;
- if p is solid, for each node v of p that has only one child u in T_i such that u is node of path p' and $p \neq p'$ (and is connected with it through a dashed edge), the directed edge $(\mu_{p'}, \mu_p)$ is an edge of $\mathcal{T}(i)$;
- if $p' = d(v)$ is dashed with length k , that is, p' corresponds to the dashed edges of a node v in T_i , let p be the solid path that v belongs to, let u_1, \dots, u_k be the corresponding children of v in T , and let p_1, \dots, p_k be the solid paths containing these children. Then, the directed edges $(\mu_{p'}, \mu_p)$ and $(\mu_{p_i}, \mu_{p'})$, $1 \leq i \leq k$, are edges of $\mathcal{T}(i)$.

Finally, given the directed trees $\mathcal{T}(i)$ ($i = 1, \dots, m$), we add a new *root vertex* ω that connects all the roots of trees $\mathcal{T}(i)$, thus, obtaining a new tree \mathcal{F} . We consider one last *root path* $\pi(\omega)$ that corresponds to ω : the nodes of this path correspond to trees T_i s of F . Any node ordering in $\pi(\omega)$ can be used.

Consider the collection of paths $\Pi(F)$ associated with the nodes of tree \mathcal{F} . The children of the root path $\pi(\omega)$ are solid paths. The children of a solid path are either solid or dashed paths. The children of a dashed path are solid paths (see Appendix, Figure 1(b)).

Using this tree of paths, we implement our data structure as follows. Each path (root, solid or dashed) is implemented through the path hash accumulator authentication scheme (Section 3), where additionally, any accumulator is realized by a *biased binary tree* ([3]). A path property \mathcal{P} , i.e., a collection of path attributes that satisfies the concatenation criterion, is defined. By the implicit path interconnection, through the idea of setting path properties as node attributes, $\mathcal{P}(p)$ includes, in general, information about the children paths of p . We include path attributes in node properties, as follows. If v is a node of path p and $L(p)$ denotes the path hash accumulation of path p , then: (i) if p is the root path or a dashed path, then $L(p')$ and $tail(p')$ are included in $\mathcal{N}(v)$, where p' is the child solid path corresponding to node v and (ii) if p is a solid path, then $L(p')$ and $tail(p')$ are included in $\mathcal{N}(v)$ when v corresponds to a solid child path p' , or $L(p')$ is included in $\mathcal{N}(v)$ when v corresponds to a dashed child path p' . The above scheme yields a digest for the forest F , namely, the path hash accumulation of the root path $\pi(\omega)$ of \mathcal{F} .

Theorem 1. *Given a forest F with n nodes, there exists a fully dynamic authenticated data structure of size $O(n)$ that supports path property queries on F with the following performance: (i) operations *destroyTree* and *newTree* take $O(1)$ time; operations *link* and *cut* each take $O(\log n)$ time; operation *property* takes $O(\log n)$ time; (ii) the answer authentication information for operation *property* has size $O(\log n)$; (iii) the answer verification time for operation *property* is $O(\log n)$. [A]*

Theorem 1 yields an authenticated data structure to answer the following queries on a dynamic forest: *path* (reports the path, if any, between two nodes), *areConnected* (answers the question “is there a path between two nodes?”), *pathLength* (reports the length of the path, if any, between two nodes) and *type*

(answers the question “is there a node of a given type in a path between two nodes?”).

Theorem 2. *Given a forest F with n nodes, there exists a fully dynamic authenticated data structure of size $O(n)$ that supports path, connectivity and type queries with the following performance, where k is the length of the path returned by operation `path`: (i) operations `destroyTree` and `newTree` take $O(1)$ time; operations `link` and `cut` each take $O(\log n)$ time; operations `areConnected`, `pathLength`, `type` each take $O(\log n)$ time; operation `path` takes $O(\log n + k)$ time; (ii) the answer authentication information and the answer verification time of each operation have size proportional to its query time. [A]*

We can apply Theorem 2 to design authenticated data structures for the following queries on a graph that evolves through node and edge insertions: `path` (reports the path, if any, between two nodes), connectivity, biconnectivity and triconnectivity (report whether two nodes are in the same connected, biconnected or triconnected component).

Theorem 3. *Given a graph G with n nodes that evolves through node and edge insertions, there exist authenticated data structures of size $O(n)$ that support path, connectivity, biconnectivity and triconnectivity queries with the following performance: (i) operation `path` takes $O(\log n + k)$ time, where k is the path length; (ii) all other queries take $O(\log n)$ time; (iii) the answer authentication information size and the answer verification time are proportional to the query time; (iv) a node insertion takes $O(1)$ time; (v) an edge insertion takes amortized $O(\log n)$ time. [A]*

5 Authenticated Geometric Searching

In this section, we consider authenticated data structures for geometric searching problems. Our main result is the authentication of the *fractional cascading* technique [10], which solves the *iterative search* problem.

Let U be an ordered universe and $\mathcal{C} = \{C_1, C_2, \dots, C_k\}$ a collection of k catalogs, where each catalog C_i is an ordered collection of n_i elements chosen from U . For any element $x \in U$, the *successor* of x in C_i is defined to be the smallest element in C_i that is greater than or equal to x . We say that we *locate* x in C_i when we find the successor of x in C_i . In the *iterative search* problem, given an element $x \in U$, we want to locate x in each catalog of \mathcal{C} . If $n = \sum_{i=1}^k n_i$ is the total number of stored elements, the fractional cascading technique succeeds in solving the iterative search problem in $O(\log n + k)$ time using $O(n)$ storage. The idea is to consider the catalogs as nodes in a connected graph and to preprocess them so that pairs of neighboring catalogs are correlated. One can perform a binary search to locate x in some catalog, and then, using the underlying graph, locate x in all the other catalogs by moving along neighboring ones and spending $O(1)$ time for each node transition.

The collection \mathcal{C} is associated to a graph as follows. Let G be a *single-source* directed acyclic graph that has bounded degree, i.e., each node of G has both in-degree and out-degree bounded by a constant d . Each node v of G is associated with a catalog C_v of \mathcal{C} and G is called the *catalog* graph of \mathcal{C} . Let s be the unique source of G . Given G , we consider a subgraph Q of G that contains s and does not contain any other node having zero in-degree in Q . The iterative search problem for the catalog graph G can then be restated as follows: given an element $x \in U$ and a subgraph Q , locate x in C_v for each vertex v of Q . We refer to Q as the *query graph*.

The fractional cascading data structure works as follows. Each catalog C_v is augmented to a catalog A_v by storing some extra elements. In A_v , elements in C_v are called *proper* and the extra elements are called *non-proper*. Augmented catalogs that correspond to adjacent nodes of G are connected via *bridges*. Let $e = (u, v)$ be an edge of G . A bridge connecting A_u and A_v is a pair (y, z) associating two non-proper elements y and z , where $y \in A_u$, $z \in A_v$ and $y = z$. Each non-proper element y belongs to exactly one bridge. Two neighboring catalogs A_u and A_v are connected through at least two extreme bridges that correspond to non-proper elements $+\infty$ and $-\infty$ respectively. Each pair of neighboring bridges $(y, z), (y', z')$ of edge (u, v) defines a *block* B which contains all the elements of A_u and A_v lying between the two bridges. If $y' \leq y$, then bridges (y, z) and (y', z') are respectively called the *higher* and the *lower* bridge of B . The size $|B|$ of a block B is the number of elements (both proper and non-proper) contained in B . For each block B , we have $\alpha d \leq |B| \leq \beta d$, where α and β are constants. Each element, proper or not, $z \in A_v$ is associated to its successor in the original catalog C_v , so, if z is proper then $\text{proper}(z) = z$, otherwise $\text{proper}(z)$ is the next proper element in A_v (see Appendix, Figure 2(b)).

Using this data structure, given a query element x and a query graph Q , the iterative search problem can be solved in $O(\log n + k)$ time, where k is the number of vertices of Q and n is the total number of proper elements in the catalogs of \mathcal{C} . Using binary search, x is located at the augmented root catalog A_s (Q always contains s) in time $O(\log n)$. Q is then traversed by considering any topological order of the nodes of Q , so that each node of Q is visited exactly once. A step from a node u to an neighbor v is performed in $O(1)$ time: if x_u is the successor of x at A_u , starting from x_u , we traverse A_u moving to higher elements until a bridge, say (y, z) , is reached that connects to A_v ; we then follow this bridge and finally traverse A_v moving to smaller elements until x is located in A_v . Bridge (y, z) is the *entrance* bridge of catalog A_v .

We now describe our hashing scheme over the complete data structure. The hashing scheme can be viewed as a two-level hashing structure, built using the path hash accumulator scheme: *intra-block* hashing is performed within each block defined in the data structure and *inter-block* hashing is performed over all the blocks of the data structure.

For the intra-block hashing, consider any edge (u, v) of G , i.e., u is one of the parents of v . Also, consider any two neighboring bridges (y', z') and (y, z) that define block B . Assume that $z, z' \in A_v$. We define P to be the sequence of

elements of B that exist in A_v plus the non-proper elements of the corresponding bridges that lie in A_v . That is, $P = \{p_1, p_2, \dots, p_t\}$ is an increasing sequence, where, if $z' \leq z$, $p_1 = z'$ and $p_t = z$. We refer to P as the *hash side* of B . Using the path hash accumulator scheme, we compute the digest $D(P)$ of sequence P . For each element p_i , we set $\mathcal{N}(p_i) = \{p_i, \text{proper}(p_i), v\}$ and in that way the path hash accumulator can support authenticated membership queries and authenticated path property queries. Here, one property of P is the corresponding node v . We iterate the process for all blocks defined in the data structure: for each block B having a hash side P in A_v , H_B is the hash path accumulation $D(P)$ of sequence P . We also define B_s to be a fictitious block, the augmented catalog A_s . The hash side of B_s is the whole block itself, so H_{B_s} is well defined. All the path hash accumulators used define the first level hashing structure.

The inter-block hashing is defined through a directed acyclic graph \mathcal{H} defined over blocks. That is, nodes of \mathcal{H} are blocks of the data structure. Suppose that w is a parent of u and u is a parent of v in G . If B is a block of an edge (u, v) , then we add to the set of edges of \mathcal{H} all the directed edges (B, B') , where B' is a block of edge (w, u) that shares elements from A_u with B . Additionally, if v is a child of the root s in G , then for all blocks B that correspond to edge (s, v) , we add to the set of edges of \mathcal{H} all the directed edges (B, B_s) . The construction of \mathcal{H} is now complete. B_s is the unique root of \mathcal{H} (see Appendix, Figure 2(b)). Each block (node) B of \mathcal{H} is associated with a label $L(B)$. If B is a leaf (sink) in \mathcal{H} , then $L(B) = H_B$. If B is the parent of blocks B_1, B_2, \dots, B_t in \mathcal{H} , listed in arbitrary order, then $L(B)$ equals the path hash accumulation over B_1, B_2, \dots, B_t using $\mathcal{N}(B_i) = \{B_i, H_{B_i}\}$. This hashing scheme over \mathcal{H} corresponds to the second level hashing structure. Finally, we set $D(\mathcal{D}) = L(B_s)$ to be the digest of the entire data structure \mathcal{D} , which is time-stamped and digitally signed by the source.

Let x be a query element and Q be a query graph. If v is a node of Q , let s_v be the successor of x in C_v . In the location process, to locate x in A_v , we find two consecutive elements y and z of A_v such that $y \leq x < z$, where each of y and z may be either proper or non-proper. They are both elements of a block B such that the entrance bridge of A_v is the higher bridge of B . Observe that z is the successor of x in A_v and that $s_v = \text{proper}(y)$ when $y = x$, or $s_v = \text{proper}(z)$ when $y < x$. We call z and B , the *target element* and the *target block* of A_v , respectively. In the location process, the traversal of Q is chosen so that each of its nodes is visited once. Also, any two visited target blocks that correspond to incident edges in Q share elements of the common augmented catalog so they are adjacent in graph \mathcal{H} . Thus, all target blocks define a subgraph T of \mathcal{H} . Edges of T are the edges of \mathcal{H} that connect neighboring target blocks (see Appendix, Figure 2(b)). For any query graph Q , graph T is a tree: the topological order used for the traversal of Q defines a directed subtree T_Q of Q and there is a correspondence between edges of T_Q and target blocks.

We can now describe the answer authentication information: (i) (intra-block) for each node v of Q , the target element z_v of A_v and a verification sequence p_v from z_v up to the path hash accumulation of the hash side of B_v , and (ii) (inter-

block) for every target block B_v of T that is not a leaf, the verification sequences from every child of B_v in T up to the pash hash accumulation $L(B_v)$.

Lemma 2. *If n is the total number of proper elements in the catalogs of \mathcal{C} and d is the bounded degree of G , then for any query graph Q of k nodes, the size of the answer authentication information is $O(\log n + k \log d) = O(\log n + k)$. [A]*

Given elements x, y, y', z, z' and a node v of Q , consider the predicates: (1) $y \leq x < z$, (2) y and z are consecutive elements in A_v , (3) $y = x$ and $y' = \text{proper}(y)$ in A_v and (4) $y < x$ and $z' = \text{proper}(z)$ in A_v . If (1), (2) and (3) hold simultaneously, then they constitute a proof that the successor of x in C_v is y' , whereas if (1), (2) and (4) hold simultaneously, they constitute a proof that the successor of x in C_v is z' . Assume that the answer given to the user is a set $A = \{(a_v, v) : v \text{ node of } Q\}$, where a_v is claimed to be the successor of x in C_v . Given A , x and the answer authentication information, the user first checks if there is any inconsistency between values a_v and z_v for every v of G with respect to the two possible proofs above. Observe that, by the answer authentication information, the user knows for each node v of Q the target element z_v and the corresponding element y_v , where $y_v < z_v$ and y_v and z_v are consecutive elements in A_v . If there is at least one inconsistency, the user rejects the answer. Otherwise, all that is needed is to verify the signed digest $D(\mathcal{D})$ of the data structure. If the digest is verified, based on the collision-resistance property of the hash function used in the scheme, the user has a proof that the answer is correct.

Lemma 3. *If n is the total number of proper elements in the catalogs of \mathcal{C} , then both intra-block and inter-block hashing schemes can be computed in $O(n)$ time using $O(n)$ storage. Also, for any query graph Q of k nodes, the answer verification time is $O(\log n + k)$. [A]*

We have thus proved the following theorem.

Theorem 4. *Let \mathcal{C} be a collection of k catalogs and n be the total number of elements stored in \mathcal{C} . If G is the catalog graph of \mathcal{C} and G is of bounded degree, then the authenticated fractional cascading data structure \mathcal{D} for G solves the authenticated iterative search problem for G with the following performance: \mathcal{D} can be constructed in $O(n)$ time and uses $O(n)$ storage; given an element x and a query graph Q with k vertices, x can be located in every catalog of Q in $O(\log n + k)$ time; the answer authentication information has size $O(\log n + k)$ and the answer verification time is $O(\log n + k)$.*

Theorem 4 provides the basis to design authenticated data structures for various fundamental two-dimensional geometric search problems, where iterative search is implicitly performed on a bounded degree catalog graph (see [11]).

Theorem 5. *There are authenticated data structures that can be constructed in $C(n)$ time, use $S(n)$ storage, answer queries in $Q(n, k)$ time and have answer authentication information size and answer verification time each $A(n, k)$ for*

the following two-dimentional geometric searching problems: (i) line intersection queries, where $C(n)$ is $O(n \log n)$, $S(n)$ is $O(n)$, and $Q(n, k)$ and $A(n, k)$ are $O((k + 1) \log \frac{n}{k+1})$; (ii) ray shooting and point location queries, where $C(n)$ is $O(n \log n)$, $S(n)$ is $O(n)$, and $Q(n, k)$ and $A(n, k)$ are $O(\log n)$; (iii) orthogonal range search, orthogonal point enclosure and orthogonal intersection queries, where $C(n)$ and $S(n)$ are $O(n \log n)$, and $Q(n, k)$ and $A(n, k)$ are $O(\log n + k)$. Here, n is the problem input size and k is the query output size.

References

- [1] A. Anagnostopoulos, M. T. Goodrich, and R. Tamassia. Persistent authenticated dictionaries and their applications. In *Proc. Information Security Conference (ISC 2001)*, volume 2200 of *LNCS*, pages 379–393. Springer-Verlag, 2001. [297](#)
- [2] S. Ar, M. Blum, B. Codenotti, and P. Gemmell. Checking approximate computations over the reals. In *Proc. ACM Symp. on the Theory of Computing*, pages 786–795, 1993. [295](#)
- [3] S. W. Bent, D. D. Sleator, and R. E. Tarjan. Biased search trees. *SIAM J. Comput.*, 14:545–568, 1985. [302](#), [310](#)
- [4] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, Jan. 1995. [295](#)
- [5] J. D. Bright and G. Sullivan. Checking mergeable priority queues. In *Digest of the 24th Symposium on Fault-Tolerant Computing*, pages 144–153. IEEE Computer Society Press, 1994. [295](#)
- [6] J. D. Bright and G. Sullivan. On-line error monitoring for several data structures. In *Digest of the 25th Symposium on Fault-Tolerant Computing*, pages 392–401. IEEE Computer Society Press, 1995. [295](#)
- [7] J. D. Bright, G. Sullivan, and G. M. Masson. Checking the integrity of trees. In *Digest of the 25th Symposium on Fault-Tolerant Computing*, pages 402–411. IEEE Computer Society Press, 1995. [295](#)
- [8] A. Buldas, P. Laud, and H. Lipmaa. Accountable certificate management using undeniable attestations. In *ACM Conference on Computer and Communications Security*, pages 9–18. ACM Press, 2000. [297](#)
- [9] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proc. CRYPTO*, 2002. [297](#)
- [10] B. Chazelle and L. J. Guibas. Fractional cascading: I. A data structuring technique. *Algorithmica*, 1(3):133–162, 1986. [297](#), [303](#)
- [11] B. Chazelle and L. J. Guibas. Fractional cascading: II. Applications. *Algorithmica*, 1:163–191, 1986. [298](#), [306](#)
- [12] R. F. Cohen and R. Tamassia. Combine and conquer. *Algorithmica*, 18:342–362, 1997. [299](#)
- [13] P. Devanbu, M. Gertz, A. Kwong, C. Martel, G. Nuckolls, and S. Stubblebine. Flexible authentication of XML documents. In *Proc. ACM Conference on Computer and Communications Security*, 2001. [297](#)
- [14] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic third-party data publication. In *Fourteenth IFIP 11.3 Conference on Database Security*, 2000. [297](#)
- [15] O. Devillers, G. Liotta, F. P. Preparata, and R. Tamassia. Checking the convexity of polytopes and the planarity of subdivisions. *Comput. Geom. Theory Appl.*, 11:187–208, 1998. [295](#)

- [16] G. Di Battista and G. Liotta. Upward planarity checking: “Faces are more than polygons”. In S. H. Whitesides, editor, *Graph Drawing (Proc. GD ’98)*, volume 1547 of *Lecture Notes Comput. Sci.*, pages 72–86. Springer-Verlag, 1998. [295](#)
- [17] G. Di Battista and R. Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15:302–318, 1996. [312](#)
- [18] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. Westbrook, and M. Yung. Maintenance of a minimum spanning forest in a dynamic plane graph. *J. Algorithms*, 13(1):33–54, 1992. [311](#)
- [19] U. Finkler and K. Mehlhorn. Checking priority queues. In *Proc. 10th ACM-SIAM Symp. on Discrete Algorithms*, pages S901–S902, 1999. [295](#)
- [20] I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *Int. Workshop on Practice and Theory in Public Key Cryptography (PKC ’2000)*, volume 1751 of *LNCS*, pages 342–353. Springer-Verlag, 2000. [297](#)
- [21] M. T. Goodrich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing. Technical report, Johns Hopkins Information Security Institute, 2000. <http://www.cs.brown.edu/cgc/stms/papers/hashskip.pdf>. [296](#), [297](#), [311](#)
- [22] M. T. Goodrich, R. Tamassia, and J. Hasic. An efficient dynamic and distributed cryptographic accumulator. In *Proc. Int. Security Conference (ISC 2002)*, volume 2433 of *LNCS*, pages 372–388. Springer-Verlag, 2002. [297](#)
- [23] M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001. [297](#)
- [24] V. King. A simpler minimum spanning tree verification algorithm. In *Workshop on Algorithms and Data Structures*, pages 440–448, 1995. [295](#)
- [25] P. C. Kocher. On certificate revocation and validation. In *Proc. Int. Conf. on Financial Cryptography*, volume 1465 of *LNCS*. Springer-Verlag, 1998. [296](#)
- [26] P. Maniatis and M. Baker. Enabling the Archival Storage of Signed Documents. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST 2002)*, Monterey, CA, USA, 2002. [297](#)
- [27] P. Maniatis and M. Baker. Secure History Preservation Through Timeline Entanglement. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, USA, 2002. [297](#)
- [28] C. Martel, G. Nuckolls, P. Devanbu, M. Gertz, A. Kwong, and S. Stubblebine. A general model for authentic data publication, 2001. <http://www.cs.ucdavis.edu/~devanbu/files/model-paper.pdf>. [296](#), [297](#), [298](#)
- [29] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, Cambridge, UK, 2000. [295](#)
- [30] K. Mehlhorn, S. Näher, M. Seel, R. Seidel, T. Schilz, S. Schirra, and C. Uhrig. Checking geometric programs or verification of geometric structures. *Comput. Geom. Theory Appl.*, 12(1–2):85–103, 1999. [295](#)
- [31] R. C. Merkle. Protocols for public key cryptosystems. In *Proc. Symp. on Security and Privacy*, pages 122–134. IEEE Computer Society Press, 1980. [296](#)
- [32] R. C. Merkle. A certified digital signature. In G. Brassard, editor, *Proc. CRYPTO ’89*, volume 435 of *LNCS*, pages 218–238. Springer-Verlag, 1990. [296](#)
- [33] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proc. 7th USENIX Security Symposium*, pages 217–228, Berkeley, 1998. [296](#), [297](#)
- [34] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, Oct. 1990. [298](#)

- [35] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990. [297](#)
- [36] D. D. Sleator and R. E. Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–381, 1983. [301](#), [310](#)
- [37] G. F. Sullivan and G. M. Masson. Certification trails for data structures. In *Digest of the 21st Symposium on Fault-Tolerant Computing*, pages 240–247. IEEE Computer Society Press, 1991. [295](#)
- [38] G. F. Sullivan, D. S. Wilson, and G. M. Masson. Certification of computational results. *IEEE Trans. Comput.*, 44(7):833–847, 1995. [295](#)
- [39] J. Westbrook and R. E. Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7:433–464, 1992. [311](#)

Appendix

Proof of Lemma 1 Consider the query property($\bar{p}(v, u)$) on p . Moreover, consider the set $\mathcal{A}(\bar{p})$ of *allocation nodes* in $T(p)$ of subpath $\bar{p} = \bar{p}(v, u)$. For a tree node w , $w \in \mathcal{A}(\bar{p})$ if the leaves of the subtree defined by w are all nodes of \bar{p} but the same is not the case for w 's parent, if any. Observe that each $w \in \mathcal{A}(\bar{p})$ corresponds to a subpath of path \bar{p} . Since $T(p)$ is a balanced binary tree, there are $O(\log n)$ allocation nodes for subpath \bar{p} that can be found in $O(\log n)$ time by tracing the leaf-to-root tree paths in $T(p)$ from v and u up to r . Since, the path property \mathcal{P} satisfies the concatenation criterion, we have that the path property $\mathcal{P}(\bar{p})$ can be computed by using the tree structure and by applying $O(\log n)$ times the concatenation function \mathcal{F} .

Clearly, the answer given to the user is the property $\mathcal{P}(\bar{p})$. For any node u of $T(p)$, let (u_1, \dots, u_k) be the node-to-root tree path connecting u with the root r of $T(p)$. We define the *verification sequence* of u to be the sequence $\mathcal{V}(u) = (s_1, s_2, \dots, s_k)$, where s_j , $1 \leq j \leq k$, is the label of the sibling of node u_j . The answer authentication information consists of three parts: (1) For each allocation node w of subpath $\bar{p} = \bar{p}(v, u)$, the property $\mathcal{P}(w)$; these properties are given as a sequence $(\alpha_1, \dots, \alpha_m)$, such that the set of leaf nodes of any allocation nodes α_i and α_{i+1} , $1 \leq i \leq m - 1$, forms a subpath of \bar{p} ; (2) For each $w \in \mathcal{A}(\bar{p})$ the labels of its children, if they exist; and the label of the siblings of the left most and right most allocation nodes of $\bar{p} = \bar{p}(v, u)$; (3) If z is the least common ancestor of v and u , the verification sequence of z .

To accept the answer, the user first recomputes $\mathcal{P}(\bar{p})$, by repeatedly applying the concatenation function \mathcal{F} on sequence $(\alpha_1, \dots, \alpha_m)$. If $\mathcal{P}(\bar{p})$ is not verified, the answer is rejected. Otherwise, the verification process is completed by the computation and verification of the path hash accumulation. Since, $\text{head}(p) \in \mathcal{P}(p)$ and $\text{tail}(p) \in \mathcal{P}(p)$, we achieve the desired security result; it can be shown that in order for an incorrect answer to be accepted as correct, at least one collision on the cryptographic hash function h must be computed.

For $\text{property}(v)$ query, we proceed as above; observe that $\text{property}(v)$ corresponds to $\text{property}(v, v)$. For $\text{locate}(p, \sigma, q)$ query, we locate the target node v by performing a top-down search in $T(p)$ starting from the root: at a node u with children w_1 and w_2 , the path selection function σ is used to select either the path

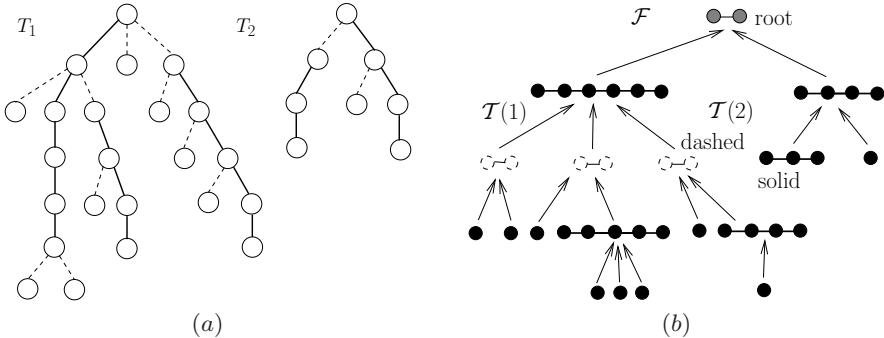


Fig. 1. (a) The Partition of trees into solid paths. (b) Trees of paths and final tree \mathcal{F}

that corresponds to w_1 or the path that corresponds to w_2 . Then, the answer is the located node v and the proof is the proof that corresponds to a $\text{property}(v,v)$ operation. For any $\text{subpath}(p,\alpha,q)$ query, a similar top-down tree search is performed using the path advance function α to first compute the target subpath; the proof is constructed by considering the corresponding allocation nodes. \square

Proof of Theorem 1 Let $\text{size}(v)$ denote the number of nodes in the subtree defined by v and let u the parent node of v . Edge $e = (u, v)$ is called *heavy* if $\text{size}(u) > \text{size}(v)/2$. The edge labeling of a dynamic tree T of n nodes with root w , such that an edge is labeled solid only if it is heavy, has the following important property: for any node u of T there are at most $\log n$ dashed edges on the path from u to w . We use this edge labeling to partition T into solid paths.

Consider all the paths that correspond to the final tree \mathcal{F} (after the dashed paths and the root path have been added). Each path p of \mathcal{F} is implemented as a biased binary tree $T(p)$ (see [3]), where node weights are defined using function $\text{size}()$. We consider the weight $w(v)$ of node v to be either a node or a path property (depending on whether v is a leaf node in $T(p)$ or not). If p is a path having no child path (μ_p is a leaf in \mathcal{F}), then $w(v) = \text{size}(v)$. Otherwise (μ_p is not a leaf in \mathcal{F}), $w(v) = w(u_1) + w(u_2)$, if v is internal node of $T(p)$ with children u_1, u_2 . Otherwise, v is a node of path p . If $v = \text{head}(p)$ and p is solid, then $w(v) = w(u_1) + w(u_2)$, where u_1, u_2 are the children of v in T (through dashed edges). If $v \neq \text{head}(p)$ and p is solid, then $w(v) = w(u) + 1$, where u is the unique dashed child of v and $w(u) = 0$, if no such child exists. If p is dashed, $w(v) = w(u) + 1$, where u is the node connected with v with the corresponding dashed edge. If p is root path, again $w(v) = w(u) + 1$, where u is the root of the corresponding tree root.

Using the above biasing and since the path property \mathcal{P} satisfies the concatenation criterion, it can be shown that any leaf-to-root path in \mathcal{F} , when considered *through* the individual biased trees $T(p)$ s, has length $O(\log n)$. The proof is based in the analysis in [36]. All operations correspond to accessing and

modifying paths of this kind (through the primitive path operations `split` and `concatenate`). In particular, operations `link()` and `cut()` can be implemented in $O(\log n)$ time by modifying only $O(\log n)$ path hash accumulators and by examining, modifying and restructuring only $O(\log n)$ nodes in total. Restructuring means connecting a node to new children.

For query $\text{property}(u, v)$, we first determine whether nodes u and v are in forest F . We can use the authenticated data structure of [21] that supports containment queries in $O(\log n)$ time with responses of length $O(\log n)$. If v and u are in F , the path property query is performed by accessing three *multipaths* in \mathcal{F} , that is, paths on paths of \mathcal{F} . Let μ_u and μ_v be the paths of \mathcal{F} that contain u and v and let μ_l be the least common ancestor μ_u and μ_v in \mathcal{F} (μ_l may overlap with μ_v and/or μ_u). Let $\Pi(u)$, $\Pi(v)$ be the multipaths from μ_u , μ_v to μ_l and $\Pi(l)$ be the multipath from μ_l to the root $\pi(\omega)$ of \mathcal{F} . Following $\Pi(u)$ and $\Pi(v)$, we compute the answer $A(u, v)$ considering the allocation nodes of each path hash accumulator visited. Similarly, we provide property proofs $\text{proof}(\Pi(u))$, $\text{proof}(\Pi(v))$ (each consists of subproofs: a subproof for each path hash accumulator that is visited). Finally, following $\Pi(l)$ up to $\pi(\omega)$, we compute the *multipath verification sequence* $\mathcal{V}(l)$ that allows the user to recalculate the signed hash value (hash path accumulation of $\pi(\omega)$) given $A(u, v)$, $\text{proof}(\Pi(u))$ and $\text{proof}(\Pi(v))$. By the biased scheme used over \mathcal{F} , the set of allocation nodes is $O(\log n)$, thus, the answer $A(u, v)$ and the proof $(\mathcal{V}(l), \text{proof}(\Pi(u)), \text{proof}(\Pi(v)))$ each have size $O(\log n)$. The verification time is also $O(\log n)$. The hashing scheme is based on the path hash accumulator. By allowing neighboring paths to share information (properties) we achieve the desired security results: any attack to our data structure can be reduced to a collision on the cryptographic hash function h . \square

Proof Sketch of Theorem 2 All queries correspond to a path property. `areConnected` corresponds to the existence of a node of the root path $\pi(\omega)$ in the path p_{vu} connecting v and u in \mathcal{F} (p_{vu} always exists). This property is expressed by assigning a unique *id* value to every path in \mathcal{F} . A similar idea is applied for query type. For `pathLength`, we include in $\mathcal{P}(v)$ the number of leaves in the subtree defined by v . `path` is answered by first performing a query `areConnected`. If there is a path, it can be found by answering a path property query where $\mathcal{P}(v)$ includes all the leaves of the subtree T_v defined by v . To this end, we need a slightly different definition for the path attribute. Since $\mathcal{P}(v) = O(\text{size}(T_v))$, the introduced complexity is $O(\log n + k)$, where k is the path length. \square

Proof Sketch of Theorem 3 For path and connectivity queries, the main idea is to maintain a spanning forest of the graph. In fact, for embedded planar graphs, we can also support deletions using the data structure described in [18]. To answer biconnectivity queries, we extend the data structure of [39]. We maintain the *block-cutvertex forest* \mathcal{B} of G . Each tree T in \mathcal{B} corresponds to a connected component of G . There are two types of nodes in T : *block nodes* that correspond to blocks (biconnected components) of G and *vertex nodes* that correspond to vertices of G . Each edge of T connects a vertex node to a block node. The block node associated with a block B is adjacent to the vertex nodes associated with

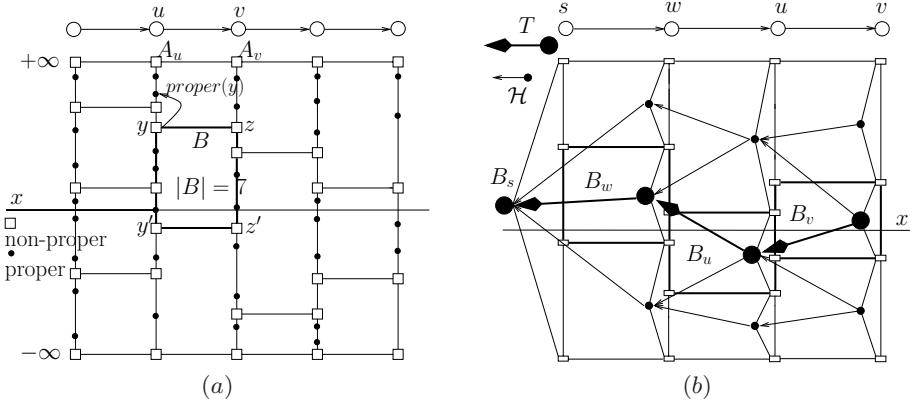


Fig. 2. (a) The fractional cascading data structure over a path. Squares and dots represent non-proper and proper elements respectively. Edge (u, v) has three blocks. (b) Inter-block hashing: DAG H defines the second level hashing. For any query element x , any query graph Q and any traversal of Q , the target blocks define a tree T

the vertices of B . We have that two vertices u and v of G are in the same biconnected component if and only if there is a path between the vertex nodes of B associated with u and v and this path has length two. Thus, operation `areBiconnected` in G is reduced to operation `pathLength` in B .

For triconnectivity queries, we extend the data structure of [17], where a biconnected graph (or component) G is associated with an *SPQR tree* T that represents a recursive decomposition of G by means of separation pairs of vertices. Each S-, P-, and R-node of T is associated with a triconnected component C of G and stores a separation pair (s, t) , where vertices s and t are called the *poles* of C . A Q-node of T is associated with an edge of G . Each vertex v of G is allocated at several nodes of T and has a unique *proper allocation node* in T . Our authenticated data structure augments tree T with V-nodes associated with the vertices of G and connects the V-node of a vertex v to the proper allocation node of v in T . Using node attributes to store the type (S, P, Q, R, or V) of a node of T and its poles, we can show that query `areTriconnected` is reduced to a small number of `pathLength` and type queries on the augmented SPQR tree. \square

Proof of Lemma 2 The hash side of B_s has size $|A_s| = O(n)$ and the hash side of any other target block has size $O(d)$. Thus, the intra-block answer authentication information consists of k verification sequences, $k - 1$ of length $O(\log d)$ and one of length $O(\log n)$, and, thus, has size $O(\log n + k \log d) = O(\log n + k)$ size. For the inter-block part, recall that G and, thus, both Q and T_Q , have out-degree bounded by d and that every target block can share elements with at most $O(d)$ other target blocks. Thus, H and T have in-degree bounded by $O(d)$. Now, all, but $L(B_s)$, the second level path hash accumulations are built

over sequences of length $O(d)$. $L(B_s)$ is built over at most dn blocks that share elements with A_s . Observe that there is a one-to-one correspondence between inter-block verification sequences and edges in T . It follows that the inter-block answer authentication information consists of $k - 2$ verification sequences of size $\log d$ and one of size $\log n$, thus, has $O(\log n + k \log d) = O(\log n + k)$ size. \square

Proof Sketch of Lemma 3 G has $O(1)$ in-degree and every target block can share elements with at most $O(1)$ other blocks. Moreover, the path hash accumulation of a sequence of length m can be computed in $O(m)$ time and uses $O(m)$ space. Finally, recall that the verification time of a path hash accumulator is proportional to the size of the verification sequence. \square

Fractal Merkle Tree Representation and Traversal

Markus Jakobsson¹, Tom Leighton^{2,3}, Silvio Micali³, and Michael Szydlo¹

¹ RSA Laboratories, Bedford, MA 01730

{mjakobsson,mszydlo}@rsasecurity.com

² MIT Laboratory for Computer Science, Cambridge, MA 02139

³ Akamai Technologies, Cambridge, MA 02142

Abstract. We introduce a technique for traversal of Merkle trees, and propose an efficient algorithm that generates a sequence of leaves along with their associated authentication paths. For one choice of parameters, and a total of N leaves, our technique requires a worst-case computational effort of $2 \log N / \log \log N$ hash function evaluations per output, and a total storage capacity of less than $1.5 \log^2 N / \log \log N$ hash values. This is a simultaneous improvement both in space and time complexity over any previously published algorithm.

Keywords: Amortization, authentication, fractal, Merkle tree.

1 Introduction

A Merkle tree [8] is a tree where the value associated with a node is a one-way function of the values of the node’s children. Merkle trees find a wide range of applications within cryptography, due to their simplicity and versatility. For many applications, one wishes to output a sequence of consecutive leaves (or leaf pre-images), along with their “authentication paths” – the latter consists of the interior nodes that constitute the siblings on the path from the leaf to the root. Given an authentication path and a leaf, one can verify the correctness of the latter with respect to the publicly known root value.

However, as elegant as Merkle trees are, they are used less than one might expect. One reason is known techniques for traversal of trees require a relatively large amount of computation, storage, or both. Such constraints make all but the smallest trees impractical, and in particular not very useful for small and powerless devices [11].

Our Contribution. We propose a technique for traversal of Merkle trees which is structurally very simple and allows for various tradeoffs between storage and computation. For one choice of parameters, the total space required is bounded by $1.5 \log^2 N / \log \log N$ hash values, and the worst-case computational effort is $2 \log N / \log \log N$ hash function evaluations per output.

It should be noted that the use of our techniques is “transparent” to a verifier, who will not need to know how a set of outputs were generated, but only

that they are correct. Therefore, our technique can be employed in any construction for which the generation and output of consecutive leaf pre-images and corresponding authentication paths is required.

Related Work and Applications. Our technique relates to and improves on a previous result by Merkle [7], who proposed a technique for Merkle tree traversal requiring a maximum of $O(\log^2 N)$ space and $O(\log N)$ computation per output, where N is the number of leaves of the tree, and one unit of computation corresponds to one hash function evaluation. (An alternative – but less efficient – method was independently proposed by Vaudenay [15] some ten years later, where *average* instead of *worst-case* costs were considered.) Our improvement is achieved by means of a careful choice of what nodes to compute, retain, and discard at each stage.

Our result also relates to recent methods for fractal traversal of hash chains [3, 14]. The most notable similarities involve the storage and computation requirements and trade-offs, and the fractal scheduling pattern. On the other hand, there are large intrinsic differences between what needs to be computed. For a Merkle tree, the desired outputs are the consecutive authentication paths, while for a hash chain, the only output is a single element. Moreover, while all elements of a hash chain are determined by a *single* starting-value, the leaves of a Merkle tree may be selected independently (e.g., via a keyed pseudo-random generator).

The leaves of the tree may either be used one by one, or many at the same time. The former type of use is well suited for applications such as TESLA [10], certification refreshal [9], wireless security [2], and micro-payments [4, 12], while the latter type finds direct use for Merkle signatures [8, 5]. This partition of applications also corresponds to the birth of the techniques we describe; while the second and third author were motivated by the case relating to Merkle signatures, the first and fourth author focused on the general case.

Outline. We begin by reviewing the goals and standard algorithms of Merkle trees (section 2). We then introduce notation for our subtrees and describe the intuition and tools for their use in our solution (section 3). After that, we describe our technique on a more detailed level (section 4), followed by a correctness and complexity analysis (section 5). A small but technical improvement (section 6) yields our final result, followed by conclusions and ideas for further work (section 7).

2 Merkle Trees and Background

Binary Trees. We first fix notation to describe binary trees. We say that a complete binary tree T has *height* H if it has 2^H leaves, and $2^H - 1$ interior nodes. Each interior node has two children labeled “0” (left), and “1” (right). With this naming convention the leaves are naturally ordered, indexed according

to the binary representation of the path from the root to the leaf. Visually, the higher this *leaf index* in $\{0, 1, \dots, 2^H - 1\}$ is, the further to the right that leaf is. We define the *altitude* of any node n to be the height of the maximal subtree of T for which it is the root. The node heights range from 0 (leaves) to H (the root). As with the leaves, interior nodes of a given height h_0 may be assigned an index in $\{0, 1, \dots, 2^{h_0} - 1\}$.

Merkle Trees. A Merkle tree is a binary tree with an assignment of a string to each node: $n \mapsto P(n) \in \{0, 1\}^k$, such that the parent's node values are one-way functions of the children's node values.

$$P(n_{parent}) = \text{hash}(P(n_{left})||P(n_{right})) \quad (1)$$

In the above and onwards, *hash* denotes the one-way function; a possible choice of such a function is SHA-1 [13].

The value of a leaf, in turn, is a one-way function of some *leaf pre-image*. For small trees these pre-images may be simply stored; alternatively for larger trees, the leaves may be calculated with a keyed pseudo-random generator. Either way, in this paper we model a leaf calculation with an oracle *LEAF-CALC*, which is assumed to require computation equal in quantity to that of *hash*. The value of the root is considered public, while (to begin with) all the values associated with leaf pre-images are known by the “tree owner” alone.

Desired Output. We wish to generate a sequence of outputs, one for each leaf. Each output has two components; (1) a leaf pre-image; and (2) the *authentication path* of the leaf, i.e., the values of all nodes that are siblings of nodes on the path between the leaf in question and the root. This is illustrated in Figure 1. Visiting the leaves according to the natural indexing, (from left to right),

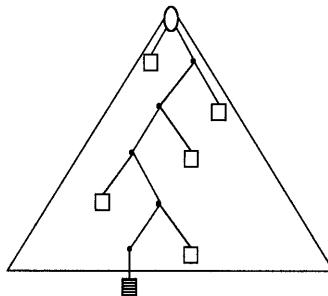


Fig. 1. The circle corresponds to the publicly known root value; the grey square to the current leaf pre-image; and the white squares to the current path siblings. The set of white squares make up the authentication path of the grey square

has the advantage that usually $leaf_i$ and $leaf_{i+1}$ share a large portion of their authentication paths.

In order to *verify* the value of a leaf pre-image, one computes (with Equation 1) the potential values of its ancestors by iterated hashing. A leaf pre-image is accepted as correct if and only if the computed root value is equal to the already known root value.

Digital Signatures. Merkle trees were originally presented as a method to convert a *one-time signature scheme* into a digital signature scheme [8] by using a block of $2k$ leaf pre-images as a one time secret key. The resulting scheme needs only the key to the pseudo random number generator as a secret key, and the root node value as the public key.

Computing Nodes: TREEHASH. A well-known technique used with Merkle trees is the use of an algorithm which computes the value $P(n)$ of a height H node, while only storing only up to $H + 1$ hash values. We use several variants of this *TREEHASH* algorithm, and recall this algorithm now to simplify the exposition of our traversal technique. Algorithm *TREEHASH* computes the value of a node n , assuming access to an oracle, *LEAF-CALC*, which returns the value of the leaf node with index $leaf \in \{0, \dots, 2^H - 1\}$. The idea is to compute the leaves sequentially, computing interior nodes whenever possible, and discarding nodes which are no longer needed. The algorithm essentially just stores the node values in a stack¹, and repeatedly applies Equation 1.

Algorithm 1: **TREEHASH** (maxheight)

1. Set $leaf = 0$ and create empty stack.
2. **Consolidate** If top 2 nodes on the stack are at the same height:
 - Pop node value P_{left} from stack.
 - Pop node value P_{right} from stack.
 - Compute $P_{parent} = \text{hash}(P_{left} || P_{right})$.
 - If height of $P_{parent} = \text{maxheight}$, output P_{parent} and stop.
 - Push P_{parent} onto the stack.
3. **New Leaf** Otherwise:
 - Compute $P_{leaf} = \text{LEAF-CALC}(leaf)$.
 - Increment $leaf$.
4. Loop to step 2.

Algorithm *TREEHASH* requires a total of $2^{\text{maxheight}} - 1$ computational units for a tree of height maxheight , assuming we count *hash* computations and *LEAF-CALC* computations equally. Fortunately, due to the fact that nodes are discarded when no longer needed, *TREEHASH* only requires storage of $\text{maxheight} + 1$ hash values at any stage. This is because at most one height may have two pebbles; the rest have at most one each. This bound is important

¹ The use of a stack to simplify the algorithm description was influenced by recent work on time-stamping [6], which also relates to hash trees.

in situations where a larger algorithm computes *TREEHASH* incrementally, applying some number of computational units to the iteration of steps 2 to 4 above, and modifying the state of the algorithm’s stack. Such intermediate *pebbles* in the stack are said to comprise the *tail* of the node calculation. For a node n at height h_0 , we express this bound as

$$\text{Space Tail}(n) \leq h_0 + 1. \quad (2)$$

We describe our uses and variants of *TREEHASH* as needed.

3 Subtree Notation and Intuition

The crux of our algorithm is the selection of which node values to compute and retain at each step of the output algorithm. We describe this selection by using a collection of subtrees of fixed height h . We begin with some notation and then provide the intuition for the algorithm.

3.1 Notation

Starting with a Merkle tree T of height H , we introduce further notation to deal with subtrees. First we choose a *subtree height* $h < H$. We let the altitude of a node n in T be the length of the path from n to a leaf of T (where, therefore, the altitude of a leaf of T is zero). Consider a node n with altitude at least h . We define the h -subtree at n to be the unique subtree in T which has n as its root and which has height h . For simplicity in the suite, we assume h is a divisor of H , and let the ratio, $L = H/h$, be the number of *levels* of subtrees. We say that an h -subtree at n is “at level i ” when it has altitude ih for some $i \in \{1, 2, \dots, H\}$. For each i , there are 2^{H-ih} such h -subtrees at level i .

We say that a series of h -subtrees $\{Tree_i\}$ ($i = 1 \dots L$) is a *stacked series of h -subtrees*, if for all $i < L$ the root of $Tree_i$ is a leaf of $Tree_{i+1}$. We illustrate our subtree notation and provide a visualization of a *stacked series of h -subtrees* in Figure 2.

3.2 Existing and Desired Subtrees

Pebbles. We say that we place a *pebble* on a node n of the tree T when we store the value $P(n)$ associated with this node.

Static View. As previously mentioned, we store some portion of the node values, and update what values are stored over time. Specifically, during any point of the output phase, there will exist a series of stacked *existing* subtrees, as in Figure 2. There are always L such subtrees $Exist_i$ for each $i \in \{1, \dots, L\}$, with pebbles on each of their nodes (except their roots). By design, for any leaf in $Exist_1$, the corresponding authentication path is completely contained in the stacked set of existing subtrees.

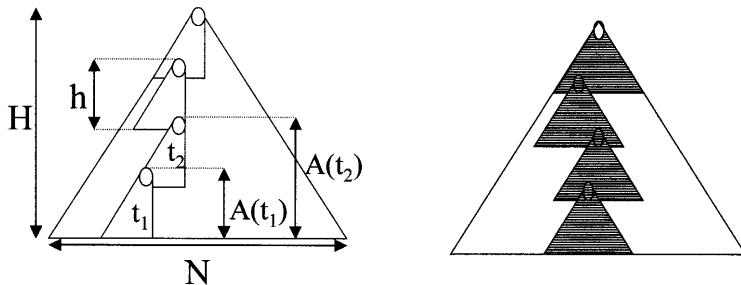


Fig. 2. (Left) The height of the Merkle tree is H , and thus, the number of leaves is $N = 2^H$. The height of each subtree is h . The altitude $A(t_1)$ and $A(t_2)$ of the subtrees t_1 and t_2 is marked. (Right) Instead of storing all tree nodes, we store a smaller set - those within the stacked subtrees. The leaf whose pre-image will be output next is contained in the lowest-most subtree; the entire authentication path is contained in the stacked set of subtrees

Dynamic View. Apart from the above set of *existing* subtrees, which contain the next required authentication path, we will have a set of *desired* subtrees. If the root of the tree $Exist_i$ has index a , according to the ordering of the height- $i \cdot h$ nodes, then $Desire_i$ is defined to be the h -subtree with index $a + 1$ (provided that $a < 2^{H-i \cdot h} - 1$). In case $a = 2^{H-i \cdot h} - 1$, then $Exist_i$ is the last subtree at this level, and there is no corresponding desired subtree. In particular, there is never a desired subtree at level L . The left part of Figure 3 depicts the adjacent existing and desired subtrees.

As the name suggests, we need to compute the pebbles in the desired subtrees. This is accomplished by adapting an application of Algorithm 2 to the root of $Desire_i$. For these purposes, the algorithm *TREEHASH* is altered to save the pebbles needed for $Desire_i$, rather than discarding them, and secondly to terminate one round early, never actually computing the root. Using this variant of *TREEHASH*, we see that each desired subtree being computed has a *tail* of saved intermediate pebbles as described in Section 2. We depict this dynamic computation in the right part of Figure 3, which shows partially completed subtrees and their associated tails.

3.3 Algorithm Intuition

We now can present intuition for our main algorithm, and explain why the existing subtrees $Exist_i$ will always be available.

Overview. The goal of the traversal is to output the leaf pre-images and authentication paths, sequentially. By design, the existing subtrees should always contain the next authentication path to be output, while the desired subtrees contain more and more completed pebbles with each round, until the existing subtree expires.

When $Exist_i$ is used in an output for the last time, we say that it *dies*. At that time, the adjacent subtree, $Desire_i$ will need to have been completed, i.e., have values assigned to all its nodes but its root (since the latter node is

already part of the parent tree.) The tree $Exist_i$ is then *reincarnated* as $Desire_i$: First all the old pebbles of $Exist_i$ are discarded; then the pebbles of $Desire_i$ (and their associated values) taken by $Exist_i$. (Once this occurs, the computation of the new and adjacent subtree $Desire_i$ will be initiated.) This way, if one can ensure that the pebbles on trees $Desire_i$ are always computed on time, one can see that there will always be completed existing subtrees $Exist_i$.

Modifying TREEHASH. As mentioned above, our tool used to compute the desired tree is a modified version of the classic *TREEHASH* in Section 2 applied to the root of $Desire_i$. This version differs in that (1) it stops the algorithm one round earlier (thereby skipping the root calculation), and (2) every pebble of height greater than ih is saved into the tree $Desire_i$. For purposes of counting, we won't consider such saved pebbles as part of the tail “proper”.

Amortizing the Computations. For a particular level i , we recall that the computational cost for tree $Desire_i$ is $2 * 2^{ih} - 2$, as we omit the calculation of the root. At the same time we know that $Exist_i$ will serve for 2^{ih} output rounds. We amortize the computation of $Desire_i$ over this period, by simply computing two iterations of *TREEHASH* each round. In fact, $Desire_i$ will be ready before it is needed, exactly 1 round in advance!

Thus, for each level, allocating 2 computational units ensures that the desired trees are completed on time. The total computation per round is thus $2(L - 1)$.

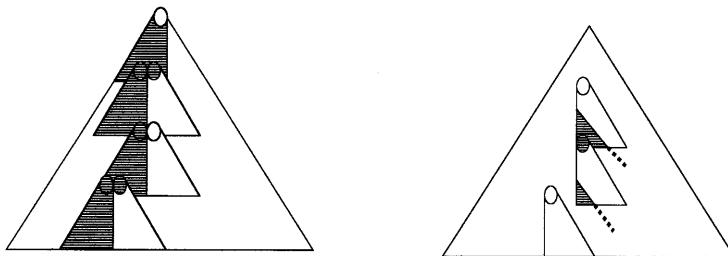


Fig. 3. (Left) The grey subtrees correspond to the *existing* subtrees (as in figure 2) while the white subtrees correspond to the *desired* subtrees. As the existing subtrees are used up, the desired subtrees are gradually constructed. (Right) The figure shows the set of desired subtrees from the previous figure, but with grey portions corresponding to nodes that have been computed and dotted lines corresponding to pebbles in the tail

4 Solution and Algorithm Presentation

Three Phases. We now describe more precisely the main algorithm. There are three phases, the *key generation* phase; the *output* phase; and the *verification* phase. During the key generation phase (which may be performed offline by a relatively powerful computer), the root of the tree is computed and output, taking the role of a public key. Additionally, the iterative output phase needs some setup, namely the computation of pebbles on the initial *existing* subtrees. These are stored on the computer performing the output phase.

The *output* phase consists of a number of rounds. During round j , the (previously unpublished) pre-image the j 'th leaf is output, along with its authentication path. In addition, some number of pebbles are discarded and some number of pebbles are computed, in order to prepare for future outputs.

The *verification* phase is identical to the traditional verification phase for Merkle trees and has been described above. We remark again that the outputs our algorithm generates will be indistinguishable from the outputs generated by a traditional algorithm. Therefore, we do not detail the verification phase, but merely the key generation phase and output phase.

4.1 Key Generation

First, the pebbles of the left-most set of stacked *existing* subtrees are computed and stored. Each associated pebble has a *value*, a *position*, and a *height*. In addition, a list of *desired* subtrees is created, one for each level $i < L$, each initialized with an empty stack for use in the modified *TREEHASH* algorithm.

Recalling the indexing of the leaves, indexed by $leaf \in \{0, 1, \dots, N - 1\}$, we initialize a counter $Desire_i.position$ to be 2^{ih} , indicating which Merkle tree leaf is to be computed next

Algorithm 2: Key-Gen and Setup

1. **Initial Subtrees** For each $i \in \{1, 2, \dots, L\}$:
 - Calculate all (non-root) pebbles in existing subtree at level i .
 - Create new empty desired subtree at each level i (except for $i = L$), with leaf *position* initialized to 2^{ih} .
2. **Public Key** Calculate and publish tree root.

4.2 Output and Update Phase

Each round of the execution phase consists of the following portions: *generating an output*, *death and reincarnation of existing subtrees*, and *growing desired subtrees*.

Generating an Output. At round j , the output consists of the j 'th leaf pre-image, and the authentication path associated to this leaf. The pebbles for this authentication path will be contained in the existing subtrees, and only the pre-image needs to be computed during this round.

Death and Reincarnation of Existing Subtrees. When the last authentication path requiring pebbles from a given existing subtree has been output, then the subtree is no longer useful, and we say that it “*dies*.” By then, the corresponding desired subtree has been completed, and the recently died existing subtree “*reincarnates*” as this completed desired subtree. Notice that a new subtree at level i is needed once every 2^{ih} rounds, and so once per 2^{ih} rounds the pebbles in the existing tree are discarded. More technically, at round j , $j = 1 \pmod{2^{ih}}$ the pebbles in the old tree $Exist_i$ are discarded; the completed tree $Desire_i$ becomes the tree new $Exist_i$; and a new, empty desired subtree is created.

Growing Desired Subtrees. In this step we grow each desired subtree that is not yet completed a little bit. More specifically, we apply two computational units to the new or already started invocations of the *TREEHASH* algorithm. Recall that the counter *position* corresponds to the next leaf to be computed within the *TREEHASH* algorithm, (which is presented with index *leaf* starting from 0). We concisely present this algorithm as follows:

Algorithm 3: Stratified Merkle Tree Traversal

1. Set $leaf = 0$.
2. **Output** Authentication Path for leaf number $leaf$.
3. **Next Subtree** For each i for which $Exist_i$ is no longer needed, i.e, $i \in \{1, 2, \dots, L\} | leaf = 1 \pmod{2^{hi}}$:
 - Remove Pebbles in $Exist_i$.
 - Rename tree $Desire_i$ as tree $Exist_i$.
 - Create new, empty tree $Desire_i$ (if $leaf + 2^{hi} < 2^H$).
4. **Grow Subtrees** For each $i \in \{1, 2, \dots, h\}$: Grow tree $Desire_i$ by applying 2 units to modified *TREEHASH* (unless $Desire_i$ is completed).
5. Increment $leaf$ and loop back to step 2 (while $leaf < 2^H$).

5 Time and Space Analysis

Time. As presented above, our algorithm allocates 2 computational units to each desired subtree. Here, a computational unit is defined to be either a call to *LEAF-CALC*, or the computation of a hash value. Since there exist are at most $L - 1$ desired subtrees, the total computational cost per round is

$$T_{max} = 2(L - 1) < 2H/h. \quad (3)$$

Space. The total amount of space required by our algorithm, or equivalently, the number of available pebbles required, may be bounded by simply counting the contributions from (1) the existing subtrees, (2) the desired subtrees, and (3) the tails.

First, there are L existing subtrees and up to $L - 1$ desired subtrees, and each of these contains up to $2^{h+1} - 2$ pebbles, since we do not store the roots. Additionally, by equation 2, the tail associated to a desired subtree at level $i > 1$ contains at most $h * i + 1$ pebbles. If we count only the pebbles in the tail which do not belong to the desired subtree, then this “proper” tail contains at most $h(i - 1) + 1$ pebbles. Adding these contributions, we obtain the sum $(2L - 1)(2^{h+1} - 2) + h \sum_{i=1}^{L-2} i + 1$, and thus the bound:

$$Space_{max} \leq (2L - 1)(2^{h+1} - 2) + L - 2 + h(L - 2)(L - 1)/2. \quad (4)$$

A marginally worse bound is simpler to write:

$$Space_{max} < 2L2^{h+1} + HL/2. \quad (5)$$

Trade-Offs. The solution just analyzed presents us with a trade-off between time and space. In general, the larger the subtrees are, the faster the algorithm will run, but the larger the space requirement will be. The parameter affecting the space and time in this trade off is h ; in terms of h the computational cost is below $2H/h$, the space required is bounded above by $2L2^{h+1} + HL/2$. Alternatively, and in terms of h , the space is bounded above by $2H2^{h+1}/h + H^2/2h$.

Low Space Solution. If one is interested in parameters requiring little space, there is an optimal h , due to the fact that for very small h , the number of tail pebbles increases significantly (when $H^2/2h$ becomes large). An approximation of this value is $h = \log H$. One could find the exact value by differentiating the expression for the space: $2H2^{h+1}/h + H^2/2h$. For this choice of $h = \log H = \log \log N$, we obtain

$$T_{max} = 2 \log N / \log \log N. \quad (6)$$

$$Space_{max} \leq 5/2 \log^2 N / \log \log N. \quad (7)$$

These results are interesting in that they it improves Merkle’s result - for both space and time (for large enough N). Merkle’s tree traversal had $T_{max} = 2 \log N$ and $Space_{max} = 0 \bmod(1/2 \log^2 N)$.

We now continue with a small technical modification, which will improve the constants in these bounds, and yield the result presented in the introduction.

6 Additional Savings

Although this modification does not affect the complexity *class* of either the space or time costs, it is of practical interest as it nearly halves the space bound in certain cases. It is presented after the main exposition in order to retain the original simplicity, as this analysis is slightly more technical. The modification is based on two observations: (1) There may be pebbles in existing subtrees which are no longer useful, and (2) The desired subtrees are always in a state of partial completion. In fact, we have found that pebbles in the an existing subtree may be discarded *nearly* as fast as pebbles are entered into the corresponding desired subtree. The modifications are as follows:

1. Discard pebbles in the trees $Exist_i$ as soon as they will never again be required.
2. Omit the *first* application of 2 units to modified *TREEHASH* algorithm.

We note that with the second modification, the desired subtrees still complete, just in time. With these small changes, for all levels $i < L$, the number of pebbles contained in *both* $Exist_i$, and $Desire_i$ can be bounded by the following expression.

$$Space_{Exist(i)} + Space_{Desire(i)} \leq 2^{ih+1} - 2 + (h - 2). \quad (8)$$

This is nearly half of the previous bound of $2 * (2^{ih+1} - 2)$. We relegate the technical proof of this to the appendix, but remark the quantity $h - 2$ measures the maximum number of pebbles in $Desire_i$ over the number of pebbles in $Exist_i$ which have been discarded. Using the estimate 8, we revise the space bound computed in the previous section as

$$Space_{max} \leq (L)(2^{h+1} - 2) + (L - 1)(h - 2) + L - 2 + h(L - 2)(L - 1)/2 \quad (9)$$

We again round this up to obtain a simpler bound.

$$Space_{max} < L 2^{h+1} H L / 2. \quad (10)$$

Specializing to the choice $h = \log\log N$, we improve the above result to

$$Space_{max} \leq 3/2 \log^2 N / \log\log N. \quad (11)$$

by reducing the constant from $5/2$ to $3/2$.

7 Future Work

While general hash functions allow for arbitrary input sizes, Merkle trees use very particular input sizes. In particular, the one-way function we need for interior nodes is two-to-one; and that used to compute leaf values from their respective pre-images is one-to-one. If special-purpose functions are developed for these purposes, this may improve the efficiency of the resulting algorithm, most notably by avoiding the overhead associated with the initial padding of short inputs, as is performed by standard hash functions. Moreover, if these special-purpose functions are derived from particular primitives, such as AES, it would be possible to prove the security of the resulting Merkle-based scheme on the assumed hardness of the primitive.

Acknowledgments

We wish to thank Ari Juels for providing helpful feedback towards making the paper easier to read.

References

- [1] D. Coppersmith and M. Jakobsson, "Almost Optimal Hash Sequence Traversal," Financial Crypto '02. Available at www.markus-jakobsson.com. 315
- [2] Y.-C. Hu, A. Perrig, and D.B. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks," Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003), IEEE, San Francisco, CA, April 2003, to appear 315
- [3] M. Jakobsson, "Fractal Hash Sequence Representation and Traversal," ISIT '02, p. 437. Available at www.markus-jakobsson.com. 315
- [4] C. Jutla and M. Yung, "PayTree: amortized-signature for flexible micropayments," 2nd USENIX Workshop on Electronic Commerce, pp. 213–221, 1996. 315
- [5] L. Lamport, "Constructing Digital Signatures from a One Way Function," SRI International Technical Report CSL-98 (October 1979). 315
- [6] H. Lipmaa, "On Optimal Hash Tree Traversal for Interval Time-Stamping," In Proceedings of Information Security Conference 2002, volume 2433 of Lecture Notes in Computer Science, pp. 357–371. Available at www.tcs.hut.fi/~helger/papers/lip02a/ 317
- [7] R. Merkle, "Secrecy, Authentication, and Public Key Systems," UMI Research Press, 1982. Also appears as a Stanford Ph.D. thesis in 1979. 315
- [8] R. Merkle, "A digital signature based on a conventional encryption function," Proceedings of Crypto '87, pp. 369–378. 314, 315, 317
- [9] S. Micali, "Efficient Certificate Revocation," Proceedings of RSA '97, and U.S. Patent No. 5,666,416. 315
- [10] A. Perrig, R. Canetti, D. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol," Cryptobytes, Volume 5, No. 2 (RSA Laboratories, Summer/Fall 2002), pp. 2–13. Available at www.rsasecurity.com/rsalabs/cryptobytes/ 315
- [11] K.S.J. Pister, J. M. Kahn and B.E. Boser, "Smart Dust: Wireless Networks of Millimeter-Scale Sensor Nodes. Highlight Article in 1999 Electronics Research Laboratory Research Summary.", 1999. Available at robotics.eecs.berkeley.edu/~pister/SmartDust/ 314
- [12] R. Rivest and A. Shamir, "PayWord and MicroMint—Two Simple Micropayment Schemes," CryptoBytes, volume 2, number 1 (RSA Laboratories, Spring 1996), pp. 7–11. Available at www.rsasecurity.com/rsalabs/cryptobytes/ 315
- [13] FIPS PUB 180-1, "Secure Hash Standard, SHA-1". Available at www.itl.nist.gov/fipspubs/fip180-1.htm 316
- [14] Y. Sella, "Traversing Hash Chain with Constant Computation," To appear in Financial Crypto '03. 315
- [15] S. Vaudenay, "One-time identification with low memory," EUROCODE'92, CISM Course and Lecture 339, pp. 217–228, Springer-Verlag 1993 315

A Proof of Space Bound

Here we prove assertion 6 which states for any level i the number of pebbles in the $Exist_i$ plus the number of pebbles in the $Desire_i$ is less than $\alpha 2 * 2^h - 2 + (h - 2)$. This basic observation is that desired subtree can grow only slightly faster than the existing subtree shrinks. To simplify the indices, we assume (without loss of generality) that we are considering the first existing-desired subtree pair at some level i .

Returned Pebbles. The first modification ensures that pebbles are returned more continuously than previously, so we quantify this. Subtree $Exist_i$, has 2^h leaves, and as each leaf is no longer required, neither may be some interior nodes above it. These leaves are finished at rounds $2^{(i-1)h}a - 1$ for $a \in \{1, \dots, 2^h\}$. We may determine the number of pebbles returned at these times by observing that a leaf is always returned, a pebble at height $i h + 1$ every two rounds, one at height $i h + 2$ every four rounds, etc. We are interested in the number returned at all times up to the time $2^{(i-1)h}a - 1$; this is the sum of the greatest integer functions:

$$A + [A/2] + [A/4] + [A/8] + \dots + [A/2^h]$$

Writing a in binary notation $a = a_0 + 2^1a_1 + 2^2a_2 + \dots + 2^ha_h$, this sum is also

$$a_0(2^1 - 1) + a_1 * (2^2 - 1) + a_2 * (2^3 - 1) + \dots + a_h(2^{h+1} - 1).$$

New Pebbles. The cost to calculate the corresponding pebbles in $Desire_i$ may also be calculated with a similar expression. Using the fact that a height h_0 node needs $2^{h_0+1} - 1$ units to compute, we see that the desired subtree requires

$$a_0(2^{(i-1)h+1} - 1) + a_1(2 * 2^{(i-1)h+2} - 1) + \dots + a_h(2 * 2^{ih+1} - 1).$$

computational units to place those same pebbles. This cost is equal to $2 * 2^{i-1}h * a - z$, where z denotes the number of nonzero digits in the binary expansion of a .

Difference. At time $2^{(i-1)h}a - 1$, a total of $2 * 2^{(i-1)h}a - 2$ units of computation has been applied to $Desire_i$, (factoring in our 1 round delay). Noting that $2^{(i-1)h} - 1$ more rounds may pass before $Exist_i$ loses any more pebbles, we see that the maximal number of pebbles during this interval must be realized at the upper end. At this point in time, the desired subtree has computed exactly the pebbles that have been removed from the existing tree, plus whatever additional pebbles it can compute with its remaining $2 * 2^{ih} - 2 + z - 2$ computational units. The next pebble, (a leaf) costs $2 * 2^{ih} - 1$ which leaves $z - 3$. Even if all of these units result in new pebbles, the total extra is still less than or equal to $1 + z - 3$. Since $z \leq h$, this extra pebbles is bounded by $h - 2$, as claimed, and Equation 8 is proved.

RSA Shortcuts

Adi Shamir

Applied Math Department, The Weizmann Institute of Science
Rehovot 76100, Israel
shamir@wisdom.weizmann.ac.il

Abstract. In this talk I'll survey a variety of unpublished enhancements, optimizations, implementation ideas and new variants of the RSA scheme which I have found over the years.

The Width- w NAF Method Provides Small Memory and Fast Elliptic Scalar Multiplications Secure against Side Channel Attacks

Katsuyuki Okeya¹ and Tsuyoshi Takagi²

¹ Hitachi, Ltd., Systems Development Laboratory
292, Yoshida-cho, Totsuka-ku, Yokohama, 244-0817, Japan

ka-okeya@sdl.hitachi.co.jp

² Technical Universität Darmstadt
Fachbereich Informatik
Alexanderstr.10, D-64283 Darmstadt, Germany
takagi@cdc.informatik.tu-darmstadt.de

Abstract. The side channel attack (SCA) is a serious attack on wearable devices that have scarce computational resources. Cryptographic algorithms on them should be efficient using small memory — we have to make efforts to optimize the trade-off between efficiency and memory. In this paper we present efficient SCA-resistant scalar multiplications based on window method. Möller proposed an SPA-resistant window method based on 2^w -ary window method, which replaces w -consecutive zeros to 1 plus w -consecutive 1 and it requires 2^w points of table (or $2^{w-1} + 1$ points if the signed 2^w -ary is used). The most efficient window method with small memory is the width- w NAF, which requires 2^{w-2} points of table. In this paper we convert the width- w NAF to an SPA-resistant addition chain. Indeed we generate a scalar sequence with the fixed pattern, e.g. $|0..0x|0..0x|...|0..0x|$, where x is positive odd points $< 2^w$. Thus the size of the table is 2^{w-1} , which is optimal in the construction of the SPA-resistant chain based on width- w NAF. The table sizes of the proposed scheme are 6% to 50% smaller than those of Möller’s scheme for $w = 2, 3, 4, 5$, which are relevant choices in the sense of efficiency for 160-bit ECC.

Keywords: Elliptic curve cryptosystem, side channel attacks, width- w NAF, pre-computation table, smart card, memory constraint.

1 Introduction

Elliptic curve cryptosystem (ECC) has a shorter key size comparing with RSA cryptosystem and thus it is suitable for implementing on wearable devices like smartcards. However, if the implementation of ECC is careless, an attacker can break the secret key by observing side channel information, namely timing of operations, power consumptions, etc [Koc96, KJJ99]. This attack is called the *side channel attack* (SCA). We have to design the implementation of cryptographic algorithms by rendering careful attention to the SCA.

In order to realize ECC on wearable devices with less computational resource, a lot of methods to improve the efficiency of ECC have been proposed. If we are allowed to use extra memory, window based methods can enhance the speed of ECC. However we have made efforts to optimize the trade-off between efficiency and memory. One of the optimized window method is the width- w NAF [Sol00]. The width- w NAF chain has at most 1 non-zero bit among any w -consecutive bits, and thus it provides faster scalar multiplication. The width- w NAF uses the table with 2^{w-2} points, which is quite small comparing with other window based addition chain, e.g. the sliding window method has 2^{w-1} pre-computed points. The width- w NAF is one of the fastest window based method with small memory.

Several algorithms that resist the SCA have been proposed, e.g., [Cor99, OS00, BJ02, FGKS02, IT02]. In this paper we deal with a countermeasure according to modifying addition chains based on windows methods. There are two different types of countermeasure using the windows method. (1) The first type tries to resist SCA by randomizing the addition chain of the secret scalar [IYTT02, LS01, OA01], which includes randomized window method proposed by Liardet-Smart [LS01], an enhancement proposed by Itoh *et al.* [IYTT02], and the randomized addition-subtraction chains method proposed by Oswald-Aigner [OA01]. However, the security of these schemes is controversial. Indeed, Okeya and Sakurai proposed an SPA against Oswald-Aigner scheme [OS02a]. Walter proved that the Liardet-Smart method is also vulnerable to a certain attack using Markov model [Wal02b]. (2) The second one is to utilize the fixed windows method but with a pre-computed table. Möller [Möll01a, Möll01b] has proposed a fixed window-method based countermeasure. His scheme is based on the 2^w -array method, and thus it requires $2^w + 1$ points (or $2^{w-1} + 1$ for signed case). Whereas the countermeasure provides good efficiency on speed, it compromises on memory consumption.

1.1 Contribution of this Paper

In this paper we propose an SCA-resistant scheme based on the width- w NAF. The proposed scheme constructs an addition chain of a fixed pattern, e.g. $|0..0x| 0..0x|...|0..0x|$, where x is chosen from the pre-computed table. The fixed digit has the $w-1$ consecutive bits and a non-zero bit x . The fixed pattern is converted from the width- w NAF chain. We optimized the size of the pre-computed table, namely we only store the odd points $< 2^w$. The proposed method offers smaller memory, i.e., 2^{w-1} , which is smaller than that of Möller's method. Indeed the table sizes of the proposed scheme are 6% to 50% smaller than those of Möller's scheme for $w = 2, 3, 4, 5$, which are relevant cases in the sense of efficiency for 160-bit ECC. We also propose an SCA-resistant scheme using the dummy operation based on the width- w NAF, which can be considered as an extension of Coron's dummy operation [Cor99]. The construction of the proposed scheme is quite simple comparing with the scheme proposed by Hitchcock *et al.*, which is constructed according to inserting dummy operation to NAF chain [HM02]. The dummy operation used in our construction is only for the zero digit $|0..00|$ — we replace the digit to $|0..0x|$ for non-zero x , so that the length of the proposed

chain remains as large as original. Therefore the proposed scheme is faster than that of the scheme of Hitchcock *et al.*

The paper is organized as follows: In Section 3 we present a survey of side channel attacks and their countermeasures. In Section 2 we shortly review ECC and describe the width- w NAF method of elliptic scalar multiplication. In Section 4 we describe our proposed countermeasure against side channel attacks. In Section 5 we conclude this paper.

2 Elliptic Curve Cryptosystem

Elliptic curve cryptosystem (ECC) is constructed using cryptographically suitable elliptic curves over finite fields \mathbf{F}_q . In this paper we consider finite fields whose characteristic are larger than 3. Elliptic curves over \mathbf{F}_q are defined by

$$E(a, b) = \{(x, y) \in \mathbf{F}_q \times \mathbf{F}_q | y^2 = x^3 + a x + b\} \cup \mathcal{O},$$

where $a, b \in \mathbf{F}_q$ such that $4a^3 + 27b^2 \neq 0$, and \mathcal{O} is the point of infinity. The elliptic curve $E(a, b)$ possesses a group structure together with \mathcal{O} as the identity element, which is assembled by the arithmetic of the definition field \mathbf{F}_q . The formulae of computing the group structure consist of the elliptic addition (ECADD) that computes $P + Q$ for the points $P, Q \in E(a, b)$ with $P \neq \pm Q$, and the elliptic doubling (ECDBL) that computes $2P$ for $P \in E(a, b)$.

The computation of ECADD/ECDBL has an inversion of the definition field. In order to avoid it, we usually deploy the Jacobian coordinate, which represent a point $P = (X : Y : Z)$, where $x = X/Z^2$ and $y = Y/Z^3$ [CMO98]. Denote by M , S , and I the computation time of the multiplication, the squaring, and the inversion in the definition field, respectively. Recovering the point in the Jacobian coordinate to the affine coordinate requires $3M + 1S + 1I$. The computation time of $\text{ECADD}^{\mathcal{J}}$ and $\text{ECDBL}^{\mathcal{J}}$ in the Jacobian coordinate are $12M + 4S$ and $4M + 6S$, respectively. In order to enhance the speed of the repeatedly used ECDBL, we employ the elliptic curve doubling in the modified Jacobian coordinate $\text{ECDBL}^{\mathcal{J}^m}$, which requires $4M + 4S$. The $\text{ECADD}^{\mathcal{J}^m + \mathcal{A} \rightarrow \mathcal{J}^m}$ requires $9M + 5S$. If we apply the $\text{ECDBL}^{\mathcal{J}^m \rightarrow \mathcal{J}}$ before the ECADD, we can save $1M$, namely $3M + 4S$. In this case we use $\text{ECADD}^{\mathcal{J} + \mathcal{A} \rightarrow \mathcal{J}^m}$, so that we can return the modified Jacobian coordinate, which requires $9M + 5S$ and it is the same computation time as $\text{ECADD}^{\mathcal{J}^m + \mathcal{A} \rightarrow \mathcal{J}^m}$. The comprehensive estimation of the computation time of other coordinates can be found in [CMO98].

2.1 Scalar Multiplication

The scalar multiplication of ECC is the most dominant computation part of ECC, which computes dP for a given point P and a scalar d . There are many algorithms of computing the scalar multiplication. The standard one is the non-adjacent form (NAF) [IEEE]. The NAF represents the n -bit scalar d by the signed binary chain, namely $d = \sum_{i=0}^n d[i]2^i$, where $d[i] = -1, 0, 1$. Then we compute the scalar multiplication as follows:

Algorithm Scalar Multiplication with NAF**INPUT** NAF $d[i]$, point P **OUTPUT** dP

1. $Q \leftarrow P$
2. For $i = n - 1$ down to 0
 - 2.1. $Q \leftarrow \text{ECDBL}(Q)$
 - 2.2. if $d[i] = \pm 1$ then $Q \leftarrow \text{ECADD}(Q, \pm P)$
3. Return Q

The inversion of a point $P = (x, y)$ is represented by $-P = (x, -y)$ that is negligible time comparing with the multiplication of the definition field. It is known that the density of non-zero bits of NAF is asymptotically $1/3$. Thus the scalar multiplication using the NAF requires n ECDBL + $(n/3)$ ECADD. For 160-bit scalar we need $1816M$, assuming $S = 0.8M$ and $I = 20M$.

2.2 Width- w NAF Method

In this section we review width- w NAF, which is an extension of the NAF using pre-computed points. The width- w NAF presents an n -bit integer d as $d = \sum_{i=0}^{n-1} d_w[i]2^i$, where $d_w[i]$ are odd integers with $|d_w[i]| < 2^{w-1}$ and there are at most one non-negative digits among w -consecutive digits.

The width- w NAF has been independently proposed by different authors [KT92], [MOC97], [BSS99]. The generation algorithm proposed by Solinas is very simple, so that it understandably illustrates the feature of the width- w NAF [Sol00]. We describe the width- w NAF by Solinas in the following.

Algorithm Width- w NAF**INPUT** An n -bit scalar value d .**OUTPUT** $d_w[n], d_w[n-1], \dots, d_w[0]$.

1. For $i = 0$ to n
 - 1.1. if $d = 1 \bmod 2$ then $d_w[i] \leftarrow d \bmod 2^w$ and $d \leftarrow d - d_w[i]$
 - 1.2. else $d_w[i] \leftarrow 0$
 - 1.3. $d \leftarrow d/2$
2. Return $d_w[n], d_w[n-1], \dots, d_w[0]$

The “ $\bmod 2^w$ ” in Step 1.2 chooses the signed residue class modulo 2^w for odd integers d , namely $\{-2^{w-1} + 1, \dots, -3, -1, 1, 3, \dots, 2^{w-1} - 1\}$. Thus, we have to pre-compute the points $P, 3P, \dots, (2^{w-1} - 1)P$ in order to represent the residue class by the pre-computed points, which has 2^{w-2} points. Since $d_w[i]$ is odd with $|d_w[i]| < 2^{w-1}$ if d in Step 1.1 is not even, the d after Step 1.1 is always divisible by 2^w . Hence, once $d_w[i] \neq 0$ holds, the next $w - 1$ consecutive bits are always zero; $d_w[i+1] = d_w[i+2] = \dots = d_w[i+w-1] = 0$. Note that the width-2 NAF is equivalent to the standard NAF. It is known that the density of the non-zero bits of the width- w NAF is asymptotically $1/(1+w)$. Then we compute the scalar multiplication using the width- w NAF as follows:

Algorithm Scalar Multiplication with Width- w NAF**INPUT** Width- w NAF $d_w[i]$, point P , precomputed points $(|d_w[i]|)P$ **OUTPUT** dP

1. $Q \leftarrow d_w[c]P$ for the largest c such that $d_w[c] \neq 0$
2. For $i = c - 1$ to 0
 - 2.1. $Q \leftarrow \text{ECDBL}(Q)$
 - 2.2. if $d_w[i] \neq 0$ then $Q \leftarrow \text{ECADD}(Q, d_w[i]P)$
3. Return Q

3 Side Channel Attacks and Their Countermeasures

In cryptographic devices such as smart cards, data other than input data and output data may “leak out” during cryptographic procedures. The computation time of cryptographic procedures is one such kind of data. So is power consumption because the smart card uses an external power source. Kocher *et al.* developed the side channel attack in which an attacker infers stored secret information in a cryptographic device by using such leaked data [Koc96, KJJ99]. This type of attack, which include timing attack, Simple Power Analysis (SPA), and Differential Power Analysis (DPA) attack, render smart cards particularly vulnerable.

A timing attack [Koc96] is a side channel attack in which an attacker infers the secret information by using computation time as leaked data. Some methods of timing attack use to statistical analysis to reveal the secret information, others infer it from a one-time computation. An SPA attack [Koc96] is a side channel attack in which an attacker infers the secret information by using power consumption as leaked data. SPA attack reveals the secret information by direct observation of a device’s power consumption without the need for statistical analysis. A DPA attack [KJJ99] is a side channel attack in which an attacker infers the secret information by using statistical analysis of power consumption.

Kocher *et al.* envisioned mainly DES [DES] and RSA [RSA78] as targets for side channel attacks. Coron generalized DPA attack to include elliptic curve cryptosystems [Cor99].

3.1 Countermeasures against Side Channel Attacks

Many countermeasures against side channel attacks have proposed. They are classified into several types: fixed procedure type, randomized addition chains type, indistinguishable operations type, data randomization type, and so on.

The fixed procedure type computes scalar multiplication using a predetermined fixed procedure of operations, which helps to prevent against timing attacks and SPA attacks. This type of countermeasures includes Coron’s dummy method [Cor99] and Montgomery ladder methods [OS00, BJ02, FGKS02, IT02]. Coron’s dummy method always computes an addition and a doubling per bit of the secret scalar, and decides to use the results depending on the bit. Montgomery ladder methods are similar to Coron’s dummy method but do not use

dummy operations. Montgomery ladder methods always compute an addition and a doubling per bit of the secret scalar, and hold two points during the computation. Which point to be doubled depends on the bit.

The randomized addition chains type computes scalar multiplication using randomized addition chains, that is, each execution uses a different addition chain, which helps to prevent against timing, SPA and DPA attacks. This type of countermeasures includes randomized exponent methods [Cor99], randomized window method [LS01], and randomized addition-subtraction chains method [OA01]. The randomized exponent methods exchange the secret scalar d for $d + \phi r$, where ϕ is the order of the basepoint, and r is a random number. The randomized window method uses a randomly chosen window width. The randomized addition-subtraction chains methods inserts random decisions of operations into the addition-subtraction chains method. However, the security of these schemes is controversial, because the number of all the possible addition chains is small comparing with that of all the secret scalars, and random numbers used early in the computation does not affect the late computation. Indeed Okeya and Sakurai has shown that the randomized addition-subtraction chains method is vulnerable to SPA under distinguishability between addition and doubling [OS02a]. Walter proved the randomized window method is vulnerable to a certain SCA using Markov model [Wal02b].

The indistinguishable operations type blinds the attacker to the distinguishability of addition and doubling using same addition formulae, which helps to prevent against SPA attacks. This type of countermeasures includes indistinguishable addition formulae [JQ01, LS01, BJ02], which are constructed on several forms of elliptic curves.

The data randomization type randomizes computing objects, which helps to prevent against DPA attacks. This type can be combined with SPA countermeasures such as the fixed procedure type, which prevents against both SPA and DPA. This type of countermeasures includes randomized projective coordinates method [Cor99, OMS01], random isomorphic curves method [JT01b, IT02], and blinding point method [Cor99]. The randomized Jacobian coordinates method, a variant of the randomized projective coordinates method, transforms the base-point $(x, y, 1)$ of projective coordinates into (r^2x, r^3y, r) using a random number r . Note that the relation $(x, y, 1) = (r^2x, r^3y, r)$ holds for non-zero r in the Jacobian coordinates because of their properties. The blinding point method generates a random point R , and computes $d(P + R) - dR$ instead of dP , where P is the basepoint and d is the secret scalar.

ECDSA scheme [ANSI] is one of the most important applications of elliptic scalar multiplication. However, the verification of ECDSA does not use any secret information. In other words, we do not need to consider SCA against the verification. On the other hand, the signature generation of ECDSA uses a one-time key pair generation, which is a scalar multiplication of a randomly chosen secret scalar. Thus, the attacker does not use the technique of averaging, that is, DPA is not available. Therefore, preventing against SPA is essential in the elliptic curve cryptosystems. We should note that the immunity against DPA is

easily obtained by combining with countermeasures of the data randomization type. Thus, we mainly consider the countermeasure against SPA in the next section.

4 Proposed Countermeasure

In this section we explain the proposed algorithm. We modify the width- w NAF to be secure against the SPA.

We aim at generating a scalar sequence that has a fixed pattern, e.g. $|0\dots0x|0\dots0x|\dots|0\dots0x|$, where x is chosen from a pre-computed table. If $d = 1 \bmod 2$ holds in step 1.1 of Algorithm Width- w NAF, we assign $u_w[i] = d \bmod 2^w$ and obtain the pattern of w -consecutive bits $|0\dots0x|$. If $u_w[i+w]$ is non-zero, we obtain the same bit pattern $|0\dots0x|$ after $u[i+w-1]$. However, if $u_w[i+w] = 0$ holds, we obtain the sequence $0|0\dots0x|$, which is no longer the fixed pattern. This exceptional zeros are called *zero runs*. We try to eliminate the zero runs by additional pre-computing several points. Firstly we show the basic algorithm of the proposed scheme in the following.

Algorithm SPA-resistant Width- w NAF

INPUT An n -bit scalar value d .

OUTPUT $d_w[n], d_w[n-1], \dots, d_w[0]$.

1. For $i = 0$ to $\lceil n/w \rceil$
 - 1.1. $u[i] \leftarrow d \bmod 2^w$
 - 1.2. if $u[i] = 0\dots0$ then $u[i] \leftarrow \bar{1}\dots\bar{1}$ and $d \leftarrow d + 2^w$
 - 1.3. $d \leftarrow d - u[i]$, $d \leftarrow d/2^w$
 - 1.4. $d_w[iw] = u[i], d_w[iw+1] = \dots = d_w[iw+w-1] = 0$
2. Return $d_w[n], d_w[n-1], \dots, d_w[0]$

Here the “mode 2^w ” in Step 1.2 is a special residue class E that satisfies the following conditions:

The set $\{P, -P | P \in E\}$ represents all different points that appear in the pre-computed points.

The class E is called as the *representative class*. For example, we can choose $P, 2P, 4P, 5P$ as the representative class for $w = 3$. The scheme proposed by Möller chooses the set $E_M = \{P, 2P, 3P, \dots, (2^{w-1})P, 2^wP\}$ for the representative class. It have $2^{w-1} + 1$ points. In the following section we will optimize the choice of the representative class E by eliminating even points.

In order to achieve the fixed pattern $|0\dots0x|0\dots0x|\dots|0\dots0x|$, we have to care the w consecutive zeros ($x = 0$), which occurs in the case of $u[i] = 0\dots0$. It can be replaced to $\bar{1}\dots\bar{1}$ due to $1|\bar{1}\dots\bar{1}$. Then we have to add 1 to d . These treatments are processed in Step 1.2. By the operation $d = d + 1$ the additional pre-computed points are required. We discuss how to deal with this problem in the next section.

4.1 Proposed Algorithm

We describe our proposed method in the following. The digits $|0..0x|$ with even (or odd) x are called even (or odd) digits, respectively. For $w = 4$ the odd digits are 0001, 0011, 0101, 0111, 1001, 1011, 1101, and 1111. We propose an algorithm that replaces all even digits to odd digits by considering the carries to neighbour digits. We replace the even digits to odd digits by looking at the most bit of the lower digit. Let us consider the even digit $|0100|$. If we have the sequence $|0100|01**|$ whose previous digit has 0 most bit, it can be replaced by $|0101|\bar{1}1**|$. Similarly the sequence $|0100|\bar{0}1 * *|$ can be replaced by $|0101|11 * *|$. If the most bit of the previous digit is 1, then we can convert $|0100|1 * * *|$ to $|0101|\bar{1} * * *|$. Thus we can always convert even digit $|0100|$ to the odd digit. By this conversion the current digit $u[i]$ and the lower digit $u[i - 1]$ are replaced to

$$u[i] = u[i] + b \text{ and } u[i - 1] = u[i - 1] - b2^w, \quad (1)$$

where $b = \text{sign}(u[i - 1])$ is the sign of the lower digit, namely $b = 1$ if $u[i - 1] > 0$ and $b = -1$ if $u[i - 1] < 0$. Note that if the lower digit is not $|0000|$, the carry arisen from the even digit is disappeared in the lower digit — the next lower digits are never changed by these treatments. Similarly we can convert all even digits except $|0000|$ to the odd digit. We show the conversions as follows:

$$\begin{array}{ll} |0010|* \rightarrow |0011|* \text{ or } |001\bar{1}|*, & |1000|* \rightarrow |1001|* \text{ or } |100\bar{1}|* \\ |0100|* \rightarrow |0101|* \text{ or } |010\bar{1}|*, & |1010|* \rightarrow |1011|* \text{ or } |101\bar{1}|* \\ |0110|* \rightarrow |0111|* \text{ or } |011\bar{1}|*, & |1100|* \rightarrow |1101|* \text{ or } |110\bar{1}|* \\ & |1110|* \rightarrow |1111|* \text{ or } |111\bar{1}|*. \end{array}$$

We explain how to deal with the lower digit $|0000|$. The zero runs $|***0|0000|01*|$ can be replaced by $|***1|\bar{1}\bar{1}\bar{1}\bar{1}|\bar{1}1**|$ or $|***\bar{1}|1111|1***|$. Then both of the higher and lower digits become odd. This conversion can be applied to the lower non-zero digits (e.g. $|***0|0000|0001|$) and longer zero runs $|***0|0000|0000|...|0000|01**|$. This conversion is achieved by the same algorithm of equation (1).

Consequently we can obtain the sequence which has only odd digits except the lowest digit. In the next section we explain how to treat the even lowest digit. We describe the proposed scheme as follows:

Algorithm SPA-resistant Width- w NAF with Odd Scalar

INPUT An odd n -bit scalar value d and $k = \lceil n/w \rceil$

OUTPUT $d_w[n], d_w[n - 1], \dots, d_w[0]$.

1. $u[0] \leftarrow d \bmod 2^w$
2. $d \leftarrow d - u[0]$
3. $d \leftarrow d/2^w$
4. For $i = 1$ to k
 - 4.1. $u[i] \leftarrow d \bmod 2^w$
 - 4.2. if $u[i]$ is even,
 $b \leftarrow \text{sign}(u[i - 1]), u[i] \leftarrow u[i] + b, u[i - 1] \leftarrow u[i - 1] - b2^w$
 - 4.3. $d_w[(i-1)w] \leftarrow u[i-1], d_w[(i-1)w+1] \leftarrow 0, \dots, d_w[(i-1)w+w-1] \leftarrow 0$
 - 4.4. $d \leftarrow d - u[i], d \leftarrow d/2^w$
5. $d_w[kw] \leftarrow u[k], d_w[kw+1] \leftarrow 0, \dots, d_w[kw+w-1] \leftarrow 0$
6. Return $d_w[n], d_w[n - 1], \dots, d_w[0]$

4.2 Treatments of LSB and MSB

The proposed scheme converts odd integers to SPA-resistant chains. In this section, we discuss how even integers treated. We also consider the treatment of the most significant bits.

Least Significant Bit (LSB): The proposed method replaces even digits to odd digits using lower digits. To put it another way, the lowest digit remains as it is. If the lowest digit is odd, we can use the pre-computed table, which stores all the odd points. If not, we might compromise the memory consumption using even points or a dummy operation, however, this is not the case.

In order to avoid performing addition of even points and the dummy operation, we use the following trick: If the scalar d is even, the scalar is converted to $d' = d + 1$. If not, it is converted to $d' = d + 2$. This procedure provides the odd scalar d' . Then, the scalar multiplication $d'P$ of a given point P is computed using the proposed method. The scalar multiplication dP is recovered performing subtraction $(d'P - P)$ or $(d'P - 2P)$.

Since the pre-computed table does not store even points, the final subtraction requires to compute $2P$. The use of the point $2P$ might increase the memory consumption, however, this is not the case. At the time when we compute $d'P$, the pre-computed table is needless. Thus, we discard the table, and the memory is reusable. This is the reason why the memory consumption does not increase.

Most Significant Bit (MSB): Compared with the bit length of a certain scalar, that of the converted scalar using the width- w NAF method is longer than by one bit. For example, $25 = (11001)_2$ is converted into $10\bar{1}001$ using the width-2 NAF (*i.e.* the standard NAF), which is longer by one bit. In contrast, the NAF representation of $19 = (10011)_2$ is $1010\bar{1}$, which has the same length to the original. Hence, upper bits are detectable by confirming whether the bit length is extended or not.

On the contrary, both 25 and 19 have the same width-2 NAF digits, which is 3. This is because in both cases, the bit length is 5, they are partitioned by 2 bits, and 5 and 2 is coprime. For example, the width-3 NAF method does not extend the bit length of any 160-bit scalar. In contrast, the width-4 NAF method extends the bit length of a certain 160-bit scalar.

The use of an NAF width which is coprime to the bit length of scalars fixes the digit length. In particular, the prime bit length of scalars has never extended their digit lengths.

4.3 Memory Consumption

We estimate the memory consumption of the proposed scheme.

Every digit of the proposed algorithm is always odd. Thus we can choose the following pre-computed table:

$$E_{odd} = \{P, 3P, 5P, \dots, (2^w - 1)P\}.$$

The density of the table $\#E_{odd}$ is 2^{w-1} , which is smaller than that of Möller's scheme $\#E_M = 2^{w-1} + 1$. We will show that E_{odd} has the smallest density in all the representative classes.

Proposition 1. *If E is a representative class, then $\#E \geq 2^{w-1}$ holds.*

Proof. Suppose that $(u[i])_{i \in [0, k]}$ and $(u'[i])_{i \in [0, k]}$ are associated with scalars d and d' , respectively. In other words, $d = \sum_{i=0}^k u[i]2^{wi}$ (resp. $d' = \sum_{i=0}^k u'[i]2^{wi}$) such that $u[i]$ or $-u[i]$ (resp. $u'[i]$ or $-u'[i]$) belongs to E . If $u[i] = u'[i]$ holds for all $i \in [0, k]$, then $d = d'$ holds. This implies that the number of all the width- w NAF representations $(u[i])_{i \in [0, k]}$ is larger than that of all the scalars d , because there exists at least one width- w NAF representations $(u[i])_{i \in [0, k]}$ for any scalar d . On the other hand, the number of all the width- w NAF representations is $(2 \cdot \#E)^{k+1}$, and that of all the scalars is 2^n . Since $k+1 = n/w$, we have $\#E \geq 2^{w-1}$. \square

Note that the representative class of Möller's scheme [Mö101a] has $2^w + 1$ points. Thus, it is not optimized in terms of memory consumption. The enhanced version of Möller's scheme [Mö101b] is not optimized either. Its representative class has $2^{w-1} + 1$ points, thus one point is wasted.

4.4 Security Analysis

In this section we discuss the security of the proposed scheme. We consider the security against SPA, DPA, and the second-order DPA.

SPA: As we mentioned before, the proposed method computes scalar multiplication through the fixed pattern $|0...0x|0...0x|...|0...0x|$ for non-zero x . The attacker could distinguish elliptic addition and doubling in the scalar multiplication by measurement of the power consumption. However, he obtains the identical sequence $|D...DA|D...DA|...|D...DA|$ for all the scalars, where A and D denote elliptic addition and doubling, respectively. Therefore, he cannot detect the secret scalar by using SPA.

DPA: The use of projective randomization such as randomized projective coordinates [Cor99] or random isomorphic curves [JT01b] prevents against DPA. These countermeasures transform values of an inputted point into other different values whenever they perform, which thwarts in attacker's prediction about specific values in the computation.

Second-Order DPA: Okeya and Sakurai [OS02c] have proposed a second-order DPA attack against Möller's window method [Mö101a]. We notice that no experimental results are described in the paper [OS02c]. Thus, the reality of the second-order DPA attack is controversial. The attack is based on the plural accesses to same entries in the table. Let i -th (resp. j -th) digit be d_i (resp. d_j). If $d_i = d_j$ holds, the same entry in the table is accessed, in particular, the Hamming weights of the entries that are accessed are identical. If not, the Hamming weights are not identical in general. On the other hand, Hamming weight is

closely related to power consumption, thus, sufficiently many measurements on power consumption distinguish whether or not d_i is equal to d_j .

Since also our proposed method uses a pre-computed table, the second-order DPA may be applicable. In order to prevent against the second-order DPA, we propose the following countermeasure: Table randomization renewal. Once an entry in the table is loaded, the entry is re-randomized and rewrote. In other words, different data are loaded whenever entries are loaded, which prevents the second-order DPA.

Remark 1. Some cryptographic applications, e.g. EC-DSA require to compute the scalar multiplication dP for randomly chosen scalar d . Joye and Tymen proposed how to generate random NAF chains without transforming them from binary chains [JT01a]. In order to generate the random odd scalar, the proposed scheme can choose random signed odd $\{\pm 1, \pm 3, \dots, \pm(2^w - 1)\}$ for each digit with the treatments of MSB and LSB.

4.5 Computation Cost

We estimate the computation cost of the proposed scheme. Both the SPA-resistant version and the 2nd-order-DPA-resistant version are considered. The digit of our proposed scheme has the fixed pattern $|0\dots0x|$, where x is non-zero integer. In order to compute the digit we compute w -time ECDBLs and ECADD, which require $(4w + 12)M + (6w + 4)S$. If the Z -coordinate of the pre-computed points are converted to 1, it requires only $(4w + 8)M + (4w + 5)S$.

At first we estimate the computation time of the SPA-resistant scheme. We assume that the scalar multiplication is computed from the most significant. The pre-computed points are generated by

$$3P = 2P + P, 5P = 3P + 2P, \dots, (2^w - 1)P = (2^w - 3)P + 2P, \quad (2)$$

which require $(12M + 4S)(2^w - 3)$. These points are converted to the affine coordinate with the computation time $(3M + 1S + 1I)(2^w - 3)$ that is equal $(6M + 1S)(2^w - 3) - 3M + 1I$ by Montgomery's trick. Further, the use of pre-computation trick [OS02b] reduces the computation time, that is, $(5 \cdot 2^{w-1} + 2w - 10)M + (2^{w-1} + 2w - 3)S + wI$. As the post computation we compute $2P$ and one ECADD that require $16M + 8S$. After computing the scalar multiplication the point is pulled back to the affine coordinate that requires $3M + 1S + 1I$. Therefore, in total we need

$$(5 \cdot 2^{w-1} + (4k + 2)w + 8k + 9)M + (2^{w-1} + (4k + 2)w + 5k + 6)S + (w + 1)I, \quad (3)$$

where $k = \lceil n/w \rceil$.

Next we assume that the computation time of the scheme that is secure against the second order DPA. The generation of the pre-computed table require $(12M + 4S)(2^w - 3)$. Before we compute of the elliptic curve addition, the value of the table is randomized, which require $4M + 1S$. The main loop we need $k(4w + 12)M + k(6w + 4)S$. For the post computation we need $16M + 10S$. For the affine conversion we need $3M + 1S + 1I$. Therefore in total the computation time is

Table 1. Computation time of the proposed schemes

	$w = 2$	$w = 3$	$w = 4$	$w = 5$
SPA-resistant Scheme	$2205M$	$1942M$	$1807M$	$1781M$
Table Size (Proposed scheme)	640 bits	1280 bits	2560 bits	5120 bits
Table Size (Möller's scheme)	960 bits	1600 bits	2880 bits	5440 bits
2nd-DPA-resistant Scheme	$3071M$	$2929M$	$2453M$	$2537M$
Table Size (Proposed scheme)	960 bits	1920 bits	3840 bits	7680 bits
Table Size (Möller's scheme)	1440 bits	2400 bits	4320 bits	8160 bits

$$(12 \cdot 2^w + (4w + 16)k - 17)M + (4 \cdot 2^w + (6w + 5)k - 1)S + 1I. \quad (4)$$

Assuming that $S = 0.8M$, $I = 20M$, then we summarize the computation time for small $w = 2, 3, 4, 5$ for 160-bit scalar in Table 1.

The computation time of the proposed SPA-resistant scheme is comparable to that of the NAF, i.e., $(1816M)$. The proposed 2nd-order-DPA resistant scheme requires more time, because the pre-computed point must be randomized and the faster addition formula ECADDE $_{Z=1}$ can not be employed.

4.6 Relation to Schemes with Dummy Operations

The other possibility to achieve the SPA-resistant scheme is to insert dummy operations during the scalar multiplication [Cor99], [HM02]. Coron proposed a scheme with dummy operation for the standard binary method [Cor99]. Hitchcock *et al.* interpolates the NAF with dummy operation (We call it the HM dummy operation)[HM02].

Here we propose a simple construction that is secure against the SPA using the dummy operation. The proposed scheme is an extension of the Coron's dummy operation to the signed width- w NAF method. The proposed method is as follows:

Algorithm SPA-resistant Width- w NAF with Dummy

INPUT A n -bit scalar value d , point P , and $k = \lceil n/w \rceil$

OUTPUT $d * P$.

1. For $i = 0$ to k
 - 1.1. $u[i] \leftarrow d \bmod 2^w$, $d \leftarrow d - u[i]$, $d \leftarrow d/2^w$
2. Pre-compute $((|u[i]|)P)$ for all $u[i]$ $((|u[i]|)P \leftarrow P$ for $u[i] = 0$)
3. $Q[0] \leftarrow u[i]P$
4. For $i = k - 1$ down to 0
 - 4.1. $Q[0] \leftarrow w$ -time ECDBL($Q[0]$)
 - 4.2. $Q[1] \leftarrow \text{ECADD}(Q[0], u[i]P)$
 - 4.3. $Q[0] \leftarrow Q[\delta(u[i])]$
5. Return $Q[0]$

The modular reduction of “mods” in step 1.1 is the signed residue class that is same as that of the width- w NAF. However, the even points are also stored and the total size of the pre-computed table is 2^{w-1} . In step 4.2 we compute a dummy

Table 2. Several SPA-resistant schemes with dummy operations

Schemes	Efficiency	
Coron dummy [Cor99]	n ECDBLs + n ECADDs	$3096M$
HM dummy [HM02]	$(10n/9)$ ECDBLs + $(5n/9)$ ECADDs	$2459M$
Proposed Scheme $w = 2$	n ECDBLs + $(n/2)$ ECDBLs	2216M

elliptic curve addition if $d[i] = 0$ holds. The assignment $Q[0] \leftarrow Q[\delta(u[i])]$ in step 4.3 selects the proper value for the scalar multiplication, where $\delta(u[i])$ is 0 if $u[i] = 0$, and 1 otherwise. Note that although we used fast formula $\text{ECDBL}^{\mathcal{J}^m \rightarrow \mathcal{J}}$ for the proposed scheme without the dummy operation, it can not be computed in step 4.1 in some cases - after a dummy operation the coordinate remains the Jacobian coordinate (not the modified Jacobian coordinate). Thus we should use the modified Jacobian coordinate during the whole scalar multiplication. In Table 2, we compare the computation time of the proposed scheme with those of the Coron's dummy operations [Cor99] and the HM dummy operation [HM02]. In the estimation we assume that the computation time of squaring $S = 0.8M$ and inversion $I = 20M$, where M is the multiplication time of the base field. Our proposed scheme is faster than both the HM scheme and Coron scheme.

5 Conclusion

In this paper we proposed an SPA-resistant scheme based on the width- w NAF. The proposed construction clearly illustrates the relationship the width- w NAF and the our proposed scheme. Our proposed construction is optimal in the sense of both efficiency and memory. The memory requirement of our scheme is smaller than that of Möller's scheme, which is based on the signed 2^w -ary method. Indeed the tables size of the proposed scheme are 6% to 50% smaller than those of Möller's scheme for $w = 2, 3, 4, 5$, which are relevant choices in the sense of efficiency for 160-bit ECC. We also proposed an SPA-resistant scheme using the dummy operation based on width- w NAF. Comparing with the methods proposed by Hitchcock *et al.* using the NAF, our proposed method is simple and faster.

References

- [ANSI] ANSI X9.62, Public Key Cryptography for the Financial Services Industry, *The Elliptic Curve Digital Signature Algorithm (ECDSA)*, (1999). [333](#)
- [BJ02] Brier, É., Joye, M., *Weierstrass Elliptic Curves and Side-Channel Attacks*, Public Key Cryptography (PKC2002), LNCS2274, (2002), 335-345. [329](#), [332](#), [333](#)
- [BSS99] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 1999. [331](#)

- [CMO98] Cohen, H., Miyaji, A., Ono, T., *Efficient Elliptic Curve Exponentiation Using Mixed Coordinates*, Advances in Cryptology - ASIACRYPT '98, LNCS1514, (1998), 51-65. [330](#)
- [Cor99] Coron, J. S., *Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems*, Cryptographic Hardware and Embedded Systems (CHES'99), LNCS1717, (1999), 292-302. [329](#), [332](#), [333](#), [337](#), [339](#), [340](#)
- [DES] National Bureau of Standards, *Data Encryption Standard*, Federal Information Processing Standards Publication 46 (FIPS PUB 46), (1977). [332](#)
- [FGKS02] Fischer, W., Giraud, C., Knudsen, E. W., Seifert, J. P., *Parallel scalar multiplication on general elliptic curves over \mathbf{F}_p hedged against Non-Differential Side-Channel Attacks*, International Association for Cryptologic Research (IACR), Cryptology ePrint Archive 2002/007, (2002). Available at <http://eprint.iacr.org/> [329](#), [332](#)
- [HM02] Hitchcock, Y., Montague, P., *A New Elliptic Curve Scalar Multiplication Algorithm to Resist Simple Power Analysis*, Information Security and Privacy, 7th Australasian Conference, (ACISP 2002), LNCS2384, (2002), 214-225. [329](#), [339](#), [340](#)
- [IIT02] Itoh, K., Izu, T., and Takenaka, M., *Address-bit Differential Power Analysis on Cryptographic Schemes OK-ECDH and OK-ECDSA*, to appear in Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002), 2002.
- [IYTT02] Itoh, K., Yajima, J., Takenaka, M., and Torii, N., *DPA Countermeasures by improving the Window Method*, to appear in Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002), 2002. [329](#)
- [IEEE] IEEE P1363, Standard Specifications for Public-Key Cryptography. <http://grouper.ieee.org/groups/1363/> [330](#)
- [IT02] Izu, T., Takagi, T., *A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks*, Public Key Cryptography (PKC2002), LNCS2274, (2002), 280-296. [329](#), [332](#), [333](#)
- [JQ01] Joye, M., Quisquater, J. J., *Hessian elliptic curves and side-channel attacks*, Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 402-410. [333](#)
- [JT01a] Joye, M., Tymen, C., *Compact Encoding of Non-adjacent Forms with Applications to Elliptic Curve Cryptography*, Public Key Cryptography 2001 (PKC2001), pp.353-364, LNCS 1992, 2001. [338](#)
- [JT01b] Joye, M., Tymen, C., *Protections against differential analysis for elliptic curve cryptography: An algebraic approach*, Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 377-390. [333](#), [337](#)
- [Kob87] Koblitz, N., *Elliptic curve cryptosystems*, Math. Comp. 48, (1987), 203-209.
- [Koc96] Kocher, C., *Timing Attacks on Implementations of Diffie-Hellman, RSA,DSS, and Other Systems*, Advances in Cryptology - CRYPTO '96, LNCS1109, (1996), 104-113. [328](#), [332](#)
- [KJJ99] Kocher, C., Jaffe, J., Jun, B., *Differential Power Analysis*, Advances in Cryptology - CRYPTO '99, LNCS1666, (1999), 388-397. [328](#), [332](#)
- [KT92] K. Koyama and Y. Tsuruoka, *Speeding Up Elliptic Curve Cryptosystems using a Signed Binary Windows Method*, Advances in Cryptology - CRYPTO '92, LNCS740, (1992), pp.345-357. [331](#)
- [LS01] Liardet, P. Y., Smart, N. P., *Preventing SPA/DPA in ECC systems using the Jacobi form*, Cryptographic Hardware and Embedded System (CHES'01), LNCS2162, (2001), 391-401. [329](#), [333](#)

- [Mil86] Miller, V. S., *Use of elliptic curves in cryptography*, Advances in Cryptology - CRYPTO '85, LNCS218,(1986), pp.417-426.
- [MOC97] Atsuko Miyaji, Takatoshi Ono, Henri Cohen, *Efficient elliptic curve exponentiation*, Information and Communication Security (ICICS 1997), (1997), pp.282-291. [331](#)
- [Mööl01a] Möller, B., *Securing Elliptic Curve Point Multiplication against Side-Channel Attacks*, Information Security (ISC2001), LNCS2200, (2001), 324-334. [329](#), [337](#)
- [Mööl01b] Möller, B., *Securing elliptic curve point multiplication against side-channel attacks*, addendum: Efficiency improvement. <http://www.informatik.tu-darmstadt.de/TI/Mitarbeiter/moeller/ecc-scaisc01.pdf>, (2001). [329](#), [337](#)
- [Mööl02] Möller, B., *Parallelizable Elliptic Curve Point Multiplication Method with Resistance against Side-Channel Attacks*, Information Security Conference (ISC 2002), LNCS 2433, (2002), 402-413.
- [NIST] National Institute of Standards and Technology, FIPS 186-2, <http://csrc.nist.gov/publication/fips/fips186-2/fips186-2.pdf>
- [OA01] Oswald, E., Aigner, M., *Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks*, Cryptographic Hardware and Embedded Systems (CHES'01), LNCS2162, (2001), 39-50. [329](#), [333](#)
- [OMS01] Okeya, K., Miyazaki, K., Sakurai, K., *A Fast Scalar Multiplication Method with Randomized Projective Coordinates on a Montgomery-form Elliptic Curve Secure against Side Channel Attacks*, The 4th International Conference on Information Security and Cryptology (ICISC 2001), LNCS2288, (2002), 428-439. [333](#)
- [OS00] Okeya, K., Sakurai, K., *Power Analysis Breaks Elliptic Curve Cryptosystems even Secure against the Timing Attack*, Progress in Cryptology - INDOCRYPT 2000, LNCS1977, (2000), 178-190. [329](#), [332](#)
- [OS02a] Okeya, K., Sakurai, K., *On Insecurity of the Side Channel Attack Countermeasure using Addition-Subtraction Chains under Distinguishability between Addition and Doubling*, The 7th Australasian Conference in Information Security and Privacy, (ACISP 2002), LNCS2384, (2002), 420-435. [329](#), [333](#)
- [OS02b] Okeya, K., Sakurai, K., *Fast Multi-Scalar Multiplication Methods on Elliptic Curves with Precomputation Strategy using Montgomery Trick*, Cryptographic Hardware and Embedded System (CHES 2002), Pre-Proceedings, (2002), 566-581. [338](#)
- [OS02c] Okeya, K., Sakurai, K., *A Second-Order DPA Attack Breaks a Window-method based Countermeasure against Side Channel Attacks*, Information Security Conference (ISC 2002), LNCS 2433, (2002), 389-401. [337](#)
- [Osw02] Oswald, E., *Enhancing Simple Power-Analysis Attacks on Elliptic Curve Cryptosystems*, to appear in Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002), 2002.
- [RSA78] Rivest, R. L., Shamir, A., Adleman, L., *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM, Vol.21, No.2, (1978), 120-126. [332](#)
- [SEC] Standards for Efficient Cryptography Group (SECG), <http://www.secg.org>
- [Sol00] Solinas, J. A., *Efficient Arithmetic on Koblitz Curves*, Design, Codes and Cryptography, 19, (2000), 195-249. [329](#), [331](#)

- [Wal02a] Walter, C. D., *Some Security Aspects of the Mist Randomized Exponentiation Algorithm*, to appear in Workshop on Cryptographic Hardware and Embedded Systems 2002 (CHES 2002), 2002.
- [Wal02b] Walter, C. D., *Breaking the Liardet-Smart Randomized Exponentiation Algorithm*, to appear in CARDIS'02. 329, 333

Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation

Kirsten Eisenträger¹, Kristin Lauter², and Peter L. Montgomery²

¹ Department of Mathematics, University of California
Berkeley, CA 94720

eisentra@math.berkeley.edu

² Microsoft Research, One Microsoft Way, Redmond, WA 98052
{klauter,petmon}@microsoft.com

Abstract. We present an algorithm which speeds scalar multiplication on a general elliptic curve by an estimated 3.8% to 8.5% over the best known general methods when using affine coordinates. This is achieved by eliminating a field multiplication when we compute $2P+Q$ from given points P, Q on the curve. We give applications to simultaneous multiple scalar multiplication and to the Elliptic Curve Method of factorization. We show how this improvement together with another idea can speed the computation of the Weil and Tate pairings by up to 7.8%.

Keywords: Elliptic curve cryptosystem, elliptic curve arithmetic, scalar multiplication, ECM, pairing-based cryptosystem.

1 Introduction

This paper presents an algorithm which can speed scalar multiplication on a general elliptic curve, by doing some arithmetic differently. Scalar multiplication on elliptic curves is used by cryptosystems and signature schemes based on elliptic curves. Our algorithm saves an estimated 3.8% to 8.5% of the time to perform a scalar multiplication on a general elliptic curve, when compared to the best-known general methods. This savings is important because the ratio of security level to computation time and power required by a system is an important factor when determining whether a system will be used in a particular context.

Our main achievement eliminates a field multiplication whenever we are given two points P, Q on an elliptic curve and need $2P + Q$ (or $2P - Q$) but not the intermediate results $2P$ and $P + Q$. This sequence of operations occurs many times when, for example, left-to-right binary scalar multiplication is used with a fixed or sliding window size.

Some algorithms for simultaneous multiple scalar multiplication alternate doubling and addition steps, such as when computing $k_1P_1 + k_2P_2 + k_3P_3$ from given points P_1, P_2 , and P_3 . Such algorithms can use our improvement directly. We give applications of our technique to the Elliptic Curve Method for factoring and to speeding the evaluation of the Weil and Tate Pairings.

The paper is organized as follows. Section 2 gives some background on elliptic curves. Section 3 gives a detailed version of our algorithm. Section 4 estimates our savings compared to ordinary left-to-right scalar multiplication with windowing. Section 5 illustrates the improvement achieved with an example. It also describes applications to simultaneous multiple scalar multiplication and the Elliptic Curve Method for factoring. Section 6 adapts our technique to the Weil and Tate pairing algorithms. Appendix A gives the pseudocode for implementing the improvement, including abnormal cases.

2 Background

Elliptic curves are used for several kinds of cryptosystems, including key exchange protocols and digital signature algorithms [IEEE]. If q is a prime or prime power, we let \mathbb{F}_q denote the field with q elements. When $\gcd(q, 6) = 1$, an elliptic curve over the field \mathbb{F}_q is given by an equation of the form

$$E_{\text{simple}} : y^2 = x^3 + ax + b$$

with a, b in \mathbb{F}_q and $4a^3 + 27b^2 \neq 0$. (See [Silverman, p. 48].)

A more general curve equation, valid over a field of any characteristic, is considered in Appendix A. The general curve equation subsumes the case

$$E_2 : y^2 + xy = x^3 + ax^2 + b$$

with a, b in \mathbb{F}_q and $b \neq 0$, which is used over fields of characteristic 2.

In all cases the group used when implementing the cryptosystem is the group of points on the curve over \mathbb{F}_q . If represented in affine coordinates, the points have the form: (x, y) , where x and y are in \mathbb{F}_q and they satisfy the equation of the curve, as well as a distinguished point \mathbf{O} (called the *point at infinity*) which acts as the identity for the group law. Throughout this paper we work with affine coordinates for the points on the curve.

Points are added using a geometric group law which can be expressed algebraically through rational functions involving x and y . Whenever two points are added, forming $P + Q$, or a point is doubled, forming $2P$, these formulae are evaluated at the cost of some number of multiplications, squarings, and divisions in the field. For example, using E_{simple} , to double a point in affine coordinates costs 1 multiplication, 2 squarings, and 1 division in the field, not counting multiplication by 2 or 3 [BSS, p. 58]. To add two *distinct* points in affine coordinates costs 1 multiplication, 1 squaring, and 1 division in the field. Performing a doubling and an addition $2P + Q$ costs 2 multiplications, 3 squarings and 2 divisions if the points are added as $(P + P) + Q$, i.e., first double P and then add Q .

3 The Algorithm

Our algorithm performs a doubling and an addition, $2P + Q$, on an elliptic curve E_{simple} using only 1 multiplication, 2 squarings, and 2 divisions (plus an extra squaring when $P = Q$). This is achieved as follows: to form $2P + Q$, where

$P = (x_1, y_1)$ and $Q = (x_2, y_2)$, we first find $(P + Q)$, except we omit its y -coordinate, because we will not need that for the next stage. This saves a field multiplication. Next we form $(P + Q) + P$. So we have done two point additions and saved one multiplication. This trick also works when $P = Q$, i.e., when tripling a point. One additional squaring is saved when $P \neq Q$ because then the order of our operations avoids a point doubling.

Elliptic curve cryptosystems require multiplying a point P by a large number k . If we write k in binary form and compute kP using the left-to-right method of binary scalar multiplication, we can apply our trick at each stage of the partial computations.

Efficient algorithms for group scalar multiplication have a long history (see [Knuth] and [Gordon1998]), and optimal scalar multiplication routines typically use a combination of the left-to-right or right-to-left m -ary methods with sliding windows, addition-subtraction chains, signed representations, etc. Our procedure can be used on top of these methods for $m = 2$ to obtain a savings of up to 8.5% of the total cost of the scalar multiplication for curves over large prime fields, depending upon the window size and curve equation which are used. This is described in detail in Section 4.

3.1 Detailed Description of the Algorithm

Here are the detailed formulae for our procedure when the curve has the form E_{simple} and all the points are distinct, none equal to \mathbf{O} . Appendix A gives details for all characteristics. That appendix also covers special cases, where an input or an intermediate result is the point at infinity.

Suppose $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are distinct points on E_{simple} , and $x_1 \neq x_2$. The point $P + Q$ will have coordinates (x_3, y_3) , where

$$\begin{aligned}\lambda_1 &= (y_2 - y_1)/(x_2 - x_1), \\ x_3 &= \lambda_1^2 - x_1 - x_2, \quad \text{and} \\ y_3 &= (x_1 - x_3)\lambda_1 - y_1.\end{aligned}$$

Now suppose we want to add $(P + Q)$ to P . We must add (x_1, y_1) to (x_3, y_3) using the above rule. Assume $x_3 \neq x_1$. The result has coordinates (x_4, y_4) , where

$$\begin{aligned}\lambda_2 &= (y_3 - y_1)/(x_3 - x_1), \\ x_4 &= \lambda_2^2 - x_1 - x_3, \quad \text{and} \\ y_4 &= (x_1 - x_4)\lambda_2 - y_1.\end{aligned}$$

We can omit the y_3 computation, because it is used only in the computation of λ_2 , which can be computed without knowing y_3 as follows:

$$\lambda_2 = -\lambda_1 - 2y_1/(x_3 - x_1).$$

Omitting the y_3 computation saves a field multiplication. Each λ_2 formula requires a field division, so the overall saving is this field multiplication.

Table 1. Costs of simple operations on E_{simple}

Doubling	$2P$	2 squarings, 1 multiplication, 1 division
Add	$P \pm Q$	1 squaring, 1 multiplication, 1 division
Double-add	$2P \pm Q$	2 squarings, 1 multiplication, 2 divisions
Tripling	$3P$	3 squarings, 1 multiplication, 2 divisions
Triple-add	$3P \pm Q$	3 squarings, 1 multiplication, 3 divisions

This trick can also be applied to save one multiplication when computing $3P$, the triple of a point $P \neq \mathbf{O}$, where the λ_2 computation will need the slope of a line through two distinct points $2P$ and P .

This trick can be used twice to save 2 multiplications when computing $3P + Q = ((P + Q) + P) + P$. Thus $3P + Q$ can be computed using 1 multiplication, 3 squarings, and 3 divisions. Such a sequence of operations would be performed repeatedly if a multiplier were written in ternary form and left-to-right scalar multiplication were used. Ternary representation performs worse than binary representation for large random multipliers k , but the operation of triple and add might be useful in another context.

A similar trick works for elliptic curve arithmetic in characteristic 2, as is shown in the pseudocode in Appendix A.

Table 1 summarizes the costs of some operations on E_{simple} .

4 Comparison to Conventional Scalar Multiplication

In this section we analyze the performance of our algorithm compared to conventional left-to-right scalar multiplication. We will refer to adding two distinct points on the curve E as elliptic curve addition, and to adding a point to itself as elliptic curve doubling. Suppose we would like to compute kP_0 given k and P_0 , where the exponent k has n bits and n is at least 160.

Assume that the relative costs of field operations are 1 unit per squaring or general multiplication and α units per inversion. [BSS, p. 72] assumes that the cost of an inversion is between 3 and 10 multiplications. In some implementations the relative cost of an inversion depends on the size of the underlying field. Our own timings on a Pentium II give a ratio of 3.8 for a 160-bit prime field and 4.8 for a 256-bit prime field when not using Montgomery multiplication. Some hardware implementations for fast execution of inversion in binary fields yield inversion/multiplication ratios of 4.18 for 160-bit exponents and 6.23 for 256-bit exponents [KoçSav2002].

The straightforward left-to-right binary method needs about n elliptic curve doublings. If the window size is one, then for every 1-bit in the binary representation, we perform an elliptic curve doubling followed directly by an elliptic curve addition. Suppose about half of the bits in the binary representation of k are 1's. Then forming kP consists of performing n elliptic curve doublings and $n/2$ elliptic curve additions.

In general, independent of the window size, the number of elliptic curve doublings to be performed will be about n asymptotically, whereas the number of elliptic curve additions to be performed will depend on the window size. Define the value $0 < \varepsilon < 1$ for a given window size to be such that the number of elliptic curve additions to be performed is εn on average. For example with window size 1, ε is $1/2$.

If we fix a window size and its corresponding ε , then the conventional algorithm for scalar multiplication needs about $2n + \varepsilon n$ field squarings, $n + \varepsilon n$ field general multiplications, and $n + \varepsilon n$ field divisions. If one inversion costs α multiplications, then the cost of a division is $(\alpha + 1)$ multiplications. So the overall cost in field multiplications is

$$(2n + \varepsilon n) + (n + \varepsilon n) + (\alpha + 1)(n + \varepsilon n) = (4 + \alpha)n + (3 + \alpha)\varepsilon n.$$

Now we analyze the percentage savings obtained by our algorithm, not including precomputation costs. The above computation includes εn sub-computations of the form $2P_1 + P_2$. Writing each as $P_1 + (P_1 + P_2)$ saves one squaring per sub-computation, reducing the overall cost to $(4 + \alpha)n + (2 + \alpha)\varepsilon n$. The technique in Section 3 saves another multiplication per sub-computation, dropping the overall cost to $(4 + \alpha)n + (1 + \alpha)\varepsilon n$. This means we get a savings of

$$2\varepsilon / ((4 + \alpha) + (3 + \alpha)\varepsilon).$$

When the window size is 1 and the inversion/multiplication ratio α is assumed to be 4.18, this gives a savings of 8.5%. When α is assumed to be 6.23, we still obtain a savings of 6.7%. When the window size is 2 and $2P$ and $3P$ have been precomputed, we find that $\varepsilon = 3/8$. So when α is 4.18, we get a savings of 6.9%, and when α is 6.23, we still obtain a savings of 5.5%. Similarly if the window size is 4, and we have precomputed small multiples of P , we still achieve a savings of 3.8% to 4.8%, depending on α .

Another possibility is using addition/subtraction chains and higher-radix methods. The binary method described in [IEEE, section A.10.3] utilizes addition/subtraction chains and does about $2n/3$ doublings and $n/3$ double-adds (or double-subtracts), so $\varepsilon = 1/3$ in this case. (See [Gordon1998, section 2.3] for an explanation of how we obtain $\varepsilon = 1/3$ in this case.) With $\alpha = 4.18$, we get a 6.3% improvement.

Scalar multiplication algorithms that use addition/subtraction chains as well as sliding window size may have lower ε , but we still obtain at least a 4.2% savings if $\varepsilon > 0.2$ and $\alpha = 4.18$.

[SaSa2001, Section 3.3] presents some possible trade-offs arising from different inversion/multiplication ratios. We discuss this further in Section 5.3.

5 Examples and Applications

5.1 Left-to-Right Binary Scalar Multiplication

Suppose we would like to compute $1133044P = (10001010010011110100)_2 P$ with left-to-right binary method. We will do this twice, the standard way and

the new way. For each method, we assume that $3P$ has been precomputed. The next table compares the number of operations needed (a = point additions, d = point doublings, div = field divisions, s = field squarings, m = field multiplies):

	Standard	Improved
$1133044P = 4(283261P)$	$2d$	$2d$
$283261P = 128(2213P) - 3P$	$7d + 1a$	$6d + 2a(\text{save } 1m)$
$2213P = 8(277P) - 3P$	$3d + 1a$	$2d + 2a(\text{save } 1m)$
$277P = 8(35P) - 3P$	$3d + 1a$	$2d + 2a(\text{save } 1m)$
$35P = 8(4P) + 3P$	$3d + 1a$	$2d + 2a(\text{save } 1m)$
$4P = P + 3P$	$1a$	$1a$
Total:	$23div + 41s + 23m$	$23div + 37s + 19m$

This saves 4 squarings and 4 multiplications. Estimating the division cost at about 5 multiplications, this savings translates to about 4.47%.

5.2 Simultaneous Multiple Scalar Multiplication

Another use of our elliptic curve double-add technique is multiple scalar multiplication, such as $k_1P_1 + k_2P_2 + k_3P_3$, where the multipliers k_1 , k_2 , and k_3 have approximately the same length. One algorithm creates an 8-entry table with

$$\mathbf{O}, \quad P_1, \quad P_2, \quad P_2 + P_1, \quad P_3, \quad P_3 + P_1, \quad P_3 + P_2, \quad P_3 + P_2 + P_1.$$

Subsequently it uses one elliptic curve doubling followed by the addition of a table entry, for each multiplier bit [Möller2001]. About 7/8 of the doublings are followed by an addition other than \mathbf{O} .

To form $29P_1 + 44P_2$, for example, write the multipliers in binary form: $(011101)_2$ and $(101100)_2$. Scanning these left-to-right, the steps are

Bits	Table entry	Action
0, 1	P_2	$T := P_2$
1, 0	P_1	$T := 2T + P_1 = P_1 + 2P_2$
1, 1	$P_1 + P_2$	$T := 2T + (P_1 + P_2) = 3P_1 + 5P_2$
1, 1	$P_1 + P_2$	$T := 2T + (P_1 + P_2) = 7P_1 + 11P_2$
0, 0	\mathbf{O}	$T := 2T = 14P_1 + 22P_2$
1, 0	P_1	$T := 2T + P_1 = 29P_1 + 44P_2$

There is one elliptic curve addition ($P_1 + P_2$) to construct the four-entry table, four doublings immediately followed by an addition, and one doubling without an addition. While doing 10 elliptic curve operations, our technique is used four times. Doing the multipliers separately, say by the addition-subtraction chains

$$1, 2, 4, 8, 7, 14, 28, 29 \quad \text{and} \quad 1, 2, 4, 6, 12, 24, 48, 44$$

takes seven elliptic curve operations per chain, plus a final add (15 total).

5.3 Elliptic Curve Method of Factorization

The Elliptic Curve Method (ECM) of factoring a composite integer N chooses an elliptic curve E with coefficients modulo N . ECM multiplies an initial point P_0 on E by a large integer k , working in the ring $\mathbb{Z}/N\mathbb{Z}$ rather than over a field. ECM may encounter a zero divisor while trying to invert a nonzero integer, but that is good, because it leads to a factorization of N . ECM uses only the x -coordinate of kP_0 .

[Mont1987, pp. 260ff] proposes a parameterization, $By^2 = x^3 + Ax^2 + x$, which uses no inversions during a scalar multiplication and omits the y -coordinate of the result. Its associated costs for computing the x -coordinate are

$P + Q$ from P, Q	2 squarings, 4 multiplications
$2P$ from P	2 squarings, 3 multiplications

To form kP from P for a large n -bit integer k , this method uses about $4n$ squarings and $7n$ multiplications, working from the binary representation of k . Some variations [MontLucas] use fewer steps but are harder to program.

In contrast, using our technique and the method in [IEEE, section A.10.3], we do about $2n/3$ doublings and $n/3$ double-adds (or double-subtracts). By Table 1, the estimated cost of kP is $2n$ squarings, n multiplications and $4n/3$ divisions.

The new technique is superior if $4n/3$ divisions cost less than $2n$ squarings and $6n$ multiplications. A division can be implemented as an inversion plus a multiplication, so the new technique is superior if an inversion is cheaper than 1.5 squarings and 3.5 multiplications.

[Mont1987] observes that one may trade two independent inversions for one inversion and three multiplications, using $x^{-1} = y(xy)^{-1}$ and $y^{-1} = (xy)^{-1}x$. When using many curves to (simultaneously) tackle the same composite integer, the asymptotic cost per inversion drops to 3 multiplications.

6 Application to Weil and Tate Pairings

The Weil and Tate pairings are becoming important for public-key cryptography [Joux2002]. The algorithms for these pairings construct rational functions with a prescribed pattern of poles and zeroes. An appendix to [BoFr2001] describes Miller's algorithm for computing the Weil pairing on an elliptic curve in detail.

Fix an integer $m > 0$ and an m -torsion point P on an elliptic curve E . Let f_1 be any nonzero field element. For an integer $c > 1$, let f_c be a function on E with a c -fold zero at P , a simple pole at cP , a pole of order $c - 1$ at \mathbf{O} , and no other zeroes or poles. When $c = m$, this means that f_m has an m -fold zero at P and a pole of order m at \mathbf{O} . Corollary 3.5 on page 67 of [Silverman] asserts that such a function exists. This f_c is unique up to a nonzero multiplicative scalar. Although f_c depends on P , we omit the extra subscript P .

The Tate pairing evaluates a quotient of the form $f_m(Q_1)/f_m(Q_2)$ for two points Q_1, Q_2 on E (see, for example, [BKLS2002]). (The Weil pairing has four

such computations.) Such evaluations can be done iteratively using an addition/subtraction chain for m , once we know how to construct f_{b+c} and f_{b-c} from (f_b, bP) and (f_c, cP) . Let $g_{b,c}$ be the line passing through the points bP and cP . When $bP = cP$, this is the tangent line to E at bP . Let g_{b+c} be the vertical line through $(b+c)P$ and $-(b+c)P$. Then we have the useful formulae

$$f_{b+c} = f_b \cdot f_c \cdot \frac{g_{b,c}}{g_{b+c}} \quad \text{and} \quad f_{b-c} = \frac{f_b \cdot g_b}{f_c \cdot g_{-b,c}}.$$

Denote $h_b = f_b(Q_1)/f_b(Q_2)$ for each integer b . Although f_b was defined only up to a multiplicative constant, h_b is well-defined. We have

$$h_{b+c} = h_b \cdot h_c \cdot \frac{g_{b,c}(Q_1) \cdot g_{b+c}(Q_2)}{g_{b,c}(Q_2) \cdot g_{b+c}(Q_1)} \quad \text{and} \quad h_{b-c} = \frac{h_b \cdot g_b(Q_1) \cdot g_{-b,c}(Q_2)}{h_c \cdot g_b(Q_2) \cdot g_{-b,c}(Q_1)}. \quad (1)$$

So far in the literature, only the f_{b+c} formula appears, but the f_{b-c} formula is useful if using addition/subtraction chains. The addition/subtraction chain iteratively builds h_m along with mP .

6.1 Using the Double-Add Trick with Parabolas

We now describe an improved method for obtaining $(h_{2b+c}, (2b+c)P)$ given (h_b, bP) and (h_c, cP) . The version of Miller's algorithm described in [BKLS2002] uses a left-to-right binary method with window size one. That method would first compute $(h_{2b}, 2bP)$ and later $(h_{2b+c}, (2b+c)P)$. We propose to compute $(h_{2b+c}, (2b+c)P)$ directly, producing only the x -coordinate of the intermediate point $bP + cP$. To combine the two steps, we construct a parabola through the points $bP, bP, cP, -2bP - cP$.

To form f_{2b+c} , we form f_{b+c} and f_{b+c+b} . The latter can be expressed as

$$f_{2b+c} = f_{b+c} \cdot \frac{f_b \cdot g_{b+c,b}}{g_{2b+c}} = \frac{f_b \cdot f_c \cdot g_{b,c}}{g_{b+c}} \cdot \frac{f_b \cdot g_{b+c,b}}{g_{2b+c}} = \frac{f_b \cdot f_c \cdot f_b}{g_{2b+c}} \cdot \frac{g_{b,c} \cdot g_{b+c,b}}{g_{b+c}}.$$

We replace $(g_{b,c} \cdot g_{b+c,b})/g_{b+c}$ by the parabola, whose formula is given below. Evaluate the formula for f_{2b+c} at Q_1 and Q_2 to get a formula for h_{2b+c} .

6.2 Equation for Parabola Through Points

If R and S are points on an elliptic curve E , then there is a (possibly degenerate) parabolic equation passing through R twice (i.e., tangent at R) and also passing through S and $-2R - S$. Using the notations $R = (x_1, y_1)$ and $S = (x_2, y_2)$ with $R + S = (x_3, y_3)$ and $2R + S = (x_4, y_4)$, a formula for this parabola is

$$\frac{(y + y_3 - \lambda_1(x - x_3))(y - y_3 - \lambda_2(x - x_3))}{x - x_3}. \quad (2)$$

The left half of the numerator of (2) is a line passing through R, S , and $-R - S$ whose slope is λ_1 . The right half of the numerator is a line passing through

$R + S$, R , and $-2R - S$, whose slope is λ_2 . The denominator is a (vertical) line through $R + S$ and $-R - S$. The quotient has zeros at R , R , S , $-2R - S$ and a pole of order four at \mathbf{O} .

We simplify (2) by expanding it in powers of $x - x_3$. Use the equation for E to eliminate references to y^2 and y_3^2 .

$$\begin{aligned} & \frac{y^2 - y_3^2}{x - x_3} - \lambda_1(y - y_3) - \lambda_2(y + y_3) + \lambda_1\lambda_2(x - x_3) \\ &= x^2 + xx_3 + x_3^2 + a + \lambda_1\lambda_2(x - x_3) - \lambda_1(y - y_3) - \lambda_2(y + y_3) \\ &= x^2 + (x_3 + \lambda_1\lambda_2)x - (\lambda_1 + \lambda_2)y + \text{constant}. \end{aligned} \quad (3)$$

Knowing that (3) passes through $R = (x_1, y_1)$, one formula for the parabola is

$$(x - x_1)(x + x_1 + x_3 + \lambda_1\lambda_2) - (\lambda_1 + \lambda_2)(y - y_1). \quad (4)$$

In the previous section we can now replace $(g_{b,c} \cdot g_{b+c,b})/g_{b+c}$ by the parabola (4) with $R = bP$ and $S = cP$.

Formula (4) for the parabola does not reference y_3 and is never identically zero since its x^2 coefficient is 1. Appendix A gives a formula for this parabola in degenerate cases, as well as for a more general curve.

6.3 Savings

We claim the pairing algorithm needs less effort to evaluate a parabola at a point than to evaluate lines and take their product at that point. The parabola does not reference y_3 , so we can omit the y -coordinate of $bP + cP$ and can use the double-add trick.

Here is a precise analysis of the savings we obtain by using the parabola when computing the Tate pairing. Again assume that we use the binary method in [IEEE, section A.10.3] to form mP , where m has n bits. (It does $2n/3$ doublings and $n/3$ double-adds or double-subtracts.) We manipulate the numerator and denominator of h_j separately, doing one division $h_j = h_{\text{num},j}/h_{\text{denom},j}$ at the very end.

Analysis of Doubling Step: The analysis of the doubling step is the same in the standard and in the new algorithms. Suppose we want to compute $(h_{2b}, 2bP)$ from (h_b, bP) . We need an elliptic curve doubling to compute $2(bP)$, after which we apply (1). If $bP = (x_1, y_1)$ and $2bP = (x_4, y_4)$ then

$$\frac{g_{b,b}}{g_{2b}} = \frac{y - y_1 - \lambda_1(x - x_1)}{x - x_4}. \quad (5)$$

The doubling (including λ_1 computation) costs 3 multiplications and a division. Evaluating (5) at Q_1 and Q_2 (as fractions) costs 2 multiplications. Multiplying four fractions in (1) costs 6 multiplications. The net cost is $3 + 2 + 6 = 11$ field multiplications (or squarings) and a field division.

Analysis of Double-Add Step: The standard algorithm performs one doubling followed by an addition to compute $(h_{2b+c}, (2b+c)P)$ from (h_b, bP) and

(h_c, cP) . Similar to the above analysis we can compute the cost as 21 field multiplications and 2 divisions. [The cost would be one fewer multiplication if one does two elliptic curve additions: $(2b + c)P = (bP + cP) + bP$.]

The new algorithm does one elliptic curve double-add operation. It costs only one multiplication to construct the coefficients of the parabola (4), because we computed λ_1 and λ_2 while forming $(2b + c)P$. Evaluating the parabola (and the vertical line g_{2b+c}) twice costs four multiplications. Multiplying five fractions costs another 8 multiplications. The total cost is $3 + 1 + 4 + 8 = 16$ field multiplications and 2 field divisions.

Total Savings: Estimating a division as 5.18 multiplications, the standard algorithm for (h_m, mP) takes $(16.18 \cdot 2n/3) + (31.36 \cdot n/3) = (21.24)n$ steps, compared to $(16.18 \cdot 2n/3) + (26.36 \cdot n/3) = 19.57n$ steps for the new method, a 7.8% improvement. A Weil pairing algorithm using the parabola will also save 7.8% over Miller’s algorithm, because we can view the Weil pairing as “two applications of the Tate pairing”, each saving 7.8%.

Sometimes (e.g., [BLS2001]) one does multiple Tate pairings with P fixed but varying Q_1 and Q_2 . If one has precomputed all coefficients of the lines and parabolas, then the costs of evaluation are 8 multiplications per doubling step or addition step, and 12 multiplications per combined double-add step. The overall costs are $32n/3$ multiplications per evaluation with the traditional method and $28n/3$ multiplications with the parabolas, a 12.5% improvement.

References

- [BKLS2002] Paulo S. L. M. Barreto, Hae Y. Kim, Ben Lynn and Michael Scott, Efficient algorithms for pairing-based cryptosystems, in *Advances in Cryptology – Crypto 2002*, M. Yung (Ed.), LNCS **2442**, Springer-Verlag, 2002, pp. 354–368. [349](#), [350](#)
- [BSS] I. F. Blake, G. Seroussi, N. P. Smart, *Elliptic Curves in Cryptography*, LMS **265** Cambridge University Press, 1999. [344](#), [346](#)
- [BoFr2001] Dan Boneh and Matt Franklin, Identity-based encryption from the Weil pairing, in *Advances in Cryptology – Crypto 2001*, J. Kilian (Ed.), LNCS **2139**, Springer-Verlag, 2001, pp. 213–229. Appendix available at <http://crypto.stanford.edu/~dabo/papers/ibe.pdf>. [349](#)
- [BLS2001] Dan Boneh, Ben Lynn, and Hovav Shacham, Short signatures from the Weil pairing, in *Advances in Cryptology – Asiacrypt 2001*, C. Boyd (Ed.), LNCS **2248**, Springer-Verlag, 2001, pp. 514–532. [352](#)
- [Gordon1998] D. M. Gordon, A survey of fast exponentiation methods, *J. Algorithms*, **27**, pp. 129–146, 1998. [345](#), [347](#)
- [IEEE] IEEE Standard Specifications for Public-Key Cryptography, IEEE Std 1363–2000, IEEE Computer Society, 29 August 2000. [344](#), [347](#), [349](#), [351](#), [353](#)
- [Joux2002] Antoine Joux, The Weil and Tate Pairings as building blocks for public key cryptosystems (survey), in *Algorithmic Number Theory, 5th International Symposium ANTS-V, Sydney, Australia, July 7–12, 2002 Proceedings*, Claus Fieker and David R. Kohel (Eds.), LNCS **2369**, Springer-Verlag, 2002, pp. 20–32. [349](#)

- [Knuth] Donald E. Knuth, *The Art of Computer Programming*, vol. 2, Seminumerical Algorithms, Addison-Wesley, 3rd edition, 1997. [345](#)
- [KoçSav2002] C. K. Koç and E. Savaş, Architectures for Unified Field Inversion with Applications in Elliptic Curve Cryptography, *The 9th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2002, Dubrovnik, Croatia, September 15–18, 2002*, vol. 3, pp. 1155–1158. [346](#)
- [Möller2001] Bodo Möller, Algorithms for multi-exponentiation, in *Selected Areas in Cryptography 2001, Toronto, Ontario, Serge Vaudenay and Amr M. Youssef (Eds.)*, LNCS **2259**, Springer-Verlag, 2002, pp. 165–180. [348](#)
- [Mont1987] Peter L. Montgomery, Speeding the Pollard and Elliptic Curve Methods of Factorization, *Math. Comp.*, v. **48**(1987), pp. 243–264. [349](#)
- [MontLucas] Peter L. Montgomery, Evaluating Recurrences of Form $X_{m+n} = f(X_m, X_n, X_{m-n})$ via Lucas Chains. Available at <ftp://cwi.nl:/pub/pmontgom/lucas.ps.gz>. [349](#)
- [SaSa2001] Yasuyuki Sakai, Kouichi Sakurai, On the Power of Multidoubling in Speeding up Elliptic Scalar Multiplication, in *Selected Areas in Cryptography 2001, Toronto, Ontario, Serge Vaudenay and Amr M. Youssef (Eds.)*, LNCS **2259**, Springer-Verlag, 2002, pp. 268–283. [347](#)
- [Silverman] Joseph H. Silverman, *The Arithmetic of Elliptic Curves*, Springer-Verlag, GTM **106**, 1986. [344](#), [349](#), [353](#)

A Pseudocode

The general Weierstrass form for the equation of an elliptic curve is:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (6)$$

subject to the condition that the coefficients a_1, a_2, a_3, a_4, a_6 satisfy a certain inequality to prevent singularity [Silverman, p. 46]. The negative of a point $P = (x_1, y_1)$ on (6) is $-P = (x_1, -a_1x_3 - a_3 - y_1)$. [This seems to require a multiplication a_1x_3 , but in practice a_1 is 0 or 1.] If $P = (x_1, y_1)$ is a finite point on (6), then the tangent line at P has slope

$$\lambda_1 = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}. \quad (7)$$

Figure 1 gives the pseudocode for implementing the savings for an elliptic curve of this general form. Given two points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ on E , it describes how to compute $2P + Q$ as well as the equation for a (possibly degenerate) parabola through P, P, Q , and $-(2P + Q)$.

Often the curve coefficients in (6) are chosen to simplify (7) — the precise choices depend on the field. For example, it is common in characteristic 2 [IEEE, p. 115] to choose $a_1 = 1$ and $a_3 = a_4 = 0$, in which case (7) simplifies to $\lambda_1 = x_1 + y_1/x_1$.

```

if ( $P = \mathbf{O}$ ) then
    if ( $Q = \mathbf{O}$ ) then
        parabola = 1;
    else
        parabola =  $x - x_2$ ;
    end if
    return  $Q$ ;
else if ( $Q = \mathbf{O}$ ) then
    if (denominator of (7) is zero) then
        parabola =  $x - x_1$ ;
        return  $\mathbf{O}$ ;
    end if
    Get tangent slope  $\lambda_1$  from (7);
    parabola =  $y - y_1 - \lambda_1(x - x_1)$ ;
     $x_3 = \lambda_1(\lambda_1 + a_1) - a_2 - 2x_1$ ;
     $y_3 = \lambda_1(x_1 - x_3) - a_1x_3 - a_3 - y_1$ ;
    return ( $x_3, y_3$ );
else
    if ( $x_1 \neq x_2$ ) then
         $\lambda_1 = (y_1 - y_2)/(x_1 - x_2)$ ; /* slope of line through  $P, Q$ . */
    else if ( $y_1 \neq y_2$  OR denominator of (7) is zero) then
        parabola =  $(x - x_1)^2$ ;
        return  $P$ ; /*  $P$  and  $Q$  must be negatives, so  $2P + Q = P$ . */
    else
        Get tangent slope  $\lambda_1$  from (7);
    end if
     $x_3 = \lambda_1(\lambda_1 + a_1) - a_2 - x_1 - x_2$ ;
    /* Think  $y_3 = \lambda_1(x_1 - x_3) - a_1x_3 - a_3 - y_1$ . */
    if ( $x_3 = x_1$ ) then
        parabola =  $y - y_1 - \lambda_1(x - x_1)$ ;
        return  $\mathbf{O}$ ; /*  $P + Q$  and  $P$  are negatives. */
    end if /* Think  $\lambda_2 = (y_1 - y_3)/(x_1 - x_3)$  */
     $\lambda_2 = (a_1x_3 + a_3 + 2y_1)/(x_1 - x_3) - \lambda_1$ ;
     $x_4 = \lambda_2(\lambda_2 + a_1) - a_2 - x_1 - x_3$ ;
     $y_4 = \lambda_2(x_1 - x_4) - a_1x_4 - a_3 - y_1$ ;
    parabola =  $(x - x_1)(x - x_4 + (\lambda_1 + \lambda_2 + a_1)\lambda_2) - (\lambda_1 + \lambda_2 + a_1)(y - y_1)$ ;
    return ( $x_4, y_4$ );
end if

```

Fig. 1. Algorithm for computing $2P + Q$ and the equation for a parabola through P, P, Q , and $-(2P + Q)$, where $P = (x_1, y_1)$ and $Q = (x_2, y_2)$

Two Efficient and Provably Secure Schemes for Server-Assisted Threshold Signatures

Shouhuai Xu^{1,*} and Ravi Sandhu²

¹ Department of Information and Computer Science
University of California at Irvine
`shxu@ics.uci.edu`

² SingleSignOn.Net and Laboratory for Information Security Technology
George Mason University
`sandhu@gmu.edu`

Abstract. Secrecy of private signing keys is one of the most important issues in secure electronic commerce. A promising solution to this problem is to distribute the signing function among multiple parties. However, a threshold signature scheme typically assumes that the shared signing function can only be activated by a quorum number of parties, which is inappropriate in settings where a user employs some public servers for a threshold protection of her private signing function (therefore the name “server-assisted threshold signatures”).

In this paper we present two efficient and provably secure schemes for server-assisted threshold signatures, where the signing function is activated by a user (but in certain enhanced way). The first one (we call **TPAKE-HTSig**) is tailored for the setting where a user has a networked device that is powerful enough to efficiently compute modular exponentiations. The second one (we call **LW-TSig**) is tailored for the setting where a user has a smart card without a cryptographic co-processor. Modular construction of the schemes ensures that any module can be substituted without weakening security of the resultant scheme, as long as the substitutive one satisfies certain security requirement. In addition to the two schemes, we also present a taxonomy of systems protecting private signing functions.

1 Introduction

Secrecy of private signing keys is one of the most important issues in secure electronic commerce. Threshold signatures enable us to achieve better security (i.e., compromise of certain number of participants does not expose the private key) and availability (i.e., a quorum number of participants suffice to sign messages). However, threshold signature schemes typically assume that the shared signing function can only be activated by a quorum number of participants; otherwise, the participant able to activate the function can always get signatures

* Work mostly done while at the Laboratory for Information Security Technology,
George Mason University.

without compromising the private key. This assumption is relevant in certain settings (e.g., enterprise applications) but inappropriate in settings where a user employs some public servers (which perhaps provide service for economic incentives) to provide a threshold protection for her private signing function (therefore the name “server-assisted threshold signatures”).

1.1 Our Contributions

We propose two efficient and provably secure schemes for server-assisted threshold signatures, where the signing function is activated by a user (but in certain enhanced way). The schemes are obtained via a *modular composition* approach, thereby any module can be substituted without weakening security of the resultant scheme, as long as the substitutive one (with, for instance, better performance) satisfies certain security requirements. The first scheme (we call **TPAKE-HTSig**) is tailored for the setting where a user has a networked device that is powerful enough to efficiently compute modular exponentiations. The second scheme (we call **LW-TSig**) is tailored for the setting where a user has a smart card with no cryptographic co-processor. In addition to the two schemes, we also present a taxonomy of systems protecting private signing functions.

REMARK. Having said that our schemes allow the substitution of component modules as long as certain security requirement is satisfied, we stress that our approach is less general than the approach of Canetti [10].

OUTLINE. In Section 2 we present certain cryptographic preliminaries. In Section 3 we present our first scheme **TPAKE-HTSig**, of which a formal security analysis is given in Appendix B. In Section 4 we present our second scheme **LW-TSig**. In Section 5, we present a taxonomy of systems for protecting private signing functions (including our schemes). We conclude in section 6. Due to space limitation, we eliminate from this version many discussions (including security analysis of **LW-TSig**), which will be found in the full version of this paper.

2 Cryptographic Preliminaries

MESSAGE AUTHENTICATION CODES (MACs). A function family $\{f_k\}_k$, where $k \in \{0, 1\}^\kappa$ for some security parameter κ , is a secure MAC family if any adversary \mathcal{A} of probabilistic polynomial-time succeeds in the following game only with negligible probability. First a random key $k \in \{0, 1\}^\kappa$ is chosen; then \mathcal{A} adaptively chooses messages m_1, \dots, m_n and receives the corresponding MAC values $f_k(m_1), \dots, f_k(m_n)$. \mathcal{A} succeeds if it manages to generate a pair $\langle m, tag \rangle$, where $m \neq m_1, \dots, m_n$ but $tag = f_k(m)$. We refer the reader to [2] for details.

SIGNATURE SCHEMES. A signature scheme **Sig** consists of three algorithms (**Sig.Init**, **Sig.Sign**, **Sig.Ver**). Taken as input a security parameter κ , the probabilistic key generation algorithm **Sig.Init** returns a pair of public and private keys (Y, X) . Taken as input a message m and the private key X , the signing algorithm **Sig.Sign** outputs a signature σ . Taken as input a tag σ , the public key Y , and a message m , the verification algorithm **Sig.Ver** returns TRUE if σ is

a valid signature, and FALSE otherwise. We require a signature scheme be secure against adaptive chosen-message attack [17, 36].

THRESHOLD SIGNATURE SCHEMES. A threshold signature scheme TSig consists of three algorithms ($\text{TSig}.\text{Init}$, $\text{TSig}.\text{Sig}$, $\text{TSig}.\text{Ver}$). We focus on a *subclass* of signature schemes such that their threshold version falls into the following framework of specification; the motivation is to make concise the security proof of the resultant scheme. (Nonetheless, this subclass covers many *practical* and popular signature schemes [31, 5, 33, 11].)

1. Taken as input a security parameter κ , the number of tolerated corrupt servers t , and the number of servers n , $\text{TSig}.\text{Init}$ outputs a public key Y so that the corresponding private key X is shared among the n servers $\{S_1, \dots, S_n\}$ using an appropriate secret sharing scheme [34, 13, 28]. Let $X \xleftarrow{(t+1,n)} (X^{(1)}, \dots, X^{(n)})$ denote such a secret sharing so that S_i holds $X^{(i)}$, where $1 \leq i \leq n$.
2. To sign a message m , $\text{TSig}.\text{Sig}$ runs as follows: S_{i_j} generates a partial signature $\sigma^{(i_j)} = g_1(m, X^{(i_j)})$, then anyone can compute $\sigma = g_2(\sigma^{(i_1)}, \dots, \sigma^{(i_w)})$ from w valid partial signatures, where $t + 1 \leq w \leq n$, $1 \leq i_j \leq n$ for $1 \leq j \leq w$, g_1 and g_2 are certain algorithms that depend on Sig .
3. $\text{TSig}.\text{Ver}$ is the same as $\text{Sig}.\text{Ver}$.

A threshold signature scheme TSig is secure if, (1) it is unforgeable under adaptive chosen-message attack, where unforgeability is captured by presenting a polynomial-time simulator that is able to emulate TSig by having oracle access to the underlying Sig , and (2) it is robust, which means that an adversary having corrupted t servers is still unable to prevent it from functioning. Concrete threshold signature schemes can be found in, for instance, [16, 18, 30, 35].

A special and useful case of threshold signature schemes is the so-called two-party signature schemes $2\text{Sig} = (\text{2Sig}.\text{Init}, \text{2Sig}.\text{Sig}, \text{2Sig}.\text{Ver})$, where $t = 1$ and $n = 2$. In this case, robustness is weakened due to the deployment of a 2-out-of-2 secret sharing. We refer the reader to [6, 25, 26, 38] for formal treatments of some two-party signature schemes.

HYBRID THRESHOLD SIGNATURE SCHEMES. A hybrid threshold signature scheme HTSig is the hybrid of a *two-party* signature scheme 2Sig and a *multi-party* signature scheme TSig corresponding to the same underlying signature scheme Sig . In such a scheme, the privileges of the participants are weighted; for example, as shown in our first scheme TPAKE-HTSig , no signatures can be generated without the consent of a user, no matter how the servers collaborate. A hybrid threshold signature scheme HTSig consists of three algorithms ($\text{HTSig}.\text{Init}$, $\text{HTSig}.\text{Sig}$, $\text{HTSig}.\text{Ver}$). Taken as input a security parameter κ and the desired access structure in the underlying secret sharing scheme, the initialization algorithm $\text{HTSig}.\text{Init}$ outputs a public key Y so that the corresponding private key X is shared among the participants (according to the given access structure). Taken as input a message m and the shares of the private key, $\text{HTSig}.\text{Sig}$ outputs a signature σ . Typically, $\text{HTSig}.\text{Ver}$ is the same as $\text{Sig}.\text{Ver}$. In Sec-

tion 3.1 we will specify how to construct, and security definition of, HTSig for the subclass of signature schemes we are interested in.

THRESHOLD PASSWORD-AUTHENTICATED KEY EXCHANGE SCHEMES. A threshold password-authenticated key exchange scheme, or TPAKE for short, enables a set of servers to collaboratively authenticate a user using a password such that compromise of certain number of servers does not enable the adversary to conduct an off-line dictionary attack. A TPAKE scheme consists of two protocols: $(\text{TPAKE}.\text{Init}, \text{TPAKE}.\text{Login})$. Suppose κ is the main security parameter, t is the number of tolerated corrupt servers, and n is the number of servers. The basic functionality of $\text{TPAKE}.\text{Init}$ is:

1. The servers collaboratively generate certain system-wide configurations (if any), some of which may be published.
2. Each user chooses her own password and sends its transformed version (e.g., after certain cryptographically-strong transformation) to the servers.

The invocation protocol $\text{TPAKE}.\text{Login}$ is an algorithm of functionality that after a user successfully authenticates herself to the servers, the user shares a different on-the-fly session key with each of the servers. We will not present any concrete $\text{TPAKE}.\text{Login}$ (indeed we don't pin down on any concrete TPAKE), instead we will present a specification (including a *security* and *robustness* definition) of TPAKE, namely $\text{TPAKE}.\text{Login}$, in Appendix A. Note that the first concrete TPAKE has been available [27].

3 The First Scheme: TPAKE-HTSig

Our first scheme is called TPAKE-HTSig, which stands for “Threshold Password-Authenticated Key Exchange based’ server-assisted Hybrid Threshold Signatures.” TPAKE-HTSig is tailored for the following real-world scenario: a user having a networked device wants to enjoy a threshold protection of her private signing function by taking advantage of multiple servers, where the user device is powerful enough to efficiently compute modular exponentiations. The challenge in designing such a solution is that we must assure that only the user can activate threshold signing sessions even if the user device has been compromised (i.e., the secrets on it have been exposed). If we do not impose any restriction on the adversarial capability, an adversary having compromised a user device can always perfectly impersonate the user. Therefore, we assume that an adversary can compromise a user device only when the user software is inactive or the password is not in the device memory to which the adversary has access (such an assumption seems reasonable and popular [4, 23, 25, 27]).

3.1 How to Construct a HTSig

Given $\text{Sig}=(\text{Sig}.\text{Init}, \text{Sig}.\text{Sig}, \text{Sig}.\text{Ver})$ that belongs to the abovementioned subclass of signature schemes, we show how to construct a hybrid threshold signature scheme $\text{HTSig}=(\text{HTSig}.\text{Init}, \text{HTSig}.\text{Sig}, \text{HTSig}.\text{Ver})$ from its two-party variant $\text{2Sig}=(\text{2Sig}.\text{Init}, \text{2Sig}.\text{Sig}, \text{2Sig}.\text{Ver})$ and multi-party variant

$\text{TSig} = (\text{TSig}.\text{Init}, \text{TSig}.\text{Sig}, \text{TSig}.\text{Ver})$. Suppose (Y, X) is a user U 's pair of public and private keys in Sig . Let S_1, \dots, S_n be the servers, and t be the number of tolerated corrupt servers.

The initialization algorithm $\text{HTSig}.\text{Init}$ has the following two steps.

1. U uses a 2-out-of-2 secret sharing to share her private key X , namely $X \xleftarrow{(2,2)} (X_U, X_S)$, where X_U is U 's share. This step corresponds to $\text{2Sig}.\text{Init}$.
2. U uses an appropriate secret sharing scheme to share X_S , namely $X_S \xleftarrow{(t+1,n)} (X_S^{(1)}, \dots, X_S^{(n)})$, where $X_S^{(i)}$ ($1 \leq i \leq n$) is S_i 's share. This step corresponds to $\text{TSig}.\text{Init}$.

The signing algorithm $\text{HTSig}.\text{Sig}$ has the following steps that correspond to the combination of $\text{2Sig}.\text{Sig}$ and $\text{TSig}.\text{Sig}$. Suppose m is the message that needs to be signed.

1. U generates a partial signature $\sigma_U = g'_1(m, X_U)$, where g'_1 is an appropriate algorithm (which depends on the underlying signature scheme Sig).
2. S_{i_j} contributes its partial signature $\sigma_S^{(i_j)} = g_1(m, X_S^{(i_j)})$, where g_1 is an algorithm that depends on Sig , $1 \leq i_j \leq n$ for $1 \leq j \leq w$, and $t+1 \leq w \leq n$.
3. Given w valid partial signatures, anyone can compute $\sigma_S = g_2(\sigma_S^{(i_1)}, \dots, \sigma_S^{(i_w)})$, where $1 \leq i_j \leq n$ for $1 \leq j \leq w$, $t+1 \leq w \leq n$, and g_2 is an appropriate algorithm that depends on Sig .
4. Given σ_U and σ_S , anyone can compute a signature $\sigma = g'_2(\sigma_U, \sigma_S)$, where g'_2 is an appropriate algorithm that depends on Sig .

The verification algorithm $\text{HTSig}.\text{Ver}$ is the same as $\text{Sig}.\text{Ver}$.

SECURITY OF HTSig. Security of a hybrid threshold signature scheme HTSig is captured by the following definition.

Definition 1. (Security of HTSig ; Informal Statement.) A hybrid threshold signature scheme HTSig is secure, if it is unforgeable and robust. A HTSig is unforgeable if there exists a polynomial-time simulator that is able to emulate HTSig while having access to a signing oracle corresponding to the underlying signature scheme Sig . More specifically, we require the existence of such a simulator in the following two cases.

1. X_U is exposed (i.e., the user device is compromised) but at most t servers are compromised (i.e., X_S is still secret).
2. X_S is exposed (i.e., at least $t+1$ servers are compromised) but X_U is secret.

A HTSig is robust if it remains to function in the presence of t corrupt servers.

INSTANTIATIONS OF HTSig. We only investigate and sketch three HTSig schemes based on RSA, DSS, and Schnorr, respectively. An efficient HTSig can be based on RSA signature scheme [31, 5]. Suppose $N = PQ$ is a RSA modulus, $\langle e, N \rangle$ and $\langle d, N \rangle$ are a pair of RSA public and private keys so that $ed = 1 \pmod{\phi(N)}$. A HTSig can be obtained as follows: the user first splits the secret exponent d

into two pieces d_U and d_S (as in [25]), then the user shares d_S among the servers as in [30]. A signature is generated in a natural way.¹

A HTSig can be based on DSS [11] by combining the schemes in [26, 16], although the resultant scheme is not very efficient. A much more efficient HTSig based on Schnorr [33] can be obtained by combining the schemes in [38, 18].

3.2 The Construction

We present a TPAKE-HTSig based on any TPAKE that satisfies the specification in Appendix A, and any HTSig that satisfies the specification in Section 3.1. The *glue* or *middleware* for integrating them together is a natural use of the on-the-fly session keys generated by the TPAKE, namely the keys are used as message authentication keys.

THE INITIALIZATION. This protocol consists of TPAKE.Init and HTSig.Init.

1. A user U and the servers $\{S_1, \dots, S_n\}$ execute TPAKE.Init. (The basic functionality is that U sends certain transformed version of her password to the servers.)
2. A user U and the servers $\{S_1, \dots, S_n\}$ execute HTSig.Init.
 - (a) U executes $X \xleftarrow{(2,2)} (X_U, X_S)$ and stores X_U on her device.
 - (b) U executes $X_S \xleftarrow{(t+1,n)} (X_S^{(1)}, \dots, X_S^{(n)})$, and S_j ($1 \leq j \leq n$) stores $X_S^{(j)}$.

THE INVOCATION. Let m be the message a user U wants to sign.

1. U initiates TPAKE.Login with a set of servers $(S_{i_1}, \dots, S_{i_w})$. As a result, U holds w different on-the-fly session keys $sk_{i_1}, \dots, sk_{i_w}$ that are also known to S_{i_1}, \dots, S_{i_w} , respectively.
2. U and $(S_{i_1}, \dots, S_{i_w})$ execute as follows. (This step is a combination of HTSig.Sig and the *middleware* – the application of the session keys.)
 - (a) U generates a partial signature $\sigma_U = g'_1(m, X_U)$.
 - (b) U sends $(m, \delta = f_{sk_{i_j}}(m))$ to server S_{i_j} , where $1 \leq j \leq w$ and f is a secure message authentication code.
 - (c) S_{i_j} , where $1 \leq j \leq w$, verifies the integrity of the received (m, δ) . If $\delta \neq f_{sk_{i_j}}(m)$, then it aborts; otherwise, it returns its partial signature $\sigma_S^{(i_j)} = g_1(m, X_S^{i_j})$ back to the user.
 - (d) Given w valid partial signatures $\sigma_S^{(i_1)}, \dots, \sigma_S^{(i_w)}$, U computes $\sigma_S = g_2(\sigma_S^{(i_1)}, \dots, \sigma_S^{(i_w)})$.
 - (e) Given σ_U and σ_S , U computes the signature $\sigma = g'_2(\sigma_U, \sigma_S)$.

¹ For real-world deployment, we have to ensure that there are no honest server that is unable to contribute its partial signatures in the signature generation process of Rabin [30]. This is so because that [30] recovers the secret share held by a dishonest server, which means that the share held by (for instance) a server that is under denial-of-service attack or temporarily down will be recovered, in spite of the fact that this server is never compromised. We are grateful to a reviewer for pointing out that Shoup's scheme [35] cannot be used in this setting.

DISCUSSION 1. Any TPAKE satisfying the specification in Appendix A, and any HTSig satisfying the specification in Section 3.1, can be used to construct a secure TPAKE-HTSig (namely *plug-and-play*).

2. If HTSig is instantiated with RSA [30] (i.e., all the servers need to be involved in THE INVOCATION) and TPAKE is instantiated with the scheme of [27], then another layer of activation is needed (e.g., the $t + 1$ servers that have authenticated a user activate the rest servers). We omit this issue in our security analysis, which nonetheless can be extended to take it into account.

3.3 Security Analysis

We focus on the unforgeability of TPAKE-HTSig; it is relatively easy to see that robustness of TPAKE-HTSig is ensured by the robustness of TPAKE and of HTSig.

Theorem. Assume the underlying HTSig and message authentication schemes are secure. If TPAKE-HTSig is broken, then TPAKE is broken. (A formal treatment is deferred to Appendix B.)

4 The Second Scheme: LW-TSig

Our second scheme LW-TSig, which stands for “Light-Weight server-assisted Threshold Signatures,” is tailored for the following real-world scenario: A user, who has a smart card (or a hard-token in general) with no cryptographic co-processor, wants to enjoy threshold protection of her private signing function. This scheme is a simple, but useful, combination of a threshold signature scheme TSig and a message authentication mechanism.

4.1 The Construction

THE INITIALIZATION. This process involves a user U (who has a pair of public and private keys (Y, X)) and the servers $\{S_1, \dots, S_n\}$, and is done in a secure environment (e.g., on a secure computer). After the process is finished, U stores (key_1, \dots, key_n) on her smart card, and S_j ($1 \leq j \leq n$) holds (key_j, X_j) .

1. U runs TSig.**Init** via executing $X \xleftrightarrow{(t+1,n)} (X_1, \dots, X_n)$, where X_j ($1 \leq j \leq n$) is S_j ’s share.
2. U chooses and secretly sends a symmetric key key_j to S_j , where $1 \leq j \leq n$.

THE INVOCATION. This process involves a user U , a signature receiver, and the set of servers. Suppose U needs to sign a message m for the receiver.

1. U generates $\{\delta_j = f_{key_j}(m)\}_{j=1}^n$ and sends it to the signature receiver, where f is a secure message authentication code.
2. The signature receiver forwards (m, δ_j) to server S_j , where $1 \leq j \leq n$.
3. Server S_j ($1 \leq j \leq n$) checks whether the request (m, δ_j) is valid using the key key_j . If the request passes the test, S_j participates in the threshold signature generation protocol TSig.**Sig** (and perhaps sends its partial signature back to the signature receiver).
4. The signature receiver obtains a signature that can be verified using TSig.**Ver**.

4.2 Security Analysis

Theorem (Informal Statement). If both the message authentication scheme and the threshold signature scheme **TSig** are secure, then **LW-TSig** is secure.

5 A Taxonomy of Systems Protecting Signing Functions

5.1 The Terms

User storage-media. This specifies the type(s) of media a user used to store her secrets, and typically reflects the user’s budget.

1. **HUMAN-MEMORY.** Human memory is used to remember passwords.
2. **SOFT-TOKEN.** A soft-token is a data structure that is stored on a networked device – typically a computer that can efficiently compute cryptographic operations like modular exponentiations. A soft-token may be further protected by a password.
3. **HARD-TOKEN.** A hard-token is a special-purpose hardware like smart card, to which an adversary may not have access. It may have a cryptographic co-processor, be tamper-resistant, and be protected with a password.
4. **SOFT- & HARD-TOKEN.** A user holds a soft-token (stored on a networked device) as well as a hard-token. One or both of them may be protected using a password.

Number of runtime key-shares. When the private signing function is active, the private key may exist in its entirety or be shared among multiple parties. A private key may exist in the form of shares before it is applied, as per [34]. Suppose at runtime a private key is of n -piece, then we consider three cases: $n = 1$ (i.e., **ONE**), $n = 2$ (i.e., **Two**), and $n > 2$ (i.e., **MULTIPLE**).

5.2 The Taxonomy

We classify systems protecting private signing functions based on two factors: **User storage-media** – the x -axis, and **Number of runtime key-shares** – the y -axis. Let a coordinate (x, y) denote the corresponding system where $x \in \{\text{HUMAN-MEMORY}, \text{SOFT-TOKEN}, \text{HARD-TOKEN}, \text{SOFT- \& HARD-TOKEN}\}$, and $y \in \{\text{ONE}, \text{Two}, \text{MULTIPLE}\}$. Now we discuss the features of each of the 12 types of systems.

1. (**HUMAN-MEMORY, ONE**). In such a system, a user remembers a password which help her download her private key from a remote server [29] or remote servers [14, 22, 27]. Typically, in the transmission process the private key is protected using an on-the-fly session key generated using a secure password-authenticated key exchange protocol [7, 4, 23, 14, 27]. Note that such a system has a single point of failure.
2. (**HUMAN-MEMORY, Two**). There are further two types of systems. First, a user remembers a password from which a piece of her private key is derived, whereas the other piece of her private key is stored on a remote server.

- A typical scenario has been mentioned in [15, 6], and it has also been mentioned that such a system is subject to off-line dictionary attack. Second, a user remembers a password whereby he can activate two remote servers to collaboratively sign messages on her behalf. This is indeed a downsized (i.e., there are 2 remote servers) version of (HUMAN-MEMORY, MULTIPLE) below.
3. (HUMAN-MEMORY, MULTIPLE). There are also two types of systems. First, a user remembers a password from which a piece of her private key is derived, the other pieces of the private key are stored on remote servers. Typically, such a system is subject to off-line dictionary attack. Second, a user remembers a password whereby he can activate multiple remote servers to collaboratively sign messages on her behalf. Such a scheme is indeed a special case of our scheme **TPAKE-HTSig**.
 4. (SOFT-TOKEN, ONE). There are two types of systems. First, a user stores a password-protected private key on a device. Such a system is subject to off-line dictionary attack once the device is compromised. (Note that the variant introduced in [19] can prevent an off-line dictionary attack, but at the price of keeping the corresponding *public* key secret.) Second, a user deploys a mechanism called *forward security*, which guarantees that compromise of a private key at time t will not expose the private keys used at any past time $t' < t$. This notion was introduced in [1], and formalized in [3] which is followed by numerous papers (e.g., [24, 20]).
 5. (SOFT-TOKEN, Two). In such a system, a private key is split into two pieces such that one is stored on the user device as a soft-token, and the other is stored on a remote server. The authentication of a user to the remote server may be based on a password. The tailored constructions [9, 25, 8] fall into this category.
 6. (SOFT-TOKEN, MULTIPLE). Our scheme **TPAKE-HTSig** is such a system, where a user splits her private key X into two pieces: X_U and X_S . The user stores her piece X_U on her networked device, and shares X_S among the remote servers.
 7. (HARD-TOKEN, ONE). This is the traditional solution where a private key is stored on a (tamper-resistant) smart card with a cryptographic co-processor. Clearly, *forward security* can be incorporated.
 8. (HARD-TOKEN, TWO). There are further two systems. First, a user stores one share of her private key on her smart card, and the other on a remote server. The smart card or the share on it may be protected by a password. In such a system, the smart card is typically equipped with a cryptographic co-processor, which we want to avoid. Second, a user stores certain symmetric keys on her smart card, and shares her private key among two remote servers that collaborate in generating signatures. This is indeed a down-sized version of our scheme **LW-TSig**.
 9. (HARD-TOKEN, MULTIPLE). Our scheme **LW-TSig** is such a system, where a user's private key is shared among a set of servers and a smart card is used to store some symmetric keys. The authentication of the user to the servers are based on symmetric key cryptography, so that the smart card does not

- need any cryptographic co-processor. Note that a password may be further deployed to protect a smart card.
10. (SOFT- & HARD-TOKEN, ONE). The most recently introduced system [12, 21] fall into this category. The basic idea underlying *key insulation* [12] is that the system life time is divided into time periods, and a smart card is used only for updating private keys corresponding to a fixed public key. As a consequence, compromise of the user device, even at the runtime of the user software, will not compromise the private key corresponding to any past or future time period. An adversary compromising the smart card alone is still unable to generate any signatures that are valid with respect to the user's public key. The enhanced notion of *intrusion resilience* [21] assures *forward security* even if both the user device and smart card are compromised simultaneously. Note that a password may be used to protect the smart card.
 11. (SOFT- & HARD-TOKEN, Two). In such a system, a user device and smart card collaborate via a two-party signature scheme [6, 26, 38].
 12. (SOFT- & HARD-TOKEN, MULTIPLE). Such a system could be seen as an extension of our schemes TPAKE-HTSig or LW-TSig, because the soft-token can hold a share of her private key (as per TPAKE-HTSig) and the smart card can hold key_1, \dots, key_n (as per LW-TSig). There are some interesting variants. For example, the smart card keeps the X_U in TPAKE-HTSig and the key_1, \dots, key_n in LW-TSig. Whenever the user needs to sign messages, she takes advantage of any (perhaps public) terminal to compute the partial signature with respect to X_U (which is of course given to the terminal and erased by the terminal after the session is finished), but the invocation of the remote servers are authenticated by the smart card (i.e., key_1, \dots, key_n never depart the smart card).

6 Conclusion

We presented two efficient and provably secure schemes for server-assisted threshold signatures, where the signing function is activated by a user (but using some enhanced authentication). The first one is tailored for the setting where the user device is able to efficiently compute modular exponentiations, and the second one is tailored for the setting where a user has a smart card without a cryptographic co-processor. We also presented a taxonomy of systems protecting private signing functions.

Acknowledgements

We are very grateful to the reviewers of RSA-CT'03 for their excellent comments that significantly improved this paper. We also thank Gene Tsudik for helpful comments.

References

- [1] R. Anderson. Invited Lecture. ACM CCS'97. [363](#)
- [2] M. Bellare, J. Kilian, and P. Rogaway. The Security of Cipher Block Chaining. Crypto'94. [356](#)
- [3] M. Bellare and S. Miner. A Forward-Secure Digital Signature Scheme. Crypto'99. [363](#)
- [4] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. Eurocrypt'2000. [358](#), [362](#), [366](#)
- [5] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. Eurocrypt'96. [357](#), [359](#)
- [6] M. Bellare and R. Sandhu. The Security of Practical Two-Party RSA Signature Schemes. manuscript. 2001. [357](#), [363](#), [364](#)
- [7] S. Bellovin and M. Merritt. Encrypted Key Exchange: Password-based Protocols Secure against Dictionary Attack. IEEE Security and Privacy'92. [362](#)
- [8] D. Boneh, X. Ding, G. Tsudik, and C. Wong. A Method for Fast Revocation of Public Key Certificates and Security Capabilities. Usenix Security'01. [363](#)
- [9] C. Boyd. Digital Multisignatures. Cryptography and Coding, pp 241-246, 1989. [363](#)
- [10] R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. IEEE FOCS'01. [356](#)
- [11] The US Digital Signature Standard, NIST, 1994. [357](#), [360](#)
- [12] Y. Dodis, J. Katz, S. Xu, and M. Yung. Strong Key-Insulated Signature Schemes. PKC'03, to appear. [364](#)
- [13] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. IEEE FOCS'87. [357](#)
- [14] W. Ford and B. Kaliski. Server-Assisted Generation of a Strong Secret from a Password. IEEE Workshops on Enabling Technologies'00. [362](#)
- [15] R. Ganesan. Yaksha: Augmenting Kerberos with Public Key Cryptography. NDSS'95. [363](#)
- [16] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. Eurocrypt'96. [357](#), [360](#)
- [17] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. SIAM J. Comput., (17)2, 1988, pp 281-308. [357](#)
- [18] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Public Key and Signature Schemes. ACM CCS'97. [357](#), [360](#)
- [19] D. Hoover and B. Kausik. Software Smart Cards via Cryptographic Camouflage. IEEE Security and Privacy'99. [363](#)
- [20] G. Itkis and L. Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. Crypto'01. [363](#)
- [21] G. Itkis and L. Reyzin. SiBIR: Signer-Based Intrusion-Resilient Signatures. Crypto'02. [364](#)
- [22] D. Jablon. Password Authentication using Multiple Servers. RSA-CT'01. [362](#)
- [23] J. Katz, R. Ostrovsky, and M. Yung. Practical Password-Authenticated Key Exchange Provably Secure under Standard Assumptions. Eurocrypt'01. [358](#), [362](#), [366](#)
- [24] H. Krawczyk. Simple Forward-Secure Signatures from any Signature Schemes. ACM CCS'00. [363](#)

- [25] P. MacKenzie and M. Reiter. Networked Cryptographic Devices Resilient to Capture. IEEE Security and Privacy'01. [357](#), [358](#), [360](#), [363](#)
- [26] P. MacKenzie and M. Reiter. Two-Party Generation of DSA Signatures. Crypto'01. [357](#), [360](#), [364](#)
- [27] P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold Password-Authenticated Key Exchange. Crypto'02. [358](#), [361](#), [362](#), [366](#)
- [28] T. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. Crypto'91. [357](#)
- [29] R. Perlman and C. Kaufman. Secure Password-based Protocol for Downloading a Private Key. NDSS'99. [362](#)
- [30] T. Rabin. A Simplified Approach to Threshold and Proactive RSA. Crypto'98. [357](#), [360](#), [361](#)
- [31] R. A. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. C. ACM. (21)2, 1978, pp 120-126. [357](#), [359](#)
- [32] R. Sandhu, M. Bellare, and Ravi Ganesan. Password-Enabled PKI: Virtual Smart-cards versus Virtual Soft Tokens. PKI Research Workshop. 2002.
- [33] C. P. Schnorr. Efficient Signature Generation by Smart Cards. J. Cryptology, 1991. [357](#), [360](#)
- [34] A. Shamir. How to Share a Secret. C. ACM, 22(11):612–613, 1979. [357](#), [362](#)
- [35] V. Shoup. Practical Threshold Signatures. Eurocrypt'00. [357](#), [360](#)
- [36] J. Stern, D. Pointcheval, J. Malone-Lee, and N. Smart. Flaws in Applying Proof Methodologies to Signature Schemes. Crypto'02. [357](#)
- [37] S. Xu and M. Yung. On the Dark Side of Threshold Cryptography. Financial Crypto'02.
- [38] S. Xu and M. Yung. A Provably Secure Two-Party Schnorr Signature Scheme. manuscript. 2002. [357](#), [360](#), [364](#)

A A Specification of TPAKE

Instead of pinning down on any concrete construction, we give a specification of TPAKE so that any concrete construction satisfying this specification can be seamlessly embedded into our TPAKE-HTSig, namely *plug-and-play*. This specification is extended from the model of [27], which in turn builds on [4, 23].

PARTICIPANTS. There are two types of protocol participants: users and servers. Let $ID = \text{Users} \cup \text{Servers}$ be a non-empty set of protocol participants (i.e., principals), where $\text{Servers} = \{S_1, \dots, S_n\}$. Each user $U \in \text{Users}$ has a secret password pwd_U , and each server $S \in \text{Servers}$ has a vector $pwd_S = [pwd_S(U)]_{U \in \text{Users}}$, where entry $pwd_S(U)$ is a password record. Let $Password$ be the dictionary from which the users' passwords are uniformly chosen (but the results can be easily extended to other password distributions). Let $D = |Password|$, which is typically small in the real world.

EXECUTION OF THE PROTOCOL. A protocol TPAKE (i.e., TPAKE.Login) is an algorithm that determines how principals behave in response to their environment. In the real world, each principal $P \in ID$ (modeled as a probabilistic polynomial-time algorithm) is able to execute TPAKE multiple times with different partners,

and we model this by allowing unlimited number of instances of each principal. Instance i of principal P is denoted by Π_P^i .

To capture the security of TPAKE, we assume there is an adversary \mathcal{A} that has complete control over the environment (mainly, the network), and thus provides the inputs to instances of principals. Since \mathcal{A} controls the network and can deny service at any time, we do not consider any denial-of-service attack (as long as it has no security implication). We further assume the network (i.e., \mathcal{A}) performs aggregation and broadcast functions. In practice, on a point-to-point network, the protocol implementer would most likely have to implement these functionalities in some way, perhaps using a single intermediate (untrusted) node to aggregate and broadcast messages. Formally, the adversary is a probabilistic algorithm with a distinguished query tape. We assume that \mathcal{A} may compromise up to t servers, and that the choice of these servers is static. In particular, we may assume that the choice is made before the initialization of TPAKE, and we may simply assume that the adversary has access to the secrets of the compromised servers. Queries written to the query tape are responded to by principals according to TPAKE. The adversary \mathcal{A} can request the following queries.

1. COMPROMISE(P). This results in that the secrets on $P \in ID$ are returned to \mathcal{A} . Note that (1) if \mathcal{A} requests COMPROMISE(U) then U 's password is not returned to \mathcal{A} , and (2) at most t servers are compromised.
2. SEND(P, i, M). This results in that message M is sent to instance Π_P^i , which then executes according to TPAKE (including the update of its internal state) and returns the result to \mathcal{A} . If this query causes Π_P^i to accept or terminate, this will also be shown to \mathcal{A} .
3. EXECUTE($U, i, ((S_{j_1}, l_{j_1}), \dots, (S_{j_w}, l_{j_w}))$), where $1 \leq j_z \leq n$ and $l_{j_z} \in \mathbb{N}$ for $1 \leq z \leq w$, $t + 1 \leq w \leq n$. This results in the execution of TPAKE between Π_U^i ($U \in Users$) and $(\Pi_{S_{j_1}}^{l_{j_1}}, \dots, \Pi_{S_{j_w}}^{l_{j_w}})$. The transcript is returned to \mathcal{A} .
4. REVEAL(U, i, S_j). This results in that the session key held by Π_U^i corresponding to server S_j (i.e., $sk_{\Pi_U^i, S_j}$) is returned to \mathcal{A} .
5. REVEAL(S_j, i). This results in that the session key held by $\Pi_{S_j}^i$ (i.e., $sk_{\Pi_{S_j}^i}$) is returned to \mathcal{A} .
6. TEST(U, i, S_j). To response to this query, Π_U^i flips a coin b , and returns $sk_{\Pi_U^i, S_j}$ if $b = 0$, and a string that is drawn uniformly from the same space otherwise. A TEST query (of either type) may be asked at any time during the execution of TPAKE, but may be asked only once.
7. TEST(S_j, i). To response to this query, $\Pi_{S_j}^i$ flips a coin, and returns $sk_{\Pi_{S_j}^i}$ if $b = 0$, and a string that is drawn uniformly from the same space otherwise. A TEST query (of either type) may be asked at any time during the execution of TPAKE, but may be asked only once.

PARTNERING USING SIDs. Let sid be the concatenation of all messages sent and received by the user instance in its communication with the set of servers. (Note that this excludes the messages that are sent only between the servers, but not to the user.) A server instance that accepts holds a partner-id pid , session-id sid , and a session key sk . A user instance that accepts

holds a session-id sid , a partner-id $pid = (pid_1, \dots, pid_w)$ and the corresponding session keys (sk_1, \dots, sk_w) . The instance Π_U^i ($U \in Users$) holding $(sid, (pid_1, \dots, pid_w), (sk_1, \dots, sk_w))$ and $\Pi_{S_j}^{l_j}$ ($S_j \in Servers$) holding (sid', pid', sk) are said to be partnered if there exists a unique $1 \leq z \leq w$ such that $pid_z = S_j$, $pid' = U$, $sid = sid'$, and $sk_z = sk$.

FRESHNESS. A user instance/server pair (Π_U^i, S_j) is fresh if: (1) there has been no COMPROMISE(S_j) query, (2) there has been no REVEAL(U, i, S_j) query, and (3) if $\Pi_{S_j}^l$ is a partner of Π_U^i , there has been no REVEAL(S_j, l) query. A server instance $\Pi_{S_j}^i$ is fresh if (1) there has been no COMPROMISE(S_j) query, (2) there has been no REVEAL(S_j, i) query, and (3) if Π_U^l is the partner to $\Pi_{S_j}^i$, there has been no REVEAL(U, l, S_j) query.

ADVANTAGE OF AN ADVERSARY \mathcal{A} IN TPAKE. The goal of the adversary \mathcal{A} is to guess the bit b . To formalize its advantage, let $\text{Succ}^{\text{TPAKE}}(\mathcal{A})$ be the event that \mathcal{A} makes a single TEST query directed to some instance Π_P^i ($P \in ID$) that has terminated and is fresh, and eventually outputs a bit $b' = b$. The advantage of \mathcal{A} attacking TPAKE is defined to be

$$\text{Adv}^{\text{TPAKE}}(\mathcal{A}) = 2 \cdot \Pr[\text{Succ}^{\text{TPAKE}}(\mathcal{A})] - 1.$$

We say that \mathcal{A} breaks TPAKE, if $\text{Adv}^{\text{TPAKE}}(\mathcal{A})$ is non-negligibly more than $\frac{q_{send}}{D}$, where q_{send} is the number of SEND(P, i, M) queries. *Security* and *robustness* requirements of a TPAKE is captured by the following definition.

Definition 2. (Security and robustness of TPAKE) *A TPAKE is said to be secure if for any probabilistic polynomial-time algorithm \mathcal{A} , $\text{Adv}^{\text{TPAKE}}(\mathcal{A}) = \frac{q_{send}}{D} + \epsilon(\kappa^*)$, where κ^* is a security parameter, q_{send} is the number of SEND(P, i, M) queries, D is the size of the dictionary from which the users choose their individual passwords, and $\epsilon(\cdot)$ is a negligible function. A TPAKE is said to be robust if compromise of t servers doesn't prevent it from functioning (i.e., outputting session keys).*

B Security Analysis of TPAKE-HTSig

B.1 Security Definition of TPAKE-HTSig

In order to capture the *security* (including *unforgeability* and *robustness*) of TPAKE-HTSig, we extend the above specification of TPAKE to capture the capability of an adversary \mathcal{F} that intends to forge signatures in TPAKE-HTSig.

PARTICIPANTS. This is the same as in the specification of TPAKE, except for that each user $U \in Users$ has a pair of public and private keys (Y, X) corresponding to an appropriate signature scheme Sig .

EXECUTION OF THE PROTOCOL. A protocol TPAKE-HTSig is an algorithm that determines how principals behave in response to their environment. In the real world, each principal $P \in ID$ (modeled as a probabilistic polynomial-time algorithm) is able to execute TPAKE-HTSig multiple times with different partners,

and we model this by allowing unlimited number of instances of each principal. Instance i of principal P is denoted by Π_P^i .

The adversary \mathcal{F} in TPAKE-HTSig is similar to the adversary \mathcal{A} in TPAKE, except for that \mathcal{F} can only request the following queries.

1. COMPROMISE(P). This results in that the secrets on $P \in ID$ are returned to \mathcal{F} . Note that (1) if \mathcal{F} requests COMPROMISE(U) then U 's password is not returned to \mathcal{F} , and (2) at most t servers are compromised.
2. SEND(P, i, M). This results in that message M is sent to instance Π_P^i , which then executes according to TPAKE-HTSig (including the update of its internal state) and returns the result to \mathcal{F} .
3. EXECUTE($U, i, ((S_{j_1}, l_{j_1}), \dots, (S_{j_w}, l_{j_w}), m)$), where $1 \leq j_z \leq n$ and $l_{j_z} \in \mathbb{N}$ for $1 \leq z \leq w$, $t + 1 \leq w \leq n$. This results in the execution of TPAKE-HTSig between Π_U^i ($U \in Users$) and $(\Pi_{S_{j_1}}^{l_{j_1}}, \dots, \Pi_{S_{j_w}}^{l_{j_w}})$. The runtime transcript is returned to \mathcal{F} .
4. REVEAL(U, i, S_j). This results in that the session key held by Π_U^i corresponding to server S_j (i.e., $sk_{\Pi_U^i, S_j}$) is returned to \mathcal{F} .
5. REVEAL(S_j, i). This results in that the session key held by $\Pi_{S_j}^i$ (i.e., $sk_{\Pi_{S_j}^i}$) is returned to \mathcal{F} .

PARTNERING USING SIDs. This is the same as in the specification of TPAKE.

FRESHNESS. This is the same as in the specification of TPAKE.

THE SUCCESS PROBABILITY OF THE ADVERSARY \mathcal{F} IN TPAKE-HTSig. Let $\text{Succ}_{\text{TPAKE-HTSig}}(\mathcal{F})$ be the event \mathcal{F} outputs a valid (with respect to U 's public key Y) signature σ on message m , while the following hold simultaneously:

1. there was no EXECUTE($U, i, ((S_{j_1}, l_{j_1}), \dots, (S_{j_w}, l_{j_w}), m)$ query;
2. if there is an un-compromised server S that ever outputs a partial signature on m , which means that an instance Π_S^l had received a message authentication tag $f_{sk}(m)$ that is valid with respect to the session key sk generated in a session of sid , then the following constraints apply:
 - (a) (Π_U^i, S) is fresh, where Π_S^l and Π_U^i are partners in session sid ;
 - (b) There was no oracle query to either Π_S^l or Π_U^i for generating message authentication tag on message m . (Note that our security analysis remains sound even though we allow this type of oracle query that may not be available in real-world systems.)

Denote $\Pr[\mathcal{F} \text{ Succ}]$ the probability that $\text{Succ}_{\text{TPAKE-HTSig}}(\mathcal{F})$ happens. We say that \mathcal{F} breaks TPAKE-HTSig, if $\Pr[\mathcal{F} \text{ Succ}]$ is non-negligibly more than $\frac{q_{send}}{D}$, where q_{send} is the number of SEND(P, i, M) queries.

We say that a TPAKE-HTSig is *robust* if compromise of t servers doesn't prevent it from functioning (i.e., outputting signatures).

B.2 The Theorems

Theorem 1. *Assume the underlying hybrid threshold signature scheme HTSig and message authentication scheme are secure. If there exists a probabilistic*

polynomial-time adversary \mathcal{F} that breaks TPAKE-HTSig with probability $\Pr[\mathcal{F} \text{ Succ}]$ that is non-negligibly more than $\frac{q_{send}}{D}$, then there exists a probabilistic polynomial-time algorithm \mathcal{A} such that $\text{Adv}^{\text{TPAKE}}(\mathcal{A})$ is non-negligibly more than $\frac{q_{send}}{D}$.

Proof. We assume that \mathcal{F} is static and compromises servers S_1, \dots, S_t at the system initialization. We stress that either X_U or pwd_U , but not both, can be compromised.

There are two ways for \mathcal{F} to forge signatures. First, it forges a message authentication tag $f_{sk}(m)$ with respect to some secret session key sk_j that is generated on-the-fly and held by an un-compromised server S_j , where $t+1 \leq j \leq n$. (This happens, for instance, when \mathcal{F} succeeds in hitting U 's password and therefore knows the on-the-fly message authentication key.) As a consequence, \mathcal{F} that already knew X_U is able to get a signature by requesting the un-compromised server S_j to contribute a partial signature. Denote by “ \mathcal{F} Succ-AU” the event that \mathcal{F} succeeds in forging a message authentication tag that is accepted by an un-compromised server S_j , and by $\Pr[\mathcal{F} \text{ Succ-AU}]$ the probability that this event happens. Second, it forges a signature without forging any message authentication tag. This implies that \mathcal{F} is able to output a signature by having access to: (1) $X_S^{(1)}, \dots, X_S^{(t)}$, and perhaps X_U , or (2) X_S which is obtained by compromising a quorum number of servers.

Note that

$$\begin{aligned} \Pr[\mathcal{F} \text{ Succ}] &= \Pr[\mathcal{F} \text{ Succ} | \mathcal{F} \text{ Succ-AU}] \cdot \Pr[\mathcal{F} \text{ Succ-AU}] + \\ &\quad \Pr[\mathcal{F} \text{ Succ} | \neg \mathcal{F} \text{ Succ-AU}] \cdot \Pr[\neg \mathcal{F} \text{ Succ-AU}] \\ &= \Pr[\mathcal{F} \text{ Succ-AU}] + \Pr[\mathcal{F} \text{ Succ} | \neg \mathcal{F} \text{ Succ-AU}] \cdot \Pr[\neg \mathcal{F} \text{ Succ-AU}]. \end{aligned}$$

In order to prove the theorem, we prove two lemmas. By Lemma 1 we show $\Pr[\mathcal{F} \text{ Succ} | \neg \mathcal{F} \text{ Succ-AU}]$ is negligible, which means the probability that \mathcal{F} succeeds in forging a signature without forging a message authentication tag is negligible. According to the assumption that $\Pr[\mathcal{F} \text{ Succ}]$ is non-negligibly more than $\frac{q_{send}}{D}$, we know that $\Pr[\mathcal{F} \text{ Succ-AU}]$ is non-negligibly more than $\frac{q_{send}}{D}$. By Lemma 2 we show that if $\Pr[\mathcal{F} \text{ Succ-AU}]$ is non-negligibly more than $\frac{q_{send}}{D}$, then there is a probabilistic polynomial-time algorithm \mathcal{A} that is able to break TPAKE with probability non-negligibly more than $\frac{q_{send}}{D}$. Therefore, the theorem holds.

Lemma 1. *If HTSig is secure, then $\Pr[\mathcal{F} \text{ Succ} | \neg \mathcal{F} \text{ Succ-AU}]$ is negligible.*

Proof. Suppose $\Pr[\mathcal{F} \text{ Succ} | \neg \mathcal{F} \text{ Succ-AU}]$ is non-negligible, then we show that there is a probabilistic polynomial-time algorithm \mathcal{F}' that is able to break HTSig with non-negligible probability. According to Definition 1, we know that there is a simulator that is able to emulate HTSig by having access to a centralized signing oracle in the underlying signature scheme Sig . The basic idea underlying the proof of this lemma is that \mathcal{F}' maintains a TPAKE-HTSig environment by establishing a TPAKE sub-environment that is beyond the scope of HTSig. Specifically, \mathcal{F}' executes as follows.

1. \mathcal{F}' emulates THE INITIALIZATION of TPAKE-HTSig as follows.
 - (a) \mathcal{F}' executes the first step of THE INITIALIZATION in TPAKE-HTSig; namely, TPAKE.**Init**. As a consequence, \mathcal{F}' knows all the configurations of the servers' and the passwords of the users'.
 - (b) \mathcal{F}' emulates the second step of THE INITIALIZATION in TPAKE-HTSig; namely, HTSig.**Init**. The simulation in the case that X_U is compromised typically differs from the simulation in the case that X_S is compromised, but the existence of such simulators is guaranteed.
2. \mathcal{F}' emulates THE INVOCATION of TPAKE-HTSig as follows.
 - (a) Suppose X_U has been compromised.
 - i. \mathcal{F}' executes TPAKE.**Login** on behalf of the servers. If an authentication session is successfully finished, then \mathcal{F}' holds the corresponding message authentication keys.
 - ii. The user sends a signing request to \mathcal{F}' .
 - iii. If the signing request is valid, \mathcal{F}' returns the simulated partial signatures corresponding to the servers chosen by the user. This is done as if \mathcal{F}' is simulating a multi-party signature scheme TSig by having oracle access to a signing oracle.
 - iv. The user computes the signature after obtaining w valid partial signatures from \mathcal{F}' , where $t + 1 \leq w \leq n$.
 - (b) Suppose X_S has been compromised. In this case, \mathcal{F}' is indeed the simulator that emulates the two-party signature scheme 2Sig corresponding to the underlying signature scheme Sig.

Since \mathcal{F} does not forge any message authentication tag, all of the signatures that \mathcal{F}' has obtained are also available to the adversary \mathcal{F} . As a consequence, any forgery by \mathcal{F} in the simulated TPAKE-HTSig is a forgery by \mathcal{F}' in HTSig.

Lemma 2. *Suppose the message authentication code is secure. If $\Pr[\mathcal{F} \text{ Succ-AU}]$ is non-negligibly more than $\frac{q_{\text{send}}}{D}$, then there exists a probabilistic polynomial-time algorithm \mathcal{A} that breaks TPAKE with probability non-negligible more than $\frac{q_{\text{send}}}{D}$.*

Proof. The basic idea underlying the proof of this lemma is that \mathcal{A} maintains a TPAKE-HTSig environment while having access to a TPAKE sub-environment. In particular, \mathcal{A} maintains a HTSig sub-environment that is beyond the TPAKE.

For notational reason, let SID be the set of the sid 's that correspond to those sessions whereby a non-null session key is defined in each of them. We can then define a total order on the elements of SID . Denote by $N = |SID|$, then we assume that each element in SID can be identified by $i \in \{1, \dots, N\}$.

Now we present the algorithm \mathcal{A} .

1. \mathcal{A} chooses a pair of public and private keys (Y, X) and executes the second step of THE INITIALIZATION in TPAKE-HTSig (to share X among the user and the servers).

2. \mathcal{A} chooses a random $z \in \{1, \dots, N\}$ corresponding to a session between $\Pi_{U'}^i$ and $\{S_{j_1}, \dots, S_{j_w}\}$, where $1 \leq j_z \leq n$ for $1 \leq z \leq w$, $t+1 \leq w \leq n$. \mathcal{A} uniformly chooses $j \in \{j_1, \dots, j_w\} \setminus \{1, \dots, t\}$ and executes $\text{TEST}(U, i, S_j)$ in the TPAKE sub-environment. The environment flips a coin b , and returns $sk = sk_{\Pi_{U'}^i, S_j}$ if $b = 0$, and a random string otherwise. For each of the other session keys, \mathcal{A} requests a REVEAL to get it. Moreover, \mathcal{A} answers \mathcal{F} 's queries in the emulated TPAKE-HTSig as follows.
- For a query $\text{COMPROMISE}(P)$, \mathcal{A} returns the corresponding secrets to \mathcal{F} .
 - For a query $\text{SEND}(P, i, M')$, there are two cases.
 - If the query is for TPAKE, \mathcal{A} forwards $\text{SEND}(P, i, M)$ to the TPAKE sub-environment and returns its response to \mathcal{F} , where M is well-defined due to the modular composition of TPAKE-HTSig.
 - If the query is for HTSig, \mathcal{A} executes according to the specification of TPAKE-HTSig and returns the result back to \mathcal{F} .
 - For a query $\text{EXECUTE}(U, i, ((S_{j_1}, l_{j_1}), \dots, (S_{j_w}, l_{j_w}), m)$, where $1 \leq j_z \leq n$ and $l_{j_z} \in \mathbb{N}$ for $1 \leq z \leq w$, \mathcal{A} forwards $\text{EXECUTE}(U, i, ((S_{j_1}, l_{j_1}), \dots, (S_{j_w}, l_{j_w})))$ to the TPAKE sub-environment, and emulates the runtime beyond TPAKE. \mathcal{A} returns the response from the TPAKE sub-environment and the emulated transcript that is beyond the TPAKE back to \mathcal{F} .
 - For a query $\text{REVEAL}(U, i, S_j)$, \mathcal{A} can answer it with probability at least $1 - \frac{1}{(n-t)N}$ since it has obtained the corresponding session key.
 - For a query $\text{REVEAL}(S_j, i)$, \mathcal{A} can answer it with probability at least $1 - \frac{1}{(n-t)N}$ since it has obtained the corresponding session key.

Now, if \mathcal{F} succeeds in forging a message authentication tag that is valid with respect to the key sk returned by the $\text{TEST}(\cdot, \cdot, \cdot)$ oracle in the TPAKE sub-environment, then \mathcal{A} returns 0 (meaning that it bets $sk = sk_{\Pi_{U'}^i, S_j}$ or $b = 0$); otherwise, \mathcal{A} returns a random bit.

We claim that if $b = 1$, which means that the sk obtained from the $\text{TEST}(\cdot, \cdot, \cdot)$ oracle is a random string, then \mathcal{F} has negligible probability in forging a valid message authentication tag with respect to $f_{sk}(\cdot)$. Otherwise, the message authentication scheme is broken. Therefore, $\Pr[\mathcal{F} \text{ Succ-AU}|b = 1]$ is negligible.

Now we compute the probability that \mathcal{A} successfully hits the bit b when \mathcal{F} succeeds in forging a message authentication tag; namely,

$$\begin{aligned} \Pr[\mathcal{A} \text{ Succ}|\mathcal{F} \text{ Succ-AU}] &= \Pr[b = 0|\mathcal{F} \text{ Succ-AU}] \\ &= \frac{\Pr[b = 0] \cdot \Pr[\mathcal{F} \text{ Succ-AU}|b = 0]}{\Pr[\mathcal{F} \text{ Succ-AU}]} \end{aligned}$$

Therefore, we have

$$\begin{aligned} \Pr[\mathcal{A} \text{ Succ}] &= \Pr[\mathcal{A} \text{ Succ}|\mathcal{F} \text{ Succ-AU}] \cdot \Pr[\mathcal{F} \text{ Succ-AU}] + \\ &\quad \Pr[\mathcal{A} \text{ Succ}|\neg\mathcal{F} \text{ Succ-AU}] \cdot \Pr[\neg\mathcal{F} \text{ Succ-AU}] \\ &= \Pr[b = 0] \cdot \Pr[\mathcal{F} \text{ Succ-AU}|b = 0] + 0.5(1 - \Pr[\mathcal{F} \text{ Succ-AU}]) \\ &= \Pr[\mathcal{F} \text{ Succ-AU}] - \Pr[b = 1] \cdot \Pr[\mathcal{F} \text{ Succ-AU}|b = 1] + \\ &\quad 0.5(1 - \Pr[\mathcal{F} \text{ Succ-AU}]) \\ &= 0.5(1 + \Pr[\mathcal{F} \text{ Succ-AU}] - \Pr[\mathcal{F} \text{ Succ-AU}|b = 1]) \end{aligned}$$

and

$$\begin{aligned}\text{Adv}^{\text{TPAKE}}(\mathcal{A}) &= 2 \cdot \Pr[\mathcal{A} \text{ Succ}] - 1 \\ &= \Pr[\mathcal{F} \text{ Succ-AU}] - \Pr[\mathcal{F} \text{ Succ-AU} | b = 1],\end{aligned}$$

which is non-negligibly more than $\frac{q_{send}}{D}$.

As a final comment, we note that the tight reduction in security comes from the fact that the simulator is of expected polynomial-time (with an expected slowdown factor no greater than $(n - t)N$).

Secure Applications of Pedersen’s Distributed Key Generation Protocol

Rosario Gennaro¹, Stanisław Jarecki², Hugo Krawczyk¹, and Tal Rabin¹

¹ IBM T.J. Watson Research

² Stanford University

Abstract. A Distributed Key Generation (DKG) protocol is an essential component of any threshold cryptosystem. It is used to initialize the cryptosystem and generate its private and public keys, and it is used as a subprotocol, for example to generate a one-time key pair which is a part of any threshold El-Gamal-like signature scheme. Gennaro et al. showed [GJKR99] that a widely-known non-interactive DKG protocol suggested by Pedersen does not guarantee a uniformly random distribution of generated secret keys even in the static adversary model. Furthermore, Gennaro et al. proposed to replace this protocol with one that guarantees a uniform distribution of the generated key but requires an extra round of reliable broadcast communication.

We investigate the question whether some discrete-log based threshold cryptosystems remain secure when implemented using the more efficient DKG protocol of Pedersen, in spite of the fact that the adversary can skew the distribution of the secret key generated by this protocol. We answer this question in the positive. We show that threshold versions of some schemes whose security reduces to the hardness of the discrete logarithm problem, remain secure when implemented with Pedersen DKG. We exemplify this claim with a threshold Schnorr signature scheme.

However, the resulting scheme has less efficient security reduction (in the random oracle model) from the hardness of the discrete logarithm problem than the same scheme implemented with the computationally more expensive DKG protocol of Gennaro et al. Thus our results imply a trade-off in the design of threshold versions of certain discrete-log based schemes between the round complexity of a protocol and the size of the modulus.

Keywords: Threshold cryptography, distributed key generation, discrete logarithm, exact security, random oracle model.

1 Introduction

Distributed Key Generation in Threshold Cryptosystems. Distributed key generation is a main component of threshold cryptosystems ([Des87, DF89]). It allows a set of n servers, a.k.a. “players”, to jointly generate a pair of public and private keys in a such a way that the public key is output in the clear while the private key is shared by the n servers via a threshold secret-sharing scheme [Sha79].

Unless an adversary compromises more than a specified threshold, e.g., $t < n/2$, out of the n servers, the generated private key remains secret and its secret-sharing can be subsequently used by these servers to jointly compute signatures or decryptions, or perform any other functionality required of the secret key in a cryptosystem. For discrete-log based threshold schemes, distributed key generation amounts to a distributed generation (i.e. without a trusted dealer) of a Shamir secret sharing [Sha79] of a random value x , and making public the value $y = g^x$. Following [GJKR99], we refer to such a protocol as DKG.

Torben Pedersen in [Ped91b] proposed a simple DKG protocol. Pedersen's protocol was then used in many discrete-log based threshold cryptosystems, e.g., [CMI93, Har94, LHL94, GJKR96, HJJ⁺97, PK96, CGS97, SG98, CG99]. It is important to point out that the DKG protocol in some of these threshold cryptosystems is used as a subprotocol every time a signature or decryption needs to be computed. For example, in all threshold implementations of ElGamal-like signatures schemes (e.g., [CMI93, Har94, LHL94, GJKR96, HJJ⁺97, PK96]), the servers need to generate a secret-sharing of a temporary secret k and a public value $r = g^k$ every time they generate a signature. They accomplish this with an instance of the Pedersen's DKG protocol.

Requirement of Uniform Distribution of the Private Key. All the above threshold cryptosystems implicitly assume that the DKG protocol generates the private and public keys with uniform distribution over their prescribed domain. In a *centralized*, i.e. standard, version of any discrete-log based scheme the secret key x is generated uniformly at random in a group Z_q for prime q . Similarly, the proposed *threshold* versions of these schemes assume that the distributed key generation protocol they employ generates the private key uniformly in Z_q . (Alternatively, the key in both cases is chosen uniformly in a group Z_{p-1} for prime p .)

Indeed, it seems necessary that the distribution of the secret key must be the same in the threshold and the centralized case if one attempts to argue the security of the threshold cryptosystem with a reduction from the security of the underlying centralized cryptosystem. For example, to argue that a threshold DSS scheme is as secure as the centralized (standard) DSS scheme, one needs to show that a forgery of a signature under the public key generated by the threshold DSS scheme implies the ability to forge signatures in an attack on a standard DSS scheme, i.e., to forge a signature under the public key generated uniformly at random. It is not clear how to argue this if the public keys generated by the threshold DSS scheme and the standard DSS scheme are chosen from different probability distributions.

However, Gennaro et al. in [GJKR99] showed that Pedersen's DKG protocol fails to generate the secret key (and the public key) with uniform distribution. They showed that an adversary who compromises even two out of n servers can skew the distribution of the generated secret key. While it is not clear if the adversary can control the distribution of the private key in a way that helps him break the cryptosystem that uses this key, the adversary's ability to skew this distribution from uniform means that the above security reduction argu-

ment does not hold. Since such reduction was the proof technique employed (most often implicitly) to argue security of existing discrete-log based threshold cryptosystems, Gennaro et al. proposed a new DKG protocol which fixes this problem by generating the private key with a uniform distribution.¹ When the DKG protocol of Gennaro et al. is substituted for Pedersen's DKG in a threshold cryptosystem whose security proof assumes uniform distribution of the generated secret – which is the case of all the discrete-log based cryptosystems mentioned above – this substitution renders the threshold cryptosystem provably secure.

Problem: Cost of the DKG Protocol. Unfortunately, the DKG protocol of Gennaro et al. is twice more expensive than the original DKG protocol by Pedersen. While Pedersen's protocol has one round of communication in the absence of faults, the protocol of Gennaro et al. requires two rounds of communication. Moreover, each communication round involves a reliable broadcast, which is a costly operation in a realistic setting like the internet (see e.g. [CP02]). The new DKG protocol also requires about twice more computation from each server.

The computation and communication cost of the DKG protocol is important because for threshold cryptosystems which have the most efficient threshold versions, for example in the threshold version of Schnorr's signature scheme, the cost of the DKG protocol is a primary factor in the cost of the scheme. If a threshold scheme is designed to handle on-line requests, and/or the number of requests is large, an increase in the communication and computation costs of the scheme by a factor of two implies a considerable expense. Reducing the cost of this scheme is furthermore worthwhile because with off-line preprocessing it is more efficient than a threshold RSA scheme (see e.g. [Sho00] and the efficiency discussions therein).

Our Contribution: Dealing with Non-Uniform Distribution on Keys. We show that a certain type of discrete-log based threshold schemes can remain secure even if it uses Pedersen's DKG protocol as a subprotocol, notwithstanding the fact that this protocol indeed does not generate secrets with uniform distribution. Namely, we show how to prove secure a threshold version of Schnorr's signature scheme [Sch89] instantiated with Pedersen's DKG protocol.

We show that this threshold scheme is secure by exhibiting a *direct* reduction of its security to the hardness of the underlying computational problem. In other words, rather than reducing the security of a threshold signature scheme to the security of the centralized version of this signature scheme, we prove its security “from scratch”. Our proofs work because, as it turns out, even though the adversary has some control over the generated public key, we can still embed an instance of the underlying hard computational problem into the part of the

¹ Another DKG protocol which also fixes the problem of Pedersen's protocol, and which moreover can be used to construct in a generic fashion secure threshold versions of discrete-log based schemes, was proposed by Frankel et al. in [FMY99]. The DKG protocols proposed by Gennaro et al. and Frankel et al. are similar in spirit but the proposal of Gennaro et al. is more efficient.

public key which is contributed by the uncorrupted players in the Pedersen's DKG protocol. We then show how to translate a successful forgery under the resulting public key into solving the embedded instance of a hard problem.

In this way we avoid the limitations of the existing proof techniques which argue security of threshold signature schemes by exhibiting a reduction to the centralized version of the same signature scheme, which, as we argued above, seems hard to do for threshold schemes which use Pedersen's DKG protocol. Indeed, for this reason our methodology does not apply to threshold version of schemes for which no known reduction to some underlying hard problem exists. Thus for example it will be difficult to use our techniques to imply security of threshold DSS or ElGamal signatures implemented with Pedersen's DKG.

However, it is very likely that our methodology can be applied to showing security of other threshold discrete-log based cryptosystems using the less expensive Pedersen's DKG protocol, if the security of the original centralized versions of these schemes can be reduced to the discrete logarithm assumption, the computational Diffie-Hellman, or some other hardness assumption.

Implications of our Result: Cost vs. Exact Security The security proof we exhibit for the threshold Schnorr signature scheme implemented with Pedersen's DKG has one important drawback: the security reduction to the underlying hard problem is less efficient than the security reduction that exist for the centralized version of this scheme. If q_H is the number of hash function queries made by the adversary who breaks the threshold implementation of this signature scheme with probability ϵ , then the discrete logarithm problem can be solved in comparable time with probability only $\epsilon^2/(q_H)^2$. This is a factor of q_H degradation of security compared to the centralized version of this scheme, for which a successful forgery with probability ϵ implies computation of the discrete log in comparable time with probability ϵ^2/q_H (by the result of Pointcheval and Stern [PS96]). In contrast to the protocol we propose, the same threshold scheme implemented with the DKG protocol of Gennaro et al. has the same exact security as the centralized scheme, i.e. it is also related by a ϵ^2/q_H factor to the security of discrete log.

Since q_H should be upper-bounded by the total number of operations performed by the adversary, the q_H degradation in the security reduction implies that we need to square the bound on the resources allowed to the adversary.² Similarly, the q_H^2 security degradation means that we need to cube the adversary's bound. Since the difficulty of computing discrete logarithms in a q -order subgroup of field F_p grows as $\exp(|q|/2)$ and $\exp(|p|^{1/3})$, squaring the amount of time available to the adversary implies having to use twice longer q and $2^3 = 8$ times longer p . Similarly, cubing the amount of adversary's time implies three times longer q and $3^3 = 27$ times longer p . Because the cost of exponentiation grows at least as $O(|q| \cdot |p|^{1.6})$, the proportion between the computational cost of a threshold Schnorr scheme implemented with Pedersen DKG and the compu-

² See Section 5 for a more detailed explanation of the consequences of security degradation in reductions.

tational cost of a threshold Schnorr scheme implemented with the DKG scheme of [GJKR99] is $1/2 \cdot (3^{1+3*1.6}/2^{1+3*1.6}) \approx 5$, if the two schemes are to achieve the same security guarantee based on the discrete logarithm assumption (the $1/2$ factor is due to the fact that Pedersen DKG requires about twice fewer cryptographic operations per player than the DKG of [GJKR99]). Since both schemes require longer moduli than customary to achieve provable security based on the DL assumption alone, the computational cost of either scheme is comparable to the communication delay incurred by one broadcast round which, in the recent implementation of [CP02], takes $100ms$ in the LAN setting and about $1s$ in the internet setting, for a group of about five to nine players. Comparing the two costs (see Section 5 for the detailed comparison), we conclude that even though it requires one more round of reliable broadcast, the threshold Schnorr protocol implemented with DKG of [GJKR99] is still more efficient than the same protocol implemented with Pedersen DKG, *provided that* the two schemes are to achieve provably same guarantee of security based on the discrete-log assumption.

We point out, however, that the mere existence of a polynomial reduction from some scheme to a discrete logarithm problem can be a *heuristic* argument suggesting that the security of the two problems is similar. If one believes in this heuristics, then both Schnorr's signatures, and the threshold Schnorr signatures built using [GJKR99] DKG, *and* the threshold Schnorr signatures built using Pedersen DKG, can be implemented using the same field in which the discrete logarithm problem is believed to be hard for modern computers, e.g. it has the 2^{80} security bound. In that case, the threshold signature scheme using Pedersen DKG will be twice more efficient than the same scheme using the DKG of [GJKR99], because the cost of broadcast will dominate the delay incurred by running the threshold scheme.

Organization: In Section 2 we summarize the communication and adversarial models and the definition of security for the threshold cryptosystems we consider. In Section 3 we recall Pedersen's DKG protocol and we give some intuition for why this protocol is good enough for proving security of certain threshold schemes. In Section 4 we recall a simple threshold version of Schnorr's signature scheme using Pedersen's DKG protocol, and we prove its security. In Section 6 we conclude with the discussion of costs vs. security tradeoffs, other potential applications of the method presented here, and some open problems.

2 Preliminaries

Computation, Communication, and Adversarial Model. We work in the standard model for threshold signature schemes. For a comprehensive overview of the models for threshold cryptography we refer the reader to [Jar01].

The computation proceeds among a set of n *players* P_1, \dots, P_n modelled by probabilistic polynomial-time Turing machines, and an adversary \mathcal{A} , also modelled as a PPT TM, who submits the messages of his choice to be signed. We assume that the players are connected by a complete network of private (i.e. untappable) and authenticated point-to-point channels because link encryption and

authentication can be achieved in this model by standard cryptographic techniques. In addition, the players have access to a dedicated broadcast channel, which can be implemented for example by [CP02].

We assume for simplicity that the initial input of all players is a specification of a discrete-log scheme chosen by a trusted third party, i.e. a triple (p, q, g) where p and q are primes and g is a generator of subgroup G_q of order q in Z_p^* . Such a triple can be publicly chosen without a trusted party via Bach's algorithm [Bac85].

We consider threshold signature schemes secure in adaptive chosen-message attack [GMR88]. The computation proceeds by the n players first executing a DKG protocol to compute a secret-sharing of some private key $x \in Z_q$ and a corresponding public key $y = g^x \bmod p$. Then every time the adversary requests a signature on some message of his choice, the players perform the threshold signature protocol on that message.

In addition to requesting signatures on adaptively chosen messages, the adversary can corrupt up to t of the n players in the network, for any value of $t < n/2$, which is the best achievable threshold. The adversary can cause the corrupted players to arbitrarily divert from the specified protocol. We assume that the adversary is *static*, i.e. that he chooses the corrupted players at the beginning of the protocol.

We assume a realistic *partially synchronous* communication model, i.e. that computation proceeds in synchronized rounds and that the messages are received by their recipients within some specified time bound. To guarantee this round synchronization, and for simplicity of discussion, we assume that the players are equipped with synchronized clocks. Notice that in this model messages sent from the uncorrupted players to the corrupted ones can still be delivered relatively fast, in which case, in every round of communication, the adversary can wait for the messages of the uncorrupted players to arrive, then decide on his computation and communication for that round, and still get his messages delivered to the honest parties on time. Therefore we should always assume the worst case that the adversary speaks last in every communication round. In the cryptographic protocols literature this is also known as a *rushing* adversary.

Negligible Probability. We call function f *negligible* if for every polynomial $P(\cdot)$, $f(k) \leq (1/P(k))$ for all sufficiently large k . We say that some event occurs with a *negligible probability* if the probability of this event is a negligible function of the security parameter k .

Assumption 1 (Discrete Log Assumption). Let $\text{PRIMES}(k)$ be the set of $\text{poly}(k)$ -bit primes p such that there exists a k -bit prime q dividing $p - 1$. For every probabilistic polynomial time algorithm I , for every p in $\text{PRIMES}(k)$, probability $\Pr[g \leftarrow G_q; x \leftarrow Z_q; I(p, q, g, g^x) = x]$ is negligible, where G_q is the subgroup of Z_p^* of order q .

Notion of Security for a Threshold Signature Scheme. We call a threshold signature scheme *secure* with adversarial threshold t , if it is both *robust* and *unforgeable*. A threshold scheme is robust if in the presence of a threshold

adversary, the threshold signature protocol produces a valid signature on the requested message, except for at most negligible probability. A scheme is unforgeable if, after an execution of the distributed key generation protocol which produces some public key, and after participating as a threshold adversary in an execution of a polynomial number of runs of the threshold signature protocol on the messages of adversary's choice, the adversary's probability of forgery, i.e. of producing a valid signature under the generated public key on some new message, is negligible.

Simulation Proof Technique. We will argue unforgeability of a threshold signature scheme using the standard technique of *simulation* of adversary's view in the distributed computation. Namely, to prove that the scheme is unforgeable based on some cryptographic assumption, say the above discrete log assumption, we will exhibit an efficient algorithm called a *simulator*, which on the input of a random instance of the hard problem, say an instance of the discrete logarithm problem, can simulate to the adversary his view of an execution of the threshold protocol in such a way that (1) this view is indistinguishable from a view of an actual random execution of the threshold protocol; and (2) if the adversary succeeds in forging a signature under the generated public key then the simulator can translate this forgery into solving the input instance of the hard problem. An existence of such a simulator algorithm concludes the proof because if the adversary has a non negligible probability of forgery against the threshold scheme then, by the simulation property (1), he will produce such a forgery during the simulation. But then, by the simulation property (2), the simulator will be able to solve the input instance of the supposedly hard problem.

Random Oracle Model. Our proofs of security are in so-called Random Oracle Model [BR93], i.e. we model a hash function like MD5 or SHA1 as ideal random oracles in the sense that if an adversary who makes some number of queries to the fixed hash function has some (non negligible) probability of success in, for example, computing a forgery against a threshold signature scheme, the same adversary should have the same probability of success if instead of a fixed hash function like MD5 or SHA1, he accesses a truly random function.

Notation. All arithmetic operations in this paper are performed in some finite group or field, in fact all operations are modulo some prime, either q or p . We will often omit writing $\text{mod } p$ or $\text{mod } q$ to simplify the notation, but the proper modulus should always be clear from the context: operations on secret values chosen in Z_q , like x and k , are modulo q , while all modular exponentiation operations and operations on public values like y or r are computed modulo p .

3 Pedersen's Distributed Key Generation Protocol

In Pedersen's DKG protocol [Ped91b], each player shares a randomly chosen secret x_i using Feldman's verifiable secret sharing (VSS) protocol [Fel87]. The secret value x is the sum of the properly shared x_i 's. Since Feldman's VSS has the additional property of revealing $y_i = g^{x_i}$, the public value $y = g^x$ is the

Distributed Key Generation Protocol Ped-DKG

1. Each player P_i chooses a random polynomial $f_i(z)$ over Z_q of degree t :

$$f_i(z) = a_{i0} + a_{i1}z + \dots + a_{it}z^t$$

P_i broadcasts $X_{ik} = g^{a_{ik}} \bmod p$ for $k = 0, \dots, t$. Denote a_{i0} by x_i and X_{i0} by y_i . Each P_i computes the shares $\bar{x}_{ij} = f_i(j) \bmod q$ for $j = 1, \dots, n$ and sends \bar{x}_{ij} secretly to player P_j .

2. Each P_j verifies the shares he received from the other players by checking for $i = 1, \dots, n$:

$$g^{\bar{x}_{ij}} = \prod_{k=0}^t (X_{ik})^{j^k} \bmod p \quad (1)$$

If the check fails for an index i , P_j broadcasts a *complaint* against P_i .

3. If more than t players complain against a player P_i , then that player is clearly faulty and he is disqualified. Otherwise P_i reveals the share \bar{x}_{ij} matching Eq. 1 for each complaining player P_j . If any of the revealed shares fails this equation, P_i is disqualified. By convention, the secret shared by a disqualified player P_i is always set to $x_i = 0$ and $y_i = 1$.
4. The public value y is computed as $y = \prod y_i \bmod p$. The secret shared value x itself is not computed by any party, but it is equal to $x = \sum x_i \bmod q$. Each player P_i keeps all the values it received during these n parallel Fel-VSS runs for the purpose of verification in subsequent secret reconstruction.

Fig. 1. Pedersen's Distributed Key Generation protocol

product of the y_i 's that correspond to those properly shared x_i 's. We present this protocol, denoted Ped-DKG, in Figure 1.

We will require that the reconstruction of the secret x shared in a Ped-DKG protocol, proceeds via n parallel reconstructions of the Fel-VSS protocol for all shared secrets x_i , and then x is computed as $x_1 + \dots + x_n \bmod q$. Note that the secret can also be reconstructed by every player P_i submitting his *polynomial* share of x , $\bar{x}_i = \sum_j \bar{x}_{ji}$. This is indeed a standard reconstruction procedure in threshold protocols. However, our proof of security requires the “additive” rather than “polynomial” secret reconstruction. Note that the two reconstruction procedures have similar cost both in the absence of faults and with faults.

The robustness property of the Ped-DKG protocol follows from robustness of Fel-VSS. The secret x is uniquely defined at the end of the protocol and no malicious behavior of up to $t < n/2$ servers can prevent its reconstruction.

Secrecy Property of Ped-DKG. The secrecy property of the Ped-DKG protocol is somewhat murky. Namely, it is not clear if the adversary learns anything more in some useful sense about the shared value x than is revealed by its public counterpart $y = g^x$. As Gennaro et al. [GJKR99] point out, a standard

formalization of a secrecy property fails, because the adversary controls to some extent the distribution of public key y 's output by this protocol, and hence it is impossible to simulate this protocol on input y chosen uniformly in G_q .

As Gennaro et al. observe, the adversary can, for example, cause this protocol to output only even y 's as follows. Assume wlog that the adversary corrupts players P_1, \dots, P_t . Call those "bad" players and call the remaining players "good". Let $x_G = x_{t+1} + \dots + x_n$ be the contribution of the good players to secret x , and let x_B be the summary contribution of the bad players to x , i.e. $x_B = \sum_{P_i \in \{1, \dots, t\}} x_i$. Then secret x shared in the DKG protocol is $x = x_G + x_B$. Because the adversary can always speak last (see Section 2), he can choose his contribution x_B after seeing $y_G = \prod_{P_i \in \{t+1, \dots, n\}} y_i = g^{x_G}$ in step 1 of the DKG protocol. Consequently, he is free to choose x_B so that the resulting value $y = y_G * y_B = y_G * g^{x_B}$, has some property that he wants to achieve. For example, we can see that by trying out at random potential values for x_B , a polynomial-time adversary can fully control $O(\ln k)$ bits of y , where k is the security parameter.

However, even though the adversary has some control over the resulting y , and hence the secrecy of the Ped-DKG protocol is unclear, it turns out that for the purpose of proving security of threshold cryptosystems which use the Ped-DKG protocol, the following "secrecy-like" property of this protocol is good enough: The public key y it produces is a product of y_G chosen with uniform distribution by the good players, and of value $y_B = g^{x_B}$, where x_B is the contribution of the bad players into the computation. Because this contribution x_B can be reconstructed by the good players via interpolation, we can show that under the discrete log assumption, for any polynomial-sized set of values $Y \subset G_q$, the adversary cannot choose his contribution x_B in such a way that the resulting $y = y_G g^{x_B}$ belongs to set Y . If he did, then let $y_{max} \in Y$ be the value that he is most likely to hit. On input a random y_T in G_q , a simulator playing on behalf of the good players chooses their contribution as $y_G = y_{max}/y_T$. Since y_T is random in G_q , so is y_G . Moreover, by the same argument that one uses to argue secrecy of Feldman's VSS protocol, the simulator can perfectly simulate the adversary's view of the contribution of the good players to any value y_G which is uniformly distributed in G_q . Then, if the adversary creates x_B s.t. $y = y_G g^{x_B} = y_{max}$ then $x_B = dlog_g(y_T)$, and hence the simulator would solve the discrete log problem on a random instance y_T .

In other words, even though the simulator cannot control the chosen value y , the adversary cannot control it too much either. In particular, he has only negligible chance of making y fall in any polynomially-sized set. In the next section we will see how this property enables simulation of certain threshold signature schemes that use the Ped-DKG protocol to create the commitment in the three-round zero-knowledge identification scheme.

4 Threshold Schnorr Signature Scheme Using Ped-DKG

Schnorr’s Signature Scheme. We first recall Schnorr’s signature scheme [Sch89]. As before, let p, q be primes and let g be a generator of subgroup G_q of order q in \mathbb{Z}_p^* . Let H be a hash function, $H : \{0, 1\}^* \rightarrow Z_q$, which we will model as an ideal random function. The private key is x , chosen at random in Z_q . The public key is $y = g^x \bmod p$. A signature on message m is computed as follows. The signer picks a one-time secret k at random in Z_q , and computes the signature on m as a pair (c, s) where $s = k + cx \bmod q$, $c = H(m, r)$, and $r = g^k \bmod p$. Signature (c, s) can be publicly verified by computing $r = g^s y^{-c} \bmod p$ and then checking if $c = H(m, r)$.

This signature scheme follows a methodology introduced by Fiat and Shamir [FS86], which converts any three-round *commit-challenge-response* zero-knowledge identification scheme where the challenge is a public coin into a signature scheme. This is done by replacing the random challenge chosen by the verifier with an output of a random function H computed on the message and the prover’s commitment. In this case the prover’s commitment is $r = g^k$, the challenge is $c = H(m, r)$, and the prover’s response is $s = k + cx$. In the random oracle model, the unforgeability of this scheme under a chosen-message attack [GMR88] reduces to the discrete log assumption, as proven by [PS96].

We recall this proof here because it helps in understanding the proof of security of the *threshold* Schnorr signature scheme below. The simulator, on input y , can produce Schnorr’s signatures on any m by picking s and c at random in Z_q , computing $r = g^s y^{-c}$ and setting $H(m, r) = c$. This simulator can also translate the adversary’s forgery into computing $dlog_y$ as follows. It runs the adversary until the adversary outputs a forgery (c, s) on some message m . Note that because H is a random function, except for negligible probability, the adversary must ask H a query (m, r) where $r = g^s y^{-c}$, because otherwise it could not have guessed the value of $c = H(m, r)$. The simulator then rewinds the adversary, runs it again by giving the same answers to queries of H until the query (m, r) , which it now answers with new randomness c' . If the adversary forges a signature on m in this run, then, except for negligible probability, it produces s' s.t. $r = g^{s'} y^{-c'}$, and hence the simulator can now compute $dlog_y = (s - s')/(c' - c)$. One can show that if the adversary’s probability of forgery is ϵ , this simulation succeeds with probability $\epsilon^2/4q_H$: $O(\epsilon)$ probability that the adversary forges in the first run times the $O(\epsilon/q_H)$ probability that it will forge on the second run *and* that it will choose to forge on the same (m, r) query out of its q_H queries to H . We refer to [PS96] for the full proof.

Threshold Version of Schnorr’s Scheme Using Pedersen’s DKG. The threshold version of Schnorr’s scheme using Pedersen’s DKG protocol Ped-DKG is a very simple protocol. It works in the straightforward way as all standard threshold dlog-based protocols, e.g. [GJKR96], except that secret reconstruction is done on additive rather than polynomial shares.³

³ Replacing computation on polynomial shares with computation on additive shares often seems necessary for the proof of security of a threshold protocol to go through.

Threshold Signature Protocol TSch

Inputs: Message m to be signed, plus the secret-sharing of x generated by the initial Ped-DKG protocol. In particular, each player P_i holds an additive share x_i of x while values $y = g^x$ and $y_i = g^{x_i}$ for every P_i are public.

Outputs: Schnorr's signature (c, s) on m .

1. Players perform an instance of the Ped-DKG protocol (Figure 1). Denote the outputs of this run of Ped-DKG as follows. Each player P_i holds an additive share k_i of the secret-shared secret k . Each of these additive shares is itself secret-shared with Fel-VSS. We denote the generated public values $r = g^k$ and $r_i = g^{k_i}$ for each P_i .
2. Each player locally computes the challenge $c = H(m, r)$.
3. Players perform the reconstruction phase of Feldman's secret-sharing of value $s = k + cx$ as follows. Each player P_i broadcasts its additive share $s_i = k_i + cx_i$. Each share is verified by checking if $g^{s_i} = r_i y_i^c$. Otherwise x_i and k_i are reconstructed and s_i is computed publicly. The protocol outputs signature (c, s) where $s = s_1 + \dots + s_n$.

Fig. 2. Threshold Signature Protocol for Schnorr's Signature Scheme

To initialize the threshold signature scheme, first the Ped-DKG protocol is executed for distributed key generation, i.e. it outputs secret-sharing of a private key x and a public value, the public key $y = g^x$. Then the threshold Schnorr signature protocol TSch proceeds as follows on input a message m (see Figure 2). First the players run an instance of the Ped-DKG protocol to generate a secret-sharing of the one time secret k and the public value $r = g^k$. Then each player locally computes the challenge $c = H(m, r)$, and each player P_i broadcasts its *additive* share $s_i = k_i + cx_i$, where k_i, x_i are P_i 's additive shares of k and x respectively. Notice that these shares can be publicly verified by checking if $g^{s_i} = K_{i0}(X_{i0})^c$, where $X_{i0} = y_i = g^{x_i}$ and $K_{i0} = r_i = g^{k_i}$ are verification values broadcasted by player P_i , during the initial key-generation Ped-DKG protocol, and the Ped-DKG protocol that generates r and the sharing of k , respectively. If verification fails for some P_i , the players reconstruct the Feldman secret-sharing of both x_i and k_i and compute s_i publicly. Finally, s is computed as $s = s_1 + \dots + s_n$. Therefore secret s can be efficiently reconstructed as in the Fel-VSS protocol.

Efficiency Considerations. Note that without faults, the above protocol requires only one round of broadcast during the Ped-DKG protocol of Step (1). Recall that broadcast is the most important factor in the delay incurred by threshold protocols in the internet setting. This is true if the verifier request-

This technique was used before to handle an adaptive adversary in [FMY99, CGJ⁺99] or to handle the no-erasure and concurrent adversary in [JL00].

ing the signature communicates directly with every player P_i . When P_i receives and validates a message m to be signed, it triggers the Ped-DKG protocol of Step 1 and broadcasts along it the message m . This allows the players to detect if the verifier submitted inconsistent requests to them. If no faults occur, there is only one more round of broadcast, in Step 3 of TSch, but this can be avoided if every player P_i sends to the verifier the value c and its share s_i . The verifier can check if signature $(c, s) = (c, s_1 + \dots + s_n)$ is valid, and the protocol falls back to the robust reconstruction only if the signature does not verify. The same protocol implemented with the DKG protocol of Gennaro et al. has two rounds of broadcast without faults. Note also that Step (1) of the protocol can be performed off-line, and thus with preprocessing the threshold Schnorr signature is non-interactive (without faults).

Security of Threshold Signature Scheme (Ped-DKG,TSch). The robustness of the (Ped-DKG,TSch) threshold signature scheme follows straightforwardly from the robustness of the Ped-DKG protocol. The interesting part is the proof of unforgeability.

Theorem 1. *Under the discrete log assumption, the threshold signature scheme (Ped-DKG, TSch) is unforgeable in the static adversarial model with adversarial threshold $t < n/2$, in the random oracle model.*

Proof: Assume that there exists an adversary which breaks the unforgeability property of this signature scheme. We'll construct a simulator **SIM** which, using this adversary, will compute a discrete logarithm on input a random challenge value y_T in G_q . In the course of this simulation, **SIM** answers adversary's queries to oracle H at random, except in a crucial case specified below. Let B be the set of corrupted (i.e. “bad”) players and G be the set of good players. Assume wlog that $P_n \in G$. Let q_H, q_S be the upper bound on the number of H queries and the number of the signature queries, respectively, that the adversary makes.

To compute discrete logarithm of a target value y_T , the simulator **SIM** embeds it in the value y_n contributed by player P_n to the public key generated in the initial Ped-DKG protocol. In other words, the simulator follows the Ped-DKG protocol on behalf of players $P_i \in G \setminus \{n\}$ as prescribed, but for player P_n the simulator *simulates* the adversarial view of the Fel-VSS protocol performed as a part of Ped-DKG by player P_n so that public value y_n broadcasted by P_n is equal to the target value y_T .

After simulating the initial key generation protocol in this way, **SIM**, for every message m submitted by the adversary for a signature, simulates an execution of TSch on this message as follows. **SIM** chooses at random in Z_q values c and s_n , computes $r_n = g^{s_n} y_n^{-c}$, and in Step (1) of TSch it follows the protocol on behalf of players $P_i \in G \setminus \{n\}$ as prescribed, but for P_n it again simulates the Fel-VSS protocol in this step so that the value r_n broadcasted by that player is what the simulator wants, i.e. $r_n = g^{s_n} y_n^{-c}$.

The simulator's goal is to make sure that whatever value r is computed in this step, the output of the H oracle on input (m, r) can be set to c . In that case the simulator can simulate the rest of the TSch protocol. The value $H(m, r)$

will be computed as c in Step (2), and in Step (3) the simulator can follow the protocol on behalf of players $P_i \in G \setminus \{n\}$ as prescribed, while for P_n it can publish the previously chosen value s_n , and since $g^{s_n} = r_n y_n^c$, this value passes. Moreover, this is a valid signature because $g^s = g^{\sum s_i} = \prod(g^{s_i}) = \prod(r_i y_i^c) = (\prod r_i)(\prod y_i)^c = ry^c$, and $c = H(m, r)$.

To extract with non-negligible probability the value c as the output of H on (m, r) where r is computed in Step (1) of TSch, the simulator **SIM**, after computing values $r_i \in G$ in the simulation of Step (1) as described above, but before sending any messages in this simulation to the adversary, flips a coin b . If $b = 0$, then during the simulation of Step (1) **SIM** answers all the (new) queries the adversary makes to H at random. If the value r computed in Step (1) is such that H was already asked on (m, r) , the simulator fails. Otherwise, **SIM** sets $H(m, r)$ as c and succeeds. If $b = 1$, then the simulator chooses at random an index $i \in \{1, \dots, q_H\}$ and if the adversary makes any $(m, r^{(j)})$ queries to H *after* the simulator publishes all values $y_i \in G$ but *before* the adversary shares his values $x_i \in B$ (these are sub-steps of the simulation of Step (1)), the simulator answers i -th such query with c . If value r output in this Step (1) is equal to $r^{(i)}$, this is simulator's success because $H(m, r) = c$. Otherwise the simulator fails.

We try to upper-bound the simulator's probability of success. We use the intuition outlined earlier in the discussion of security of the Ped-DKG protocol in Section 3. If the adversary chooses his contribution $r_B = \prod_{i \in B} r_i$ to r so that H was not asked on r , the simulator succeeds with probability $1/2$. Let ϵ_{hit} be the probability with which the adversary, after seeing $r_G = \prod_{i \in G} r_i$, chooses his contribution r_B so that it "hits" some value $r = r_{G \cap B}$ on which oracle H has been queried *before* the simulation of Step (1) starts. Therefore, if $b = 1$ and the output r of Step (1) is such that H has been queried (m, r) either before the simulation of this step or during this simulation, the probability that $r = r^{(i)}$ chosen by the simulator is upper bounded by $1/q_H - \epsilon_{hit}$, the probability that **SIM** guesses the right index i minus the probability that the adversary hits some value r on which the oracle H was queried before the simulation. Assuming that ϵ_{hit} is small, the probability that the simulator successfully passes the simulation of this run of the TSch protocol can be upper-bounded as $\epsilon_{ss} = 1/(2q_H) - \epsilon_{hit}$. Therefore, if **SIM** repeats this simulation c/ϵ_{ss} times, the probability that it fails is at most e^{-c} . In this way, with probability $(1 - 1/e^c)^{q_S}$, the simulator successfully goes through the simulation of each of the q_S instances of the TSch protocol. Setting $c = \ln 2q_S$, this probability is more than a half.

Since the adversary's view in this simulation is the same as in the protocol execution, then assuming that the adversary forges with non negligible probability ϵ , the simulator will get some forged signature (c, s) on some message m with probability $\epsilon/2$. By applying the same "forking lemma" argument as [PS96] used to prove the security of the standard Schnorr signature (see the beginning of this section), we can argue the following. **SIM**, with probability at least $\epsilon/4$ gets one forgery *and* if he re-winds this adversary to the point when the adversary asks query $H(m, r)$ where $r = g^s y^{-c}$, and then continues to simulate from that point on with fresh randomness (in particular answering this query to H with

fresh randomness c'), then **SIM** has about $(\epsilon/4)/q_H$ chance of getting a second forgery on the same message m but relative to a different random function H . In this case **SIM** gets two pairs (c, s) and (c', s') s.t. $r = g^s y^{-c} = g^{s'} y^{-c'}$ and thus can compute $dlog_g y$ and hence also $dlog_g y_n = dlog_g y_T$ because **SIM** knows all $x_i = dlog_g y_i$ for $P_i \neq P_n$.

Therefore, if the assumed threshold signature forger runs in time T and succeeds in forgery with probability ϵ , then if probability p is small, **SIM** computes the discrete logarithm in time $2c/\epsilon_{ss} * T = 2 \ln(2q_S)/\epsilon_{ss} * T$ with probability at least about $\epsilon^2/(16q_H)$. If T is big enough so that $\epsilon \approx 1$ then the simulator's computation is longer than the adversary's computation by the factor $\alpha = 16q_H * 2 \ln 2q_S/\epsilon_{ss}$. If $\epsilon_{hit} < 1/(4q_H)$, then $\epsilon_{ss} = 1/(2q_H) - \epsilon_{hit} > 1/(4q_H)$, and hence $\alpha < 2^7 q_H^2 \ln(2q_S)$. Taking $q_S < 2^{30}$, we get $\alpha < 2^{11.5} q_H^2$ factor of security degradation.

Now, if ϵ_{hit} is not small enough then we can solve discrete logarithm otherwise, following the argument in Section 3. Recall that ϵ_{hit} is the probability which with the adversary, after seeing $r_G = \prod_{i \in G} r_i$, chooses his contribution r_B so that it “hits” some value $r = r_{GRB}$ on which oracle H has been queried before the simulation of Step (1) starts. Note that there are at most q_H of such values. Let r_{max} be the value that the adversary is most likely to hit. On input a random y in G_q , the simulator chooses the contribution of the good players as $r_G = r_{max}/y$. Since y is random in G_q , so is r_G . The simulator can simulate an execution of Ped-DKG so that the good players' contribution is r_G by the secrecy property of Fel-VSS. Then, if the adversary does hit the value r_{max} , which happens with probability ϵ_{hit}/q_H , and supplies k_B s.t. $r = r_{GRB} = r_G g^{k_B} = r_{max}$ then $k_B = dlog_g(y)$, and hence the simulator solves the discrete log problem on a random instance y with probability ϵ_{hit}/q_H . Hence if $\epsilon_{hit} > 1/(4q_H)$, we can reduce this adversary to an attack on the discrete logarithm scheme with similar factor $4q_H^2$ of security degradation. \square

5 Security vs. Efficiency Implications

We showed that even though Pedersen's DKG protocol Ped-DKG does not generate secret keys with uniform distribution, it generates them randomly enough for us to show security for a threshold Schnorr signature protocol TSch implemented with Ped-DKG. This is an improvement because with previously known proof methods, discrete-log based schemes had to use the DKG protocol of Gennaro et al which requires two rounds of broadcast instead of one round incurred by Pedersen's DKG. This is especially important for a scheme like Schnorr's, whose main cost lies in the DKG subprotocol that it uses, and which is also the most efficient discrete-log based threshold signature scheme, so reducing its communication cost by half is worthwhile.⁴

⁴ In fact, the on-line part of the computation in threshold Schnorr signature is more efficient than in threshold RSA. On the other hand, threshold RSA can be fully non-interactive, so its overall cost, as opposed to on-line cost, is smaller.

However, the security reduction we are able to show for the TSch threshold Schnorr signature scheme has a q_H^2 factor of security degradation compared to the security of the discrete log problem [DLP]. This is a factor of q_H degradation over provable security of the centralized version of Schnorr signatures. Recall that Pointcheval-Stern [PS96] show a reduction from DLP to Schnorr signatures which has a q_H factor of security degradation. Because the security reduction from Schnorr signatures to their threshold version implemented with the DKG protocol of Gennaro et al (let's denote this threshold Schnorr scheme gjkr-TSCh) is tight, it follows that there is also a q_H factor in the degradation of (provable) security between the TSch scheme and the gjkr-TSCh scheme.

The degradation in the provable security can be interpreted in two ways. One can ignore it and take the mere existence of a polynomial reduction from some scheme to the DLP as a good coin, and claim that since the two problems are shown to be polynomially related, one can securely implement the scheme in question over a field in which DLP is believed to be hard. This is, however, only a heuristic argument. Formally, existence of a security reduction from some scheme to DLP with a degradation factor f implies that if one takes b as a target security bound – i.e. if one wants to claim that the constructed scheme is secure against an adversary performing about b operations – then one needs to use a group in which DLP is believed to be hard against an adversary performing about $b \cdot f$ operations. Therefore, the less efficient reduction for the TSch scheme means that the TSch scheme should be implemented over a larger field to guarantee the same b security bound. In fact, the increase of the computation time resulting from the fact that the TSch scheme needs to work over a larger field to guarantee the same security as the gjkr-TSCh scheme outweighs the benefits resulting from the fact that the TSch scheme requires only one round of broadcast while gjkr-TSCh needs two.

More specifically, if we take $b = 2^{80}$ as the target security bound, and assume that $q_H \approx 2^{80}$ as well, then the [PS96] results imply that Schnorr's signatures can be securely run in a group with at least $2^{80} \cdot 2^{80} = 2^{160}$ DLP security. Because the security reduction from the gjkr-TSCh scheme to the Schnorr signatures is tight, this implies that the gjkr-TSCh scheme is secure in the same 2^{160} -DLP group. On the other hand, the security reduction of Theorem 1 implies that the TSch scheme is secure in a group with $2^{80} \cdot 2^{160} = 2^{240}$ security of the DLP. In other words, the gjkr-TSCh threshold Schnorr scheme requires a factor of $\alpha = 160/80 = 2$ growth in the DLP security parameter, while the TSch threshold Schnorr scheme requires a factor of $\alpha = 240/80 = 3$ growth in the DLP security parameter.

Recall that the cost of exponentiation modulo p with an $|q|$ -bit exponent grows at best like $O(|q| \cdot |p|^{1.6})$. If we consider the DLP security in the classic number field setting, the factor α growth in the security parameter implies factor α^3 growth in the size of modulus p and factor α growth in the size of the modulus q . Therefore the overall cost of exponentiation grows like $\alpha \cdot (\alpha^3)^{1.6} = \alpha^{5.8}$. For elliptic curves, the sizes of both p and q grow by factor α , and hence the overall cost of exponentiation grows only by factor $\alpha^{2.6}$.

As a reference point for the speed of cryptographic operations we take the performance table of [Wei00], where on a Celeron 850 MHz an exponentiation in a number field with $|p| = 1024$ and $|q| = 160$ takes 2 ms and an exponentiation over a 168-bit elliptic curve takes 5 ms. We assume, after Lenstra and Verhulst [LV01], that in both settings DLP has security parameter 80 (i.e. the DLP in such groups is secure against an adversary with a $b = 2^{80}$ computational upper bound). If n is the number of players and $t \approx .5 \cdot n$, then in the TSch threshold Schnorr protocol, each player makes $1.5 \cdot n$ long exponentiations, while in the threshold Schnorr implemented with the [GJKR99] DKG protocol (let's denote this protocol as gjkr-TSch) each player performs $2.5 \cdot n$ long exponentiations.

Taking it all together, for a threshold system with $n = 7$ players, in the standard number field setting, each player's computation in the TSch protocol with 2^{80} security guarantees would take about $11 \cdot 3^{5.8} \cdot 2ms = 12.8s$, while in gjkr-TSch each player's computation takes only about $18 \cdot 2^{5.8} \cdot 2ms = 2s$. In this setting, the gjkr-TSch scheme is a winner, because taking the SINTRA implementation of reliable broadcast [CP02] as a reference point, a round of reliable broadcast between about 7 players would take about 1s on the internet and only about 100ms on LAN, and therefore the computation cost incurred by TSch outweighs the communication delay caused by an extra round of broadcast incurred by gjkr-TSch.

The TSch scheme may be slightly faster than gjkr-TSch in the elliptic curve setting with players distributed over the internet, but probably not when the players are on a LAN network. Each player's computation is $18 \cdot 2^{2.6} \cdot 5ms = .5s$ for gjkr-TSch, and $11 \cdot 3^{2.6} \cdot 5ms = .95s$ for TSch. Therefore, if the players are distributed over the internet where one round of broadcasts causes a delay of about 1s, the $.45s$ difference in computation is outweighed by the broadcast cost, and therefore the TSch scheme is preferable. On the other hand, the gjkr-TSch scheme still wins with TSch if the players are connected via a LAN, where the extra round of broadcast costs only about $.1s$.

On the other hand, if one takes our reduction as a *heuristic* argument that the TSch threshold Schnorr signature scheme with security parameter 80 can be achieved in fields where DLP also has security parameter 80, then the TSch scheme would win with gjkr-TSch: The computation time per player in TSch would be only about $11 \cdot 5ms = 55ms$ compared with $18 \cdot 5ms = 90ms$ for gjkr-TSch. Moreover the TSch scheme, having only one round of broadcast would incur a communication delay of about 100ms in the LAN setting or 1s in the internet setting, while the gjkr-TSch scheme would take, respectively, 200ms and 2s.

6 Open Problems

We point out that our methodology can be applied to showing security of other threshold discrete-log based cryptosystems implemented with the less expensive Pedersen's DKG protocol, if there is a security reduction to the original centralized versions of these schemes from the discrete logarithm assumption, or

from some other computational assumption in the discrete-log setting, e.g. the computational Diffie-Hellman. It is clear, for example, that our methodology applies to a threshold version of Chaum-Pedersen undeniable signature protocol [CP92], which is secure under the Diffie-Hellman assumption. Other candidates for this methodology include, but are not limited to, the threshold version of Cramer-Shoup encryption scheme given in [CG99] or the “Modified ElGamal” signature scheme of [PS96].

References

- [Bac85] E. Bach. Analytic Methods in the Analysis and Design of Number-Theoretic Algorithms ACM Distinguished Dissertation (1984). MIT Press, Cambridge, MA, 1985. [378](#)
- [BB89] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in a constant number of rounds. In *Proc. 8th ACM Symp. on Principles of Distributed Computation*, pages 201–209, 1989.
- [BR93] Mihir Bellare and Phillip Rogaway. Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993. [379](#)
- [CG99] R. Canetti and S. Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In *Eurocrypt '99*, pages 90–106, 1999. LNCS No. 1592. [374](#), [389](#)
- [CGJ⁺99] Ran Canetti, Rosario Gennaro, Stanisław Jarecki, Hugo Krawczyk, and Tal Rabin. Adaptive security for threshold cryptosystems. In *Proc. CRYPTO 99*, pages 98–115. Springer-Verlag, 1999. LNCS No. 1666. [383](#)
- [CGS97] R. Cramer, R. Gennaro, and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Eurocrypt '97*, pages 103–118, 1997. LNCS No. 1233. [374](#)
- [CP92] D. Chaum and T. Pederson. Wallet databases with observers. In *Crypto '92*, LNCS No. 740, pages 89–105, 1992. [389](#)
- [CP02] C. Cachin, and J. A. Poritz. Secure Intrusion-tolerant Replication on the Internet. In *Proc. Intl. Conference on Dependable Systems and Networks (DNS-2002)*, Washington DC, USA, IEEE, 2002. (see also <http://eprint.iacr.org/>) [375](#), [377](#), [378](#), [388](#)
- [CMI93] M. Cerecedo, T. Matsumoto, and H. Imai. Efficient and secure multi-party generation of digital signatures based on discrete logarithms. *IEICE Trans. Fundamentals*, E76-A(4):532–545, 1993. [374](#)
- [Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. *Crypto'87*, pages 120–127, 1987. LNCS No. 293. [373](#)
- [DF89] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Crypto '89*, pages 307–315, 1989. LNCS No. 435. [373](#)
- [Fel87] P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In *Proc. 28th FOCS*, pages 427–437. IEEE, 1987. [379](#)
- [FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto'86*, pages 186–194, 1986. LNCS No. 263. [382](#)

- [FMY99] Y. Frankel, P. D. MacKenzie, and M. Yung. Adaptively-secure distributed Public Key systems. In *Algorithms – ESA’99, 7th Annual European Symposium, Prague*, pages 4–27, 1999. LNCS No. 1643. [375](#), [383](#)
- [GJKR96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. In *Information and Computation* 164, pp.54–84, 2001. [374](#), [382](#)
- [GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. The (in)security of distributed key generation in dlog-based cryptosystems. In *Eurocrypt ’99*, pages 295–310, 1999. LNCS No. 1592. [373](#), [374](#), [377](#), [380](#), [388](#)
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Computing*, 17(2):281–308, April 1988. [378](#), [382](#)
- [GRR98] Rosario Gennaro, Michael Rabin, and Tal Rabin. Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In *Proc. 17th ACM Symp. on Principles of Distributed Comp.*. ACM, 1998.
- [Har94] L. Harn. Group oriented (t, n) digital signature scheme. In *IEE Proc.-Comput. Digit. Tech.*, 141(5):307–313, Sept 1994. [374](#)
- [HJJ⁺97] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. Proactive public key and signature systems. In *1997 ACM Conference on Computers and Communication Security*, 1997. [374](#)
- [Jar01] S. Jarecki. Efficient Threshold Cryptosystems. *MIT PhD Thesis*, June 2001, theory.lcs.mit.edu/~cis/cis-theses.html. [377](#)
- [JL00] Stanisław Jarecki and Anna Lysyanskaya. Adaptively secure threshold cryptosystems without erasures. In *Eurocrypt’00*, pages 221–242, 2000. LNCS. No. 1807. [383](#)
- [LHL94] C.-H. Li, T. Hwang, and N.-Y. Lee. (t, n) threshold signature schemes based on discrete logarithm. In *Eurocrypt ’94*, pp. 191–200, 1994. LNCS No. 950. [374](#)
- [LV01] A. K. Lenstra and E. R. Verheul. Selecting Cryptographic Key Sizes. In *Journal of Cryptology*, vol. 14(4), 2001, pages 255–293. [388](#)
- [Ped91a] Torben Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Crypto ’91*, pages 129–140. 1991.
- [Ped91b] Torben Pedersen. A threshold cryptosystem without a trusted party. In *Eurocrypt ’91*, pages 522–526, 1991. LNCS No. 547. [374](#), [379](#)
- [PK96] C. Park and K. Kurosawa. New ElGamal Type Threshold Digital Signature Scheme. *IEICE Trans. Fundamentals*, E79-A(1):86–93, January 1996. [374](#)
- [PS96] D. Pointcheval, and J. Stern, Security Proofs for Signature Schemes. *Eurocrypt’96*, pages 387–398, 1996. LNCS No. 1070. [376](#), [382](#), [385](#), [387](#), [389](#)
- [Sha79] A. Shamir. How to Share a Secret. *CACM*, 22:612–613, 1979. [373](#), [374](#)
- [Sch89] P. Schnorr. Efficient identification and signatures for smart cards. - *Crypto ’89*, pages 235–251, 1989. LNCS No. 435. [375](#), [382](#)
- [Sho00] Victor Shoup. Practical threshold signatures. In *Eiurocrypt ’00*, pages 207–220. Springer-Verlag, 2000. [375](#)
- [SG98] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. In *Eurocrypt ’98*, pages 1–16, 1998. LNCS No. 1403. [374](#)
- [Wei00] Wei Dai. Benchmarks for the Crypto++ 4.0 library performance. Available at <http://www.eskimo.com/~weidai/cryptlib.html> [388](#)

Seeing through MIST Given a Small Fraction of an RSA Private Key

Colin D. Walter

Comodo Research Lab
10 Hey Street, Bradford, BD7 1DQ, UK
Colin.Walter@comodogroup.com
www.comodogroup.com

Abstract. In smartcard encryption and signature applications, randomised algorithms are used to increase tamper resistance against attacks based on side channel leakage. MIST is one of these. As is the case with the classical m -ary and sliding windows exponentiation algorithms, the most significant half of the public modulus yields information which can be used to halve the number of key digits which need to be guessed to recover the secret key from a MIST side channel trace. Lattice based methods are used to reduce this to just one quarter of the least significant digits. This enables the strength of the MIST exponentiation algorithm to be gauged more accurately under several threat models.

Keywords: Addition chains, division chains, randomized exponentiation, Mist, randomary exponentiation, RSA, side channel leakage, power analysis, SPA, DPA, SEMA, DEMA, blinding, smartcard.

1 Introduction

Smartcards have very limited scope for the inclusion of physical security measures. So tamper resistance should also be built into the component algorithms whenever possible. Because exponentiation is such a major process in many crypto-systems, including RSA, Diffie-Hellman and ECC, it is the main target for improved algorithmic methods. Initial side channel attacks made use of timing differences [8] due to conditional subtractions during modular multiplications [17]. Such differences are now easily avoided [19]. However, because unprotected hardware gates use different amounts of power depending upon whether or not they are switched to a different state, power usage by any chip is also data dependent, and so provides side channel leakage which may enable secret keys to be recovered. Although the first power attacks required averaging over a number of exponentiations in order to reduce the effects of noise and dependence on irrelevant data [9, 12], recent progress suggests that there is enough data in the trace of a single exponentiation to allow careful averaging to reveal the key [18]. In particular, one can average over subsets of digit-by-digit products within each long integer multiplication rather than the sequential traces of many exponentiations. In addition, electro-magnetic leakage seems to provide a very much more powerful means of obtaining key data than power variation [15, 16, 4, 1].

Such side channels and methods enable all the classical exponentiation methods to be attacked: both sliding windows and m -ary [7]. Consequently, a new breed of algorithms has had to be developed, namely the *randomised* exponentiation algorithms. Two such algorithms appeared at the CHES 2001 conference [11, 14] and further ones at, for example, the RSA 2002 [21] and CHES 2002 conferences [5, 6]. The first two seem to provide little extra security if one assumes squares and multiplies (adds and doubles) can usually be distinguished during a single exponentiation and no key blinding is used [13, 23]; they are also only suitable for elliptic curve (ECC) applications because of the requirement for the computation of inverses. However, the more recent algorithms seem to be more robust [22], and some of them can be applied easily to RSA because no inverses need be computed.

The purpose of this article is to contribute further to an assessment of the strength of the third of these randomised exponentiation algorithms, namely MIST [21]. First, elementary methods are applied to show that half of the digits generated by MIST suffice to reveal the key when the public modulus and exponent are known. The techniques of Boneh *et al.* [2] have been used formerly on classical representations of the secret key to show that only a quarter of the bits of the key need be known before algorithmic methods can be applied to determine the complete key in polynomial time. Here those techniques are adapted to the “randomary” representation of the key which MIST generates. The results are similar: a quarter of the randomary digits are sufficient to enable recovery of the full key in polynomial time.

These results are then applied to reduce the search space for keys which are partially revealed through side channel leakage. Although the conclusions show the computational effort to break the keys is still infeasible for standard key lengths, there is little room left for complacency. Key blinding or keys of at least 1024 bits would still appear to be necessary for good tamper resistance. By comparison, of course, the quality of side channel leakage which we assume is instantly fatal to the classical algorithms.

2 The MIST Algorithm

For notation, suppose that the RSA crypto-system has known public modulus N , public exponent E and private exponent D (after any required blinding), so that plaintext P and ciphertext C are related by $P = C^D \bmod N$. It is side channel leakage from this exponentiation that an attacker uses to extract the secret key D .

The MIST algorithm [20, 21] is a random-ary exponentiation method very similar to m -ary exponentiation, but it reverses the order of processing the digits of D , and the base m is varied randomly for each digit choice. The following version computes $P = C^D \bmod N$.

THE MIST EXPONENTIATION ALGORITHM

```

{ Pre-condition:  $D \geq 0$  }
 $Q \leftarrow C$  ;
 $P \leftarrow 1$  ;
While  $D > 0$  do
Begin
  Choose a random base  $m$  from set  $S$  ;
   $d \leftarrow D \bmod m$  ;
  If  $d \neq 0$  then  $P \leftarrow Q^d \times P \bmod N$  ;
   $Q \leftarrow Q^m \bmod N$  ;
   $D \leftarrow D \bmod m$  ;
  { Loop invariant:  $C^{D.Init} = Q^D \times P \bmod N$  }
End ;
{ Post-condition:  $P = C^{D.Init} \bmod N$  }

```

The values of d are analogous to digits of D in representations where the base m is constant. Generally, the random choice of m is from a fixed set with known security and efficiency properties, such as $S = \{2, 3, 5\}$. When the random base set S consists of the single divisor 2, the method simplifies to the binary square-and-multiply algorithm in which the least significant exponent bit is processed first. In general, for a singleton set $S = \{m\}$, the algorithm simplifies to classical m -ary exponentiation but performed from right to left rather than from left to right. Space and time efficiency were shown in [21] to be comparable with 4-ary exponentiation when addition chains for computing Q^m compute Q^d en route rather than sequentially, and m is chosen with a suitable bias which favours $m = 2$ or $d = 0$.

The random choice of bases generates different exponentiation schemes on successive runs and so makes impossible the usual averaging process required for power analysis (SPA/DPA) [9] or electro-magnetic analysis (SEMA/DEMA) [15]. Hence, we assume that the attacker has extracted data from a single exponentiation which enables him to determine either i) which operations are squares and which are multiplications, or ii) which operations share an argument. In [22], the following results were established:

Theorem 1 ([22], Thm. 9). *Suppose squares and multiplies can be distinguished, but not individual reuse of operands. Then, for the choice of parameters given in [22], the average number of exponents which can generate the same sequence of squares and multiplications as a given one for D is bounded below by $D^{3/5}$.*

Theorem 2 ([22], Thm. 8). *For the choice of parameters given in [22], the average number of exponents with exponentiation schemes that have an operand sharing pattern identical to a given one for D is about $D^{1/3}$.*

Both of these theorems are established by counting the number of choices for the pairs (m, d) . For example, addition chains for $(2, 1)$ and $(3, 0)$ are chosen to

give them the same operand sharing patterns and square & multiply sequences. Every time one of these patterns turns up, there is an ambiguity about the choice for D which effectively doubles the size of the search space for the correct D . In other cases there is a unique choice, or even three or four choices for (m, d) which give the pattern which the attacker has recognised. Simulations indicate that there is very little collapsing of the search space as a result of duplication in the reconstructed exponents. However, as one of the reviewers has pointed out, the size of the search space is not necessarily an accurate measure of the time to recover the correct key. If efficiency considerations dictate non-uniform selection criteria for d , then some exponents are more likely than others. So, on average, an intelligent search for the correct exponent would only have to traverse a fraction of the entire space. Nevertheless, if each d is equally likely, then half the space must be traversed. This latter is the case assumed in the illustrative results below, but it is easy to adapt the deductions to alternative situations.

If the digits and bases are indexed from 0 to $n-1$ in the order generated, then the following notation, initial conditions and properties hold, just as for a fixed base m :

$$\begin{aligned} D_0 &= D \\ d_i &\equiv D_i \bmod m_i \\ D_{i+1} &= D_i \text{ div } m_i \\ d_{n-1} &\neq 0 \\ D_n &= 0 \end{aligned} \tag{1}$$

$$\begin{aligned} D_i &= m_i D_{i+1} + d_i \\ D_i &= ((\dots(d_{n-1} m_{n-2} + d_{n-2})\dots)m_{i+1} + d_{i+1})m_i + d_i \\ D &= ((\dots(d_{n-1} m_{n-2} + d_{n-2})\dots)m_1 + d_1)m_0 + d_0 \end{aligned} \tag{2}$$

This indexing follows that of digits in the standard base m representation. Some additional related quantities are of use later:

$$\begin{aligned} \mu_i &= \prod_{j=0}^{i-1} m_j \\ \delta_i &= ((\dots(d_{i-1} m_{i-2} + d_{i-2})\dots)m_1 + d_1)m_0 + d_0 \end{aligned} \tag{3}$$

which satisfy

$$\begin{aligned} \delta_i &\equiv D \bmod \mu_i \\ D_i &= D \text{ div } \mu_i \\ D &= \mu_i D_i + \delta_i \end{aligned} \tag{4}$$

Exponent blinding is assumed throughout, since this is probably essential to help address the vulnerability of applications for which the algorithm is considered necessary. So D will always represent the exponent actual used, with its digits as above, while D' will be the original unblinded secret key. Notation for the blinding is introduced in the next section.

3 Halving the Search Space

The starting point for both of the main results is the observation that $\phi(N)$ has a good approximation given in terms of N . If $N = PQ$ is the prime factorisation of N , the standard choice of P and Q to have the same number of bits allows an attacker to assume that $P < Q < 2P$. Then $2\sqrt{N} < P+Q < 3\sqrt{N/2}$ and so $\phi(N) = N - (P+Q) + 1$ is bounded by

$$N - 3\sqrt{N/2} + 1 < \phi(N) < N - 2\sqrt{N} + 1 \quad (5)$$

This interval has length less than $\frac{1}{8}\sqrt{N}$, so that three more than half the most significant bits of $\phi(N)$ can be determined trivially.

Suppose that the public and private exponents, E and D' respectively, are related by

$$D' \times E = 1 + k\phi(N) \quad (6)$$

where it is reasonable to assume that D' is chosen to make $D' < \phi(N)$ so that $k < E$. However, to such D' a blinding factor should normally be added [8], giving the secret key

$$D = D' + r\phi(N) \quad (7)$$

which is actually used for an exponentiation. Here r is a random number, often of 32 bits. Then

$$D = \frac{1 + (k+rE)\phi(N)}{E} \quad (8)$$

Let B be an upper bound on such r . Then, in effect, the coefficient $k+rE$ of $\phi(N)$ is a random number in the range 0 to BE . The attacker just has to generate each of the $O(BE)$ possible values of the random coefficient of $\phi(N)$ in order to obtain a set containing an approximation to the value of D used in the exponentiation which he has observed. If there is no blinding, $r = 0$ is taken, and $B = 1$ will give the relevant results for that case.

For a given choice of r and k , let D_l and D_u denote the (irrational) lower and upper bounds on D determined by equations (5) and (8). So, for the assumptions made above,

$$D_u - D_l = \frac{k+rE}{E}(3\sqrt{1/2} - 2)\sqrt{N} < \frac{B}{8}\sqrt{N} \quad (9)$$

Next the attacker must generate all possibilities for the last half or so of the digits of D , taking enough of them to construct a D_j satisfying both

$$D_j \geq \sqrt{\frac{(k+rE)N}{E}} \quad \text{and} \quad D_j \geq D_u - D_l \quad (10)$$

The first inequality is usually a consequence of the second. By (9), this is the case if $\sqrt{\frac{k+rE}{E}} \leq \frac{k+rE}{E}(3\sqrt{1/2} - 2)$, i.e. if $\frac{k+rE}{E} \geq (3\sqrt{1/2} - 2)^{-2}$, which holds for $r \geq 68$. So we can normally ignore it in the following estimates.

Depending on the threat model for side channel leakage, the digit choices will be restricted in some way. We will consider in more detail the two particular cases covered by Theorems 1 and 2. In each case, the ambiguities for each base/digit pair (m_i, d_i) , $j \leq i \leq n-1$, were described in detail in [22]. The attacker's next problem is to identify which such choice is correct.

By (8) and the first inequality of (10), $D_j > \sqrt{D}$, so that (4) gives $\delta_j < \mu_j < \sqrt{D} < D_j$. Thus $\mu_j = D \operatorname{div} D_j$ by (4). By the second inequality of (10), $D_l/D_j < D/D_j < D_u/D_j \leq D_l/D_j + 1$. So, combining these,

Lemma 1. *For j as chosen above, $\mu_j = D \operatorname{div} D_j = \lfloor D_l/D_j \rfloor$ or $\lfloor D_u/D_j \rfloor$.*

These two quantities can be calculated from the assumed values of D_l and D_u and used to reject D_j if neither expression has the characteristic property of μ_j , namely being a product of elements from the chosen base set $S = \{2, 3, 5\}$. This should almost completely determine the correct D_j .

Our next task is to estimate how many values of D_j will be accepted by this process. It is reasonable to assume $\lfloor D_l/D_j \rfloor$ and $\lfloor D_u/D_j \rfloor$ are effectively random in terms of their prime factorisations. So we need to know the probability that a random number of size D/D_j will be a product of powers of only 2, 3 and 5.

Suppose $\mu = 2^x 3^y 5^z < K$. Then there are at most $\log_2 K$ possible choices for x , $\log_3 K$ for y and $\log_5 K$ for z , making a total of at most $(\log 2 \log 3 \log 5)^{-1} (\log K)^3$ choices for μ . Differentiating with respect to K , this in turn means a maximum density of

$$3(\log 2 \log 3 \log 5)^{-1} (\log K)^2 / K \quad (11)$$

for numbers of size K with the required form. So these forms are quite rare.

Combining (5) and (8) with the equality in (9) yields essentially

$$\frac{\sqrt{N} - 3\sqrt{1/2}}{3\sqrt{1/2} - 2} < \frac{D}{D_u - D_l} < \frac{\sqrt{N} - 2}{3\sqrt{1/2} - 2} \quad (12)$$

If D_j is chosen to be minimal such that the second inequality of (10) holds then $D_u - D_l \leq D_j \leq 5(D_u - D_l)$ because 5 is the maximum base. Thus, ignoring insignificant terms, the numbers of interest are bounded below by $D \operatorname{div} D_j \geq \lfloor D_l/5(D_u - D_l) \rfloor \geq \lfloor \frac{\sqrt{N} - 3\sqrt{1/2}}{5(3\sqrt{1/2} - 2)} \rfloor \approx \frac{5}{3}\sqrt{N}$ and bounded above by $D \operatorname{div} D_j \leq D_u/(D_u - D_l) \leq \frac{\sqrt{N} - 2}{3\sqrt{1/2} - 2} < 9\sqrt{N}$. So we may use $K = \frac{5}{3}\sqrt{N}$ in (11) to provide an upper bound on the density of $\{2, 3, 5\}$ -powers in the region containing μ_j . If, instead, D_j is chosen minimal such that the first inequality of (10) holds then $D \operatorname{div} D_j > \frac{1}{5}\sqrt{D} \approx \frac{1}{5}\sqrt{rN}$ where $r < 68$. So again we must choose K in (11) to be $O(\sqrt{N})$.

However, by Theorems 1 and 2, there are up to $D_j^{3/5}$ or $D_j^{1/3}$ values of D_j with the right patterns to consider. Here $O(D_j) = O(D_u - D_l) = O(B\sqrt{N})$ by (9). Hence, by (11), the expected number of values D_j for which $D_l \operatorname{div} D_j$ or $D_u \operatorname{div} D_j$ has the right form is at most

$$O(B^{3/5} \{\log N\}^2 N^{-1/5}) \quad \text{or} \quad O(B^{1/3} \{\log N\}^2 N^{-1/3}) \quad (13)$$

respectively for each choice of k and r . In both cases, this is less than 1 for some expected sizes of N (192 bits in the case of ECC, say) and B (32 bits, say). This indicates that essentially only one solution is likely to exist, which will be the required one if k and r are chosen correctly. However, without the aid of any leaked data, an exhaustive search will have D_j cases to consider and will reduce the exponent of N in this expression to 0. Thus, any side channel which reduces the search space to below that of an exhaustive search will reduce the expected number of D_j generating the right form to at most $O(B\{\log N\}^2)$.

Once a feasible D_j is determined, the remaining part of D is straightforward to reconstruct iteratively. Using equation 2(i) repeatedly, each new value D_i ($j-1 \geq i \geq 0$) is constructed from the preceding D_{i+1} and the divisibility of the associated μ_i by only 2, 3 or 5 is checked in the same way as for D_j in order to determine which choice of (m_i, r_i) is correct.

The effort for this is minimal at least for $i \approx j$. However, as i decreases, the probabilities change: $\mu_i = D \text{ div } D_i$ has fewer and fewer factors, and so is more and more likely to be of the correct form. For $\mu_i \approx 3 \times 2^9$ there are only about $\mu_i^{3/5} \approx 54$ or $\mu_i^{1/3} \approx 9$ remaining choices for extending D_i to D in a way consistent with the side channel leakage. Also, only 79 values from the interval $[1 \dots 3 \times 2^9]$ give subsequent μ_i 's with the right form. So there are very few incorrect choices which satisfy all the criteria and every plausible case can be investigated in full. For $\mu_i > 3 \times 2^9$, (11) shows that at most 1 in 12 numbers have the requisite form. The average base value m_i is 2.50 so that the previous t base values reduce D by a factor of about 2.50 t . So there are around $(2.50^t)^{3/5}$ or $(2.50^t)^{1/3}$ choices respectively for these t base/digit pairs. Any one of these has a probability at most 12^{-t} of being acceptable under the divisibility criterion. Hence at most about $(2.50^t)^{3/5}12^{-t}$ or $(2.50^t)^{1/3}12^{-t}$ of the incorrect choices will survive t base choices. These both tend to 0 as t increases. Hence, although one may temporarily have to consider some additional incorrect choices for D_i , these will disappear very quickly as i decreases. Indeed, with a probability of less than 1 in 12 of acceptability and at most 4 choices for extending D_{i+1} to D_i , the average number of further iterations which an incorrect value survives is less than $\sum_{i=1}^{\infty} i(\frac{4}{12})^i = \frac{3}{4}$. Thus, normally there will be fewer than two values of D_{i+1} which are under consideration for generating D_i – the correct one (if k and r were selected correctly) and at most one incorrect one. The computational cost of generating possible D from a correct D_j is therefore of the same order as that of applying the divisibility criterion j times: once for each $i < j$; and the computational cost for an incorrect D_j is just the cost of a constant number of applications of the divisibility criterion.

In total, there were $O(BE)$ values for (k, r) , and hence for D_l and D_u , and $O(\{B\sqrt{N}\}^{3/5})$ or $O(\{B\sqrt{N}\}^{1/3})$ ways for choosing D_j from each D_l or D_u . Thus, from this method there will be respectively at most

$$O(EB^{8/5}N^{3/10}) \quad \text{or} \quad O(EB^{4/3}N^{1/6}) \tag{14}$$

possible values for D_j , and hence of D , which are generated and need checking. As noted above, almost all D_j will be rejected by the divisibility criterion on

its first or second application and not lead to a viable D . These figures give the order of work involved in an attack. For standard RSA key sizes of 1024 or more bits and no other data to narrow the search space even further, this is still a computationally infeasible task for any E and no blinding ($B = 1$). Observe that increased security is obtained more cheaply by increasing the public exponent E or the size of the blinding factor B rather than the modulus N .

Other parameter choices and threat scenarios for MIST can be tested in a similar way to see if the search space still remains large enough to prevent a successful attack.

4 Quartering the Search Space

The lattice-based techniques of Coppersmith [3] produce complementary results to those in the previous section. They enable one to deduce the secret key from the least significant digits rather than the most significant ones. Existing relevant work for recovering secret keys from the public modulus and such partial knowledge of the key is exclusively centred on a bit-based view of the key. With bases 3 and 5 as possibilities, the bit-based view must be reformulated in a less base-dependent way. Fortunately, the generalisations involve few complications. So, analogously to the quarter of bits of D that Boneh *et al.* [2] use, a similar argument here requires on average about a quarter of the digit/base pairs to be established. Effectively, the computational effort must then be directed at an exhaustive search of a space whose size is a fractional power of D with exponent only a quarter of the value given in Theorems 1 or 2. The main result needed in the proof is the following:

Theorem 3 ([2], Cor. 2.2). *Let $N = PQ$ be an n -bit RSA modulus. Let $\mu \geq 2^{n/4}$ be given and suppose $P_0 \stackrel{\text{def}}{=} P \bmod \mu$ is known. Then it is possible to factor N in time polynomial in n .*

Both here and in [2], the choice made for μ is a product of base values. This makes μ a power of 2 in [2], but a product of powers of 2, 3 and 5 here. Suppose pairs (m_i, d_i) have been guessed correctly for $0 \leq i \leq j-1$ and some j . Then δ_j and μ_j are known, and satisfy $D \equiv \delta_j \bmod \mu_j$, as in equation 4(i). μ_j is the value which will be taken for μ in the theorem, so we choose j large enough that $\mu = \mu_j \geq 2^{n/4}$.

Rewriting equation (8) entirely in terms of N and P rather than N and $\phi(N)$ gives

$$DE = 1 + (k+rE)(N-P-N/P+1) \quad (15)$$

Reducing this modulo $\mu = \mu_j$, we see that $P_0 \equiv P \bmod \mu$ is a solution for x in the equation

$$\delta_j E \equiv 1 + (k+rE)(N-x-N/x+1) \bmod \mu \quad (16)$$

and hence a root of

$$(k+rE)x^2 - (1-\delta_j E+(k+rE)(N+1))x + (k+rE)N \bmod \mu \quad (17)$$

The coefficients here are all divisible by $k+rE$ because $(k+rE)\phi(N) = 1 - DE \equiv 1 - \delta_j E \pmod{\mu}$. Suppose $g = \gcd\{k+rE, \mu\}$. Then, dividing (17) through by $k+rE$, $P_0 \equiv P \pmod{\mu}$ is a root of the quadratic

$$x^2 - ((1 - \delta_j E)/(k+rE) + N+1)x + N \pmod{\mu g^{-1}} \quad (18)$$

As before, the attack must consider separately every possible value for $k+rE$. g is easily computed once $k+rE$ is chosen, and so (18) is obtained.

Solutions are most easily obtained by first completing the square. $N+1$ and $(DE-1)/(k+rE) = \phi(N)$ are both even, so the coefficient of x in (18) is also even. In fact, $Q_0 \equiv Q \pmod{\mu}$ is a second solution to (18) and the coefficient of x is then $-(P_0 + Q_0)$. So, by taking $y = x - \frac{1}{2}((1 - \delta_j E)/(k+rE) + N+1)$ in (18), x is obtained from the solutions of an equation of the form

$$y^2 \equiv c \pmod{\mu g^{-1}} \quad (19)$$

where c is easily computed. For solutions to exist, any power of 2, 3 or 5 which divides c must occur to an even power. If this is not the case, the wrong $k+rE$ must have been chosen, and so the next one should be selected. If it is the case, then that power can be removed from y easily and replaced later. So, without loss of generality, assume c is prime to each of 2, 3 and 5. Thus we are looking for a solution prime to μg^{-1} .

The solutions to (19) are obtained by solving modulo the maximal 2-, 3- and 5-powers which divide μg^{-1} and then using the Chinese Remainder Theorem to reconstruct the solutions modulo μg^{-1} . Solutions modulo the prime powers are obtained by lifting solutions progressively from lower powers of the prime. For higher powers of 2 than 2^2 , Boneh *et al.* ([2], App. A) have previously noted that there are at most 4 solutions if there are solutions at all, and have provided a construction process for them. For an odd prime p , residues $\pmod{p^t}$ ($t \geq 1$) which are prime to p form a cyclic group of order $\phi(p^t)$ under multiplication (e.g. [10], Thm. 7.2.10). Hence (19) will have exactly two solutions, say $\pm y_t$, or none at all for $\mu g^{-1} = p^t$. Any solutions modulo p^{t+1} must then have the form $\pm(y_t + \delta_t p^t)$ for some δ_t in $[0 \dots p-1]$. Each δ_t can be tried in turn to find the solutions. Once more, no solutions at any point means the wrong value for $k+rE$; otherwise there will be two.

Using the Chinese Remainder Theorem to combine the results modulo the powers of 2, 3 and 5, the number of solutions is the product of the number of solutions modulo the individual prime powers, namely $4 \times 2 \times 2 = 16$, assuming at least $2^3 \times 3 \times 5$ divides the modulus and each has at least one solution. These 16 solutions are all different because, by reducing them modulo each of the three prime powers, different solutions sets modulo the prime powers are recovered.

The effort in obtaining and combining the solutions in this way for a fixed $k+rE$ is proportional to the sum of the exponents of the powers of 2, 3 and 5 dividing μg^{-1} . It is thus a computation of order at most $\log(\mu g^{-1})$, and hence of order at most $\log \mu$. As j was chosen so that $\mu \geq 2^{n/4}$ and there are up to $B E$ choices for $k+rE$, the solutions now provide all possible values for the P_0 in Theorem 3 for a computational effort of $O(nBE)$. Hence,

Theorem 4. *For a public RSA modulus N with n bits, public exponent E and exponent blinding using a random multiple of $\phi(N)$ which is bounded above by B , given the least significant randomary digits of D whose product of bases is at least $N^{1/4}$, it is possible to factorise N in time which is polynomial in n and linear in BE .*

As a corollary, it is possible to deduce the maximum effort required to deduce the secret key under the assumptions stated in Theorems 1 and 2. The number of partial keys which need to be tested is of order $(N^{1/4})^{3/5}$ and $(N^{1/4})^{1/3}$ respectively.

Corollary 1. *Under the threat scenarios described in Theorems 1 and 2 with known public modulus N and encryption key E , it is possible to factorise N in time which is a product of polynomial time in $\log N$ and linear time in $BEN^{3/20}$ or $BEN^{1/12}$ respectively.*

It is now clear that exponent blinding is important in the prevention of such attacks, and such blinding provides more tamper resistance per bit than increasing the key length. If similar but more powerful attacks than those described in [22] can be developed, then blinding will become strongly advisable if the key length is under 1024 bits. Furthermore, we have assumed the bases m_i were chosen at random. However, biasing the choice for efficiency or other reasons further reduces the time for executing the attack.

5 Conclusion

Two techniques have been demonstrated for using knowledge of the public RSA modulus and exponent in order to reduce the computational effort of recovering the secret key from partial knowledge of a randomary digit expansion of D generated by the MIST exponentiation algorithm. The first relies on the scarcity of large numbers which are products of powers of only 2, 3 and 5. The other generalises the well-known, lattice-based, “Bellcore Attack” method of Boneh, Durfee and Frankel [2].

Under threat models in which squares and multiplies or operand re-use can be recognised during a single exponentiation, these techniques do not undermine the confidence that power and EMR attacks (SPA/DPA and SEMA/DEMA) on the MIST exponentiation algorithm still appear to be computationally infeasible for standard key lengths and sensible implementations which include appropriate blinding. Of course, under the same assumptions, the classical sliding windows and m -ary exponentiation schemes provide no resistance whatsoever.

References

- [1] D. Agrawal, B. Archambeault, J. R. Rao & P. Rohatgi, *The EM Side-Channels*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2002, *to appear*. [391](#)
- [2] D. Boneh, G. Durfee & Y. Frankel, *Exposing an RSA Private Key Given a Small Fraction of its Bits*, Advances in Cryptology – AsiaCrypt '98, K. Ohta & D. Pei (editors), LNCS **1514**, Springer-Verlag, 1998, 25–34. [392](#), [398](#), [399](#), [400](#)
- [3] D. Coppersmith, *Small Solutions to Polynomial equations and low exponent RSA vulnerabilities*, Journal of Cryptology **10** (1997), 233–260. [398](#)
- [4] K. Galdolfi, C. Mourtel & F. Olivier, *Electromagnetic Analysis: Concrete Results*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, 251–261. [391](#)
- [5] J. C. Ha & S. J. Moon, *Randomized signed-scalar multiplication of ECC to resist power attacks*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2002, *to appear*. [392](#)
- [6] K. Itoh, J. Yajima, M. Takenaka, & N. Torii, *DPA Countermeasures by improving the window method*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2002, *to appear*. [392](#)
- [7] D. E. Knuth, *The Art of Computer Programming*, vol. **2**, “Seminumerical Algorithms”, 2nd Edition, Addison-Wesley, 1981, 441–466. [392](#)
- [8] P. Kocher, *Timing Attack on Implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology – CRYPTO '96, N. Koblitz (editor), LNCS **1109**, Springer-Verlag, 1996, 104–113. [391](#), [395](#)
- [9] P. Kocher, J. Jaffe & B. Jun, *Differential Power Analysis*, Advances in Cryptology – CRYPTO '99, M. Wiener (editor), LNCS **1666**, Springer-Verlag, 1999, 388–397. [391](#), [393](#)
- [10] R. Kumanduri & C. Romero, *Number Theory with Computer Applications*, Prentice Hall, 1998, ISBN 0-13-801812-X. [399](#)
- [11] P.-Y. Liardet & N. P. Smart, *Preventing SPA/DPA in ECC Systems using the Jacobi Form*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, 391–401. [392](#)
- [12] T. S. Messerges, E. A. Dabbish & R. H. Sloan, *Power Analysis Attacks of Modular Exponentiation in Smartcards*, Cryptographic Hardware and Embedded Systems (Proc. CHES 99), C. Paar & Ç. Koç (editors), LNCS **1717**, Springer-Verlag, 1999, 144–157. [391](#)
- [13] K. Okeya & K. Sakurai, *On Insecurity of the Side Channel Attack Countermeasure using Addition-Subtraction Chains under Distinguishability between Addition and Doubling*, Information Security and Privacy, L. Batten & J. Seberry (editors), LNCS **2384**, Springer-Verlag, 2002, 420–435. [392](#)
- [14] E. Oswald & M. Aigner, *Randomized Addition-Subtraction Chains as a Countermeasure against Power Attacks*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, 39–50. [392](#)
- [15] J.-J. Quisquater & D. Samyde, *ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards*, Smart Card Programming and Security (e-Smart 2001), LNCS **2140**, Springer-Verlag, 2001, 200–210. [391](#), [393](#)

- [16] J.-J. Quisquater & D. Samyde, *Eddy current for Magnetic Analysis with Active Sensor*, Proc. e-Smart 2002, Nice, September 2002, 183–194. [391](#)
- [17] C. D. Walter & S. Thompson, *Distinguishing Exponent Digits by Observing Modular Subtractions*, Topics in Cryptology – CT-RSA 2001, D. Naccache (editor), LNCS **2020**, Springer-Verlag, 2001, 192–207. [391](#)
- [18] C. D. Walter, *Sliding Windows succumbs to Big Mac Attack*, Cryptographic Hardware and Embedded Systems – CHES 2001, Ç. Koç, D. Naccache & C. Paar (editors), LNCS **2162**, Springer-Verlag, 2001, 286–299. [391](#)
- [19] C. D. Walter, *Precise Bounds for Montgomery Modular Multiplication and Some Potentially Insecure RSA Moduli*, Topics in Cryptology – CT-RSA 2002, B. Preneel (editor), LNCS **2271**, Springer-Verlag, 2001, 30–39. [391](#)
- [20] C. D. Walter, *Improvements in, and relating to, Cryptographic Methods and Apparatus*, UK Patent Application 0126317.7, Comodo Research Laboratory, 2001. [392](#)
- [21] C. D. Walter, *MIST: An Efficient, Randomized Exponentiation Algorithm for Resisting Power Analysis*, Topics in Cryptology – CT-RSA 2002, B. Preneel (editor), LNCS **2271**, Springer-Verlag, 2002, 53–66. [392](#), [393](#)
- [22] C. D. Walter, *Some Security Aspects of the MIST Randomized Exponentiation Algorithm*, Cryptographic Hardware and Embedded Systems – CHES 2002, B. Kaliski, Ç. Koç & C. Paar (editors), LNCS **2523**, Springer-Verlag, 2002, *to appear*. [392](#), [393](#), [396](#), [400](#)
- [23] C. D. Walter, *Breaking the Liardet-Smart Randomized Exponentiation Algorithm*, Proc. Cardis '02, USENIX, *to appear*. [392](#)

Simple Backdoors for RSA Key Generation

Claude Crépeau¹ and Alain Slakmon²

¹ School of Computer Science, McGill University
3480 rue University, room 318, McConnell Eng. Bldg,
Montréal (Québec), Canada H3A 2A7

crepeau@cs.mcgill.ca

² Département de mathématiques
Collège de Bois-de-Boulogne
10555 avenue de Bois-de-Boulogne
Montréal (Québec), Canada H4N 1L4

Alain.Slakmon@bdeb.qc.ca

Abstract. We present extremely simple ways of embedding a backdoor in the key generation scheme of RSA. Three of our schemes generate two genuinely random primes p and q of a given size, to obtain their public product $n = pq$. However they generate private/public exponents pairs (d, e) in such a way that appears very random while allowing the author of the scheme to easily factor n given only the public information (n, e) . Our last scheme, similar to the PAP method of Young and Yung, but more secure, works for any public exponent e such as 3, 17, 65537 by revealing the factorization of n in its own representation. This suggests that nobody should rely on RSA key generation schemes provided by a third party.

1 Introduction

As we all know, the RSA public-key Cryptosystem and Digital signature schemes are now in the public domain, which means that anybody may include them as means of confidentiality and authenticity in software products, smartcards, etc. The question that we raise here is how much can a user tell that an implementation of RSA he uses is safe and actually protects him? Recall the (in)famous “NSA-KEY” incident of Microsoft’s CryptoAPI system. How easy is it for software developers and smartcard builders to embed a backdoor in their RSA key generation scheme that will be unnoticed by the user but allows the author to defeat the confidentiality and authenticity of the resulting RSA public-key Cryptosystem and Digital signature scheme ?

We present extremely simple ways of embedding a backdoor in the key generation scheme of RSA. Three of our schemes generate two genuinely random primes p and q of a given size, to obtain their public product $n = pq$. However they generate private/public exponents pairs (d, e) in such a way that appears very random while allowing the author of the scheme to easily factor n given only the public information (n, e) . Our fourth scheme, similar to the PAP method of Young and Yung, but more secure, works for any public exponent e

such as 3, 17, 65537 by embedding a backdoor to the factorization of n in its own representation. Our methods will modify only slightly the running time of the standard key generation process and thus will be unnoticeable from a timing point of view. Moreover, we conjecture that backdoored keys will be distributed in such a way that even with large samples they will remain indistinguishable from genuinely random keys. Our schemes do not expect the generation algorithm to have memory of its previous executions. We assume our algorithms are ran on a memoryless device, thus discarding even simpler methods based on pseudorandom generation that would be deterministically generating keys in a way reproducible by the software developer. Our methods use randomization in a way to generate a large set of keys, each of which is breakable by the developer.

For this purpose, one scheme relies on the well known attacks on RSA small private exponents by Wiener and Boneh and Durfee. The scheme creates a random weak pair of private/public exponents (δ, ϵ) with small δ , and transforms it into a random looking private/public exponents (d, e) which are related to (δ, ϵ) in a secret way that the author of the generation scheme may invert. From public knowledge of (e, n) only, the author may recover ϵ , break the easy instance (ϵ, n) and discover δ . Factorization of n is easily obtained using standard techniques, once δ, ϵ are known.

Our next two schemes rely on the more recent attacks on RSA small public exponents given parts of the private exponent by Boneh, Durfee and Frankel. The schemes create a random weak pair of private/public exponents (δ, ϵ) with small ϵ , and transforms this ϵ and parts of the corresponding δ into a random looking private/public exponents (d, e) which are related to (δ, ϵ) in a secret way that the author of the generation scheme may invert. From public knowledge of (e, n) only, the author may recover ϵ and parts of δ , break the easy instance (ϵ, n) given parts of δ and discover the whole δ . Factorization of n can easily be obtained using standard techniques, once δ, ϵ are known.

Our last scheme relies on the possibility of hiding at least half the bits of p in the representation of n . Factoring n is possible using Coppersmith's method once half the bits of p are recovered.

Our first scheme generates (d, e) pairs such that $|e| \approx |n|$. However, our second scheme generates (d, e) pairs such that $|e| \approx |n|/2$ while the third generates (d, e) pairs such that $|e| \approx |n|/4$. Our last scheme generates (d, e) pairs for any e of arbitrary size.

This suggests that nobody should rely on RSA key generation schemes provided by a third party. This is most striking in the smartcard model, unless some guarantees are provided that all such attacks to key generation cannot have been embedded. Restriction to RSA moduli with predetermined portions as proposed by Lenstra would not be of any help to prevent our methods. Even in software implementations, unless the actual source code is provided, it is non trivial to find out exactly what the key generation mechanism is. One can easily imagine that software companies who want to keep their code secret for market advantage will not make it easy to decompile their programs to make sense of their implementation know-how.

2 Reminders and Relation to Other Work

The RSA cryptosystem and digital signature schemes [10] are based on the generation of two random primes p, q of roughly equal size and generation of random exponents d, e such that $de \equiv 1 \pmod{\phi(n)}$, where $n = pq$.

Algorithm 2.1 (RSA key generation)

```

1: Pick random primes  $p, q$  of the right size, let  $n := pq$  be a  $k$ -bit integer.
2: repeat
3:   Pick a random odd  $e$  such that  $|e| \leq k$ .
4:   until  $\gcd(e, \phi(n)) = 1$ .
5:   Compute  $d := e^{-1} \pmod{\phi(n)}$ .
6: return( $p, q, d, e$ ).

```

The pair (n, e) may be made publicly available so that the function $m^e \pmod{n}$ be used for encryption, whereas $c^d \pmod{n}$ be used for decryption of messages m, c , $1 \leq m, c \leq n - 1$. Similarly, the private function $m^d \pmod{n}$ may be used to produce a signature c , whereas $c^e \pmod{n}$ may be compared to m as a signature verification procedure of messages m, c , $1 \leq m, c \leq n - 1$.

Wiener [13] demonstrated that small private exponents may be efficiently recovered if $d < n^{25}/3$ and this result was recently improved by Boneh and Durfee [1] who showed a similar result for $d < n^{292}$. Moreover, it is a well known fact [8] that given a multiple of $\phi(n)$ such as $de - 1$ satisfying $de \equiv 1 \pmod{\phi(n)}$, it is easy to factor n .

Boneh, Durfee and Frankel [2] recently demonstrated two interesting results allowing to recover the whole of d given a small e , n and parts of d . Let $n = pq$ such that $p/q < 4$ be an RSA moduli. We use the following two theorems from their work to construct our schemes:

Theorem 1 ([2], Theorem 1.2, part 1). *Let t be an integer in the range $[|n|/4, \dots, |n|/2]$ and e be a prime in the range $[2^t, \dots, 2^{t+1}]$. Suppose we are given (n, e) , and the t most significant bits of d . Then we can compute the whole of d and factor n in time $\text{poly}(|n|)$.*

Theorem 2 ([2], Theorem 4.6). *Let t be an integer in the range $[1, \dots, |n|/2]$ and e be an integer in the range $[2^t, \dots, 2^{t+1}]$. Suppose we are given (n, e) , the t most significant bits of d , and the $|n|/4$ least significant bits of d . Then we can factor n in time $\text{poly}(|n|)$.*

Some quite interesting results by Joye, Paillier and Vaudenay [6] may be used to speed up prime generation at Step 1 of the key generation protocols of section 3 since the primes are not chosen according to particular rules such as those in Section 5. Another useful sequence of result is the proof by Rivest and Shamir [9] that the $|n|/3$ most significant bits of p are sufficient to factor n efficiently, and an improvement due to Coppersmith [3] reducing the number of required bits to $|n|/4$.

De Weger [5] and Slakmon [11] have considered the algorithm of Wiener in a context where the primes p, q are partially known. In particular, we use the following result of Slakmon :

Theorem 3 ([11], Proposition 3.2.1). *Let t be an integer in the range $[1, \dots, |n - \phi(n)|]$ and d be an integer in the range $[1, \dots, 2^{|n - \phi(n)| - t/2}]$. Suppose we are given (n, e) , and the $|n - \phi(n)| - t$ most significant bits of $n - \phi(n)$. Then we can factor n in time $\text{poly}(|n|)$.*

Our schemes are in the line of work by Young and Yung [14, 15, 16] on kleptography. Our schemes are different from theirs and involve a minimal amount of extra calculations to maintain the key generation time roughly the same as an honest RSA key generation scheme. Note however that our schemes of Section 5 are very similar to the PAP method of [14]. Although the PAP scheme would be foiled by the methods of Lenstra [7] to force certain bits of n to be chosen by the user, our schemes will resist to these countermeasures.

2.1 The Scenario

We assume that a legitimate user (called the distinguisher) is given access to the RSA key generation process as a black-box, where he is not allowed to see the code of the generator. However he can sample output tuples (e, d, p, q) from the generator to his will.

The source code of a valid RSA key generator is provided to the distinguisher as well as the code of our cheating generators, except for a secret key, unspecified in the code. The task of the distinguisher is to figure out which is which only from sample outputs of these generators and from running time analysis.

Our goal is to make this distinguishing task as difficult as possible with the simplest and most efficient modification to the standard key generation mechanism. To make this task more significant, we allow the distinguisher extra powers not usually provided to users of such key generation schemes. Our distinguisher may

1. keep one prime out of p, q and request the other one afresh
2. keep the primes p, q and request several valid pairs of exponents d, e .

We believe that despite these extra powers, our cheating mechanisms remain indistinguishable in output distribution and in (approximate) running time.

Notations We use the column “:” for concatenation. Let $m|_\ell$ be the ℓ least significant bits of integer m , and similarly $m|^{\ell}$ be the ℓ most significant bits of integer m .

3 Hidden Exponent Key Generation Algorithms

3.1 Hidden Small Private Exponent δ

The basis of our “cheating” RSA-HSD $_{\beta}$ key generation scheme is to imbed a backdoor β in the scheme and use it to hide instances of small values of the private exponent δ . This is done as described in the following protocol, using an unspecified permutation π_{β} of odd integers smaller than n to themselves. Remember that $|n|$ is a k -bit integer. We discuss several classes of simple choices for π_{β} in Section 4.

Algorithm 3.1 (RSA-HSD $_{\beta}$ key generation)

- 1:** Pick random primes p, q of the appropriate size, set $n := pq$.
- 2: repeat**
- 3:** Pick a random odd δ such that $\gcd(\delta, \phi(n)) = 1$ and $|\delta| \leq k/4$.
- 4:** Compute $\epsilon := \delta^{-1} \bmod \phi(n)$; $e := \pi_{\beta}(\epsilon)$.
- 5: until** $\gcd(e, \phi(n)) = 1$.
- 6:** Compute $d := e^{-1} \bmod \phi(n)$.
- 7:** return(p, q, d, e).

The instances produced by the above key generation scheme satisfy all properties required except for the fact that d, e are not entirely random, but are only random within a smaller set of possibilities specified by the images through π_{β} of inverses modulo $\phi(n)$ of small exponents δ . Notice that Step 3 may be made non-repetitive at the price of biasing the distribution of the δ ’s toward the smaller ones by setting $\delta := \delta / \gcd(\delta, \phi(n))$.

In terms of running time, this algorithm compares very well with the standard RSA key generation Algorithm 2.1: the running time of Steps 1 and 6 are identical to the original whereas the loop from Steps 2 to 5 will run a number of times roughly equal to the original loop; the number of gcd calculations inside the loop is about three times as much as the original (that is if we use the non-repetitive trick suggested above). As long as the computation of $\pi_{\beta}(\epsilon)$ is negligible with respect to calculations such as gcds, the difference in running time may be made quite negligible.

Once n, e are made public n may be factored as follows, given the secret backdoor β .

Algorithm 3.2 (RSA-HSD $_{\beta}$ attack (n, e))

- 1:** Given (n, e) , compute $\epsilon := \pi_{\beta}^{-1}(e)$.
- 2:** Compute δ from (n, ϵ) using Wiener’s low exponent attack.
- 3:** Given (ϵ, δ) factor n as p, q .
- 4:** return(p, q).

At extra cost in the attack, larger values of δ may be used, up to $n^{0.292}$ using the Boneh-Durfee cryptanalytic attack instead of Wiener’s.

The main drawback of this method is that the generated e 's will have roughly full size $|e| \approx |n|$. This means that any restriction on the size of e ($|e| < c|n|$ for $c < 1$) would foil the attack.

3.2 Hidden Small Prime Public Exponent ϵ

The basis of our “cheating” RSA-HSPE $_{\beta}$ key generation scheme is to imbed a backdoor β in the scheme and use it to hide instances of small values of the public prime exponent ϵ together with some partial information about the corresponding private exponent δ . This is done as described in the following protocol, using an unspecified permutation π_{β} of odd integers smaller than n . Remember that $|n|$ is a k -bit integer. We discuss several classes of simple choices for π_{β} in Section 4.

Algorithm 3.3 (RSA-HSPE $_{\beta}$ key generation)

```

1: Pick random primes  $p, q$  of the appropriate size, set  $n := pq$ .
2: repeat
3:   Pick a prime  $\epsilon$  such that  $\gcd(\epsilon, \phi(n)) = 1$  and  $|\epsilon| = k/4$ .
4:   Compute  $\delta := \epsilon^{-1} \bmod \phi(n)$ ;  $\delta_H := \delta^{\lceil \frac{k}{4} \rceil}$ ;  $e := \pi_{\beta}(\delta_H : \epsilon)$ .
5: until  $\gcd(e, \phi(n)) = 1$ .
6: Compute  $d := e^{-1} \bmod \phi(n)$ .
7: return( $p, q, d, e$ ).

```

The instances produced by the above key generation scheme satisfy all properties required except for the fact that d, e are not entirely random, but are only random within a smaller set of possibilities specified by the images through π_{β} of concatenations of δ_H, ϵ of small prime public exponents ϵ .

The size of the concatenations $(\delta_H : \epsilon)$ produced are $k/2$. Therefore, using extra random padding, we have freedom to generate exponents e in the range $\sqrt{n} < e < \phi(n)$.

In terms of running time, this algorithm compares poorly with the standard RSA key generation Algorithm 2.1: on one hand, the running time of Steps 1 and 6 are identical to the original while the loop from Steps 2 to 5 will run a number of times roughly equal to the original loop; the number of gcd calculations inside the loop is about three time as much as the original (since ϵ is prime). On the other hand, unfortunately, generating prime ϵ is going to be time consuming despite the fact that $|e| = |n|/4$.

The main advantage of this method is its simplicity and the fact that the keys produced may be as small as $|n|/2$.

Once n, e are made public n may be factored as follows, given the secret backdoor β .

Algorithm 3.4 (RSA-HSPE_β attack (n, e))

- 1:** Given (n, e) , compute $(\delta_H : \epsilon) := \pi_{\beta}^{-1}(e)$.
- 2:** Compute δ from (n, δ_H, ϵ) using BDF low public prime exponent attack (Theorem 1) with partial knowledge of private exponent.
- 3:** Given (ϵ, δ) factor n as p, q .
- 4:** return(p, q).

3.3 Hidden Small Public Exponent ϵ

The basis of our “cheating” RSA-HSE_β key generation scheme is to imbed a backdoor β in the scheme and use it to hide instances of small values of the public exponent ϵ together with some partial information about the corresponding private exponent δ . This is done as described in the following protocol, using an unspecified permutation π_{β} of odd integers smaller than n to themselves. Remember that $|n|$ is a k -bit integer. We discuss several classes of simple choices for π_{β} in Section 4. Let t be an integer in the range $[1, \dots, |n|/2]$.

Algorithm 3.5 (RSA-HSE_β key generation)

- 1:** Pick random primes p, q of the appropriate size, set $n := pq$.
- 2: repeat**
- 3:** Pick a random ϵ such that $\gcd(\epsilon, \phi(n)) = 1$ and $|\epsilon| = t$.
- 4:** $\delta := \epsilon^{-1} \bmod \phi(n)$; $\delta_H := \delta^{\lceil t \rceil}$; $\delta_L := \delta \lfloor \frac{k}{4} \rfloor$; $e := \pi_{\beta}(\delta_H : \delta_L : \epsilon)$.
- 5: until** $\gcd(e, \phi(n)) = 1$.
- 6:** Compute $d := e^{-1} \bmod \phi(n)$.
- 7:** return(p, q, d, e).

The instances produced by the above key generation scheme satisfy all properties required except for the fact that d, e are not entirely random, but are only random within a smaller set of possibilities specified by the images through π_{β} of concatenations of $\delta_H, \delta_L, \epsilon$ of small public exponents ϵ . Notice that Step 3 may be made non-repetitive at the price of biasing the distribution of the ϵ ’s toward the smaller ones by setting $\epsilon := \epsilon / \gcd(\epsilon, \phi(n))$.

However, to avoid detection, it is necessary to randomize (or discard) the $\ell \geq 2$ least significant bits of δ_L where $2^\ell \mid \phi(n)$ and $2^{\ell+1} \nmid \phi(n)$. This is because $\epsilon \delta \equiv 1 \pmod{2^\ell}$ and therefore $\delta \rfloor_\ell$ is always the inverse modulo 2^ℓ of $\epsilon \rfloor_\ell$. All such obvious redundancy must be removed in order to allow the permutation π_{β} to remain simple and fast to compute. Another such example is that the most significant bits of δ_H are not uniformly distributed because they come from an integer modulo n , where n is publicly known. It is certainly better to specify δ_H as a binary sequence describing the position of δ with respect to n (as in a binary search between 1 and n).

In general, if we request at Step 3 that $|\epsilon| = t$ for $t \in [1, \dots, k/2]$, the total size of the concatenated input $(\delta_H, \delta_L, \epsilon)$ is $2t + k/4$. Asymptotically if we set $t := \gamma k$ for some small $\gamma > 0$ the size is $(1/4 + 2\gamma)k \approx k/4$. Therefore, using extra random padding or by using a larger t , we have freedom to generate exponents e in the range $\sqrt[4]{n} < e < \phi(n)$. Notice however that despite the fact that γ vanishes asymptotically, one should make sure that γk be at least, say 80, to prevent brute force attacks.

In terms of running time, this algorithm compares very well with the standard RSA key generation Algorithm 2.1: the running time of Steps 1 and 6 are identical to the original whereas the loop from Steps 2 to 5 will run a number of times roughly equal to the original loop; the number of gcd calculations inside the loop is about three time as much as the original (that is if we use the non-repetitive trick suggested above). As long as the computation of $\pi_\beta(\epsilon)$ is negligible with respect to calculations such as gcds, the difference in running time may be made quite negligible.

Once n, e are made public, n may be factored as follows, given the secret backdoor β .

Algorithm 3.6 (RSA-HSE _{β} attack (n, e))

- 1: Given (n, e) , compute $(\delta_H : \delta_L : \epsilon) := \pi_\beta^{-1}(e)$.
- 2: Compute δ from $(n, \delta_H, \delta_L, \epsilon)$ using BDF low public exponent attack (Theorem 2) with partial knowledge of private exponent.
- 3: Given (ϵ, δ) factor n as p, q .
- 4: return (p, q) .

4 Choices of π_β

Our main simple and very easy to compute permutation is

$$\pi_\beta(x) = x \oplus (2\beta)_{\lfloor |x| \rfloor}.$$

It appears sufficient for schemes of Section 3 in the sense that the instance spaces are sufficiently large that even if a distinguisher tries to discover the fact that our schemes have been used they will likely fail. In scheme RSA-HSD for instance, the distinguisher is able to compute the XOR of ϵ 's corresponding to small δ 's by computing the XOR of the corresponding e 's, but those will look very random.

Similarly in scheme RSA-HSE, XORing e 's together is likely to look very random as long as they have been processed as described above, to remove obvious redundancy. In scheme RSA-HSPE, the XOR of e 's will yield the XOR of primes ϵ 's but this too is random enough to be indistinguishable.

4.1 More Examples

Of course one can always rely on different cryptosystems to generate efficient π 's, for instance

$$\pi_\beta(x) = \text{DES}_\beta(x) \text{ or } \pi_\beta(x) = \text{AES}_\beta(x)$$

may be used. However it seems more desirable to use permutations π that use the same kind of arithmetic as RSA itself since these operation are readily available for normal key generation. It may be a problem in a smartcard scenario to use program space to implement DES or AES.

When working with fixed sizes k bit information to permute, a very simple way to create randomization is to choose β as a prime in the range $2^{k-1} \pm 2^{k/2}$ and then define

$$\pi_\beta(x) = x^{-1} \bmod \beta$$

which is computed with a single extended gcd calculation.

4.2 Permutations Using Operations Modulo $n + 1$

Another example uses an even translation of the odd exponents x modulo an even number such as $n + 1$. Let N be an upper bound on all the n 's produced by the key generation scheme, and let β be a fixed parameter, picked at random such that $N \leq \beta \leq 2N$. The permutation

$$\pi_\beta(x) = (x + 2\beta) \bmod (n + 1)$$

maps the odd integers modulo $n+1$ to themselves. Notice that the probability that this permutation maps an element to a value greater than $\phi(n)$ is negligible, since $(n + 1) - \phi(n) = p + q$ which is exponentially small with respect to $\phi(n)$. This permutation is different for each n and so makes it even harder to notice the cheat. However, this specific permutation is not a good choice in the context of Section 3.1 since Vaudenay [12] found a way to identify our RSA-HSD generated keys within 24 hours of posting of our proposal on the web [4] !

This can be generalized in several directions using several extra hidden parameters. First notice that $n + 1 - 2\sqrt{n}$ is always an upper bound on $\phi(n)$ and thus

$$\pi_{\beta,\mu}(x) = (x + 2\beta) \bmod (n + 1 - 2m)$$

may also be used, where $m := \mu \bmod \left\lfloor \sqrt{n} \right\rfloor$ for any fixed μ , such that $\sqrt{N} \leq \mu \leq 2\sqrt{N}$. In other words, μ is an arbitrary constant at least half the size of the largest n 's we want to generate.

Notice, however, that generalizing expression $e + 2\beta$ to affine functions using yet another secret parameter α , $1 \leq \alpha \leq N$:

$$\pi_{n,\alpha,\beta,\mu}(e) = ((2\alpha + 1)e + 2\beta) \bmod (n + 1 - 2m)$$

will cause a problem if $\gcd(2\alpha + 1, \phi(n), n + 1 - 2m) > 2$. In this case, given two different sets of exponents, but with the same modulus (p, q, e, d) and (p, q, e', d') , a user could compute $e' - e$ which is the same as $(2\alpha + 1)(e' - e)$ up to a multiple of $(n + 1 - 2m)$. The user could notice that $\gcd(e' - e, \phi(n)) > 2$ all the time which is unusual, despite the fact that μ and β are unknown.

4.3 Discussion: Avoiding and Securing Hidden Exponent Attacks

Of course, a very simple strategy will foil these attacks : make sure d (or e) is picked from an exponentially large subset S of possible values which is only an exponentially small fraction of all the d 's, such that $\gcd(d, \phi(n)) = 1$. For example, consider the set of valid private exponents to be $S = \{d | \gcd(d, \phi(n)) = 1 \text{ and } d < \sqrt{n}\}$. It seems quite unlikely that one can find a simple permutation $\pi_\beta(e)$ that frequently maps inverses modulo $\phi(n)$ of d 's, $d < n^{1/4}$, to inverses inverses modulo $\phi(n)$ of d 's in S , in a random looking fashion. But, who knows really if such a thing is impossible ?

Bad choices of S can be no help to foil the attack. For example, $S = \{d | \gcd(d, \phi(n)) = 1 \text{ and } (d^{-1} \bmod \phi(n)) < \sqrt{n}\}$ is easily foiled by methods of Sections 3.2 and 3.3.

Alternatively, forcing some redundancy onto d 's may help foil the attack. For instance, $S = \{d | \gcd(d, \phi(n)) = 1 \text{ and } d = (x : x)\}$ where x is a half size odd number, seems a good countermeasure to our hidden exponent schemes.

The indistinguishability of resulting schemes depend extensively on the permutation chosen but in many cases the simpler ones seem to suffice. Notice also that the security of our three schemes decrease as they produce e 's of smaller and smaller size. Indeed, even more relevant is the fact that the set of possible e 's gets smaller from the first to the third scheme. The size of the valid e 's set is an important factor of security.

Finally, notice that the last of the three schemes, which produces the smallest e 's may involve extra weaknesses similar to those already exposed because of the redundancy of the lsb's of ϵ and the corresponding lsb's of δ . Further analysis of this redundancy should be performed. It could otherwise lead to countermeasures.

5 Hidden Prime Factor

Our last proposal is very similar to the PAP (Pretty-Awful-Privacy) of [14] but we address and solve a number of deficiencies left unnoticed or unresolved by their scheme. We investigate the idea of imbedding some bits of the prime factor p in the product $n = pq$, thus choosing q to be a special prime satisfying a number of constraints. However, as long as the key β remains secret it should be hard to tell from the distribution of p, q, n 's produced that cheating is going on. Our proposal differs from PAP in two major ways:

- After [3] we only hide the half most significant bits of p
- We make sure the distribution of numbers n, p, q is similar to the honest one.

The PAP method generates numbers n where the most significant bits are uniformly distributed which is not the proper distribution for products of two randomly selected (prime) integers of a fixed size. For instance, if one picks two random (prime) integers of 512 bits each, their product will be 1023 bits long with probability 38% whereas with probability 48% it will be 1024 bits long with leading bits “10” and with probability 14% it will be 1024 bits long with leading bits “11”. This issue was ignored by [14]. Of course this problem does not happen if p and q are picked over the more appropriate interval $[\sqrt{2} \times 2^{511}, \dots, 2^{512} - 1]$, but we see that the distribution of p, q influences the distinguishability of the result.

Let e be some fixed public exponent such as 3, 17, 65537, etc, for which an appropriate n must be found. Our scheme RSA-HP $_{\beta}$ proceeds as follows:

Algorithm 5.1 (RSA-HP $_{\beta}(e)$ key generation)

- 1:** Pick a random prime p of the right size, s.t. $\gcd(e, p - 1) = 1$.
- 2:** Pick a random odd q' of the appropriate size, set $n' := pq'$.
- 3:** Compute $\tau := n'^{\frac{k}{8}}$, $\mu := \pi_{\beta}(p^{\frac{k}{4}})$ and $\lambda := n'^{\lfloor \frac{5k}{8} \rfloor}$.
- 4:** Set $n := (\tau : \mu : \lambda)$ and $q := \lfloor n/p \rfloor + (1 \pm 1)/2$ so that it is odd.
- 5:** **while** $\gcd(e, q - 1) > 1$ or q is composite **do**
 - Pick a random even m such that $|m| = \frac{k}{8}$,
 - Set $q := q \oplus m$ and $n := pq$.
- 6:** Compute $d := e^{-1} \bmod \phi(n)$.
- 7:** **return**(p, q, d, e).

The instances produced by the above key generation scheme is the product of a truly random p and a somewhat random q such that

- the top $k/8$ bits of n have the correct distribution of such a product
- the next $k/4$ bits of n are an “encryption” of the $k/4$ most significant bits of p
- the least $k/8$ bits of q are randomly chosen so that q is prime.

The running time of the above algorithm is more or less the same as the standard algorithm where p and q are individually picked at random until they are prime. Prime p is produced exactly in the same way, whereas prime q is picked according to the method of Step 5, which on average takes the same number of steps as Step 1. All other extra computations are negligible with respect to a single primality test.

Algorithm 5.2 (RSA-HP $_{\beta}$ attack (n, e))

- 1:** Given n , compute $p^{\frac{k}{4}} := \pi_{\beta}^{-1}(n^{\lceil \frac{3k}{8} \rceil} \lfloor \frac{k}{4} \rfloor)$.
- 2:** Factor n as p, q using Coppersmith’s partial information attack.
- 3:** **return**(p, q).

Unlike the methods of Section 3, the simple permutation $\pi_\beta(x) = x \oplus (2\beta)_{|x|}$ is definitely not secure; upon receiving two pairs (p, q) and (p', q') generated as above, one can easily check that

$$(n' \oplus n)^{\lceil \frac{3k}{8} \rceil}_{\lfloor \frac{k}{4} \rfloor} = (p' \oplus p)^{\lceil \frac{k}{4} \rceil}$$

which should not happen normally. For this method we recommend permutations from Sections 4.1 and 4.2. Again the last permutation of Section 4.1 $\pi_\beta(x) = x^{-1} \bmod \beta$ is definitely not secure by itself; upon receiving a pair (p, q) generated as above, one can easily compute

$$n^{\lceil \frac{3k}{8} \rceil}_{\lfloor \frac{k}{4} \rfloor} p^{\lceil \frac{k}{4} \rceil} - 1$$

which should be a multiple of the secret prime β . Running this experiment several times will lead to several multiples of β and a simple gcd calculation will yield β . Notice however that if $p^{\lceil \frac{k}{4} \rceil}$ is padded with a large enough number of extra random bits, the above attack is foiled.

Our favorite permutation is computing of a modular inverse mod a fixed predetermined prime near $2^{\frac{k}{4}}$ as proposed at the end of Section 4.1 and XORing with a fixed string:

$$\pi_{\beta,\mu}(x) = \left(x \oplus (2\mu)_{|x|} \right)^{-1} \bmod \beta \quad \text{or} \quad \pi_{\beta,\mu}(x) = (x^{-1} \bmod \beta) \oplus (2\mu)_{|\beta|}$$

which seem to foil both attacks presented above.

The unfortunate drawback of this method is that, while the first prime p can be picked according to any rule, the second prime q is picked according in a way that will not modify too much of its bits, once a first approximation is found. This opens the door to some attacks that may use this deficiency. Also it requires that a more elaborate encryption be used to hide the half of p in n . If by accident a “bad” prefix is selected, the amount of attempts to reach a prime may be very high. However on average the number of such attempts is the same as standard generation. Moreover we must find a valid portion of n that is uniformly distributed as long as p and q are picked according to any particular distribution. In the above example, we make the assumption that despite the fact that the first few and last bits of n are not uniformly distributed, the $k/4$ positions past the first $k/8$ are.

5.1 Combining with Exponent Method

Using the result of Slakmon (Theorem 3) we combine the above method with the hidden small exponent method with larger exponents ! For instance, if n is used to subliminally transmit the $|n|/6 \approx |n - \phi(n)| - |n|/3$ most significant bits of $n - \phi(n)$ (instead of $|n|/4$ msb’s of p required by Coppersmith’s method), then a variation of Wiener’s method is able to recover hidden exponents d up to size $|n - \phi(n)| - |n|/3/2 \approx |n|/3$ which is better than both Wiener and Boneh-Durfee methods. Thus exponents d up to $n^{0.333}$ may be used and broken.

Increasing the size of acceptable d 's increases the security of the hidden small exponent method and reducing the amount of information transmitted subliminally through n increases the security of the hidden prime method. However, the resulting method suffers the deficiencies of both methods.

5.2 Discussion: Avoiding and Securing Hidden Prime Attacks

Providing constraints in the way of Lenstra's work [7] do not seem to be an effective way of stopping the attack. Our method offers sufficient freedom in the choice of p and q that fixing the msb's or lsb's of n is not strong enough to foil our attack.

However, if we try to reduce the running time to be comparable to the primes generation algorithm of Joye, Paillier and Vaudenay [6] our method falls short because we could not find a way to generate p and q efficiently using their method while subliminally sending through n some portions of the bits of p .

An important issue to secure the hidden prime methods is to identify parts of n that are sufficiently uniformly distributed when p and q are picked from their respective distributions. Making very strict restrictions on the distributions of p and q can make that task very difficult.

6 Conclusions

We have introduced in Sections 3 and 5 a variety of *simple* backdoors that can be used to generate apparently normal RSA keys (p, q, d, e) in such a way that the owner of a secret key imbedded in the generation scheme may recover the private primes p and q from the public information e, n . Each of these schemes use (n, e) as a subliminal channel to carry special information useful to factor n . The security of the subliminal channel is parametrized by the choice of a secret mapping π_β . Several possibilities of π_β were proposed in Section 4.

We are well aware that no proof of security of our schemes have been provided or even hinted. Indeed, introducing a backdoor is somewhat like introducing a new computational assumption. Only time will tell whether these backdoors resist to cryptanalysis. The scheme of Section 3.1 is the only one that has been made public so far (on a web page) and has survived cryptanalysis for more than a year.

We challenge the cryptology community to break the several schemes/ permutations possibilities proposed in this paper.

Acknowledgements

We thank Dan Boneh, Don Coppersmith, Jean-Marc Robert, Serge Vaudenay, and Moti Yung for helpful discussions and for breaking some early schemes. We are grateful to Martin Courchesne and Simon Wong for their web programming assistance.

References

- [1] D. BONEH AND G. DURFEE, *Cryptanalysis of RSA with private key d less than $n^{0.292}$* , Information Theory, IEEE Transactions on, 46 (2000), pp. 1339–1349. 405
- [2] D. BONEH, G. DURFEE, AND Y. FRANKEL, *An attack on RSA given a small fraction of the private key bits*, in Advances in Cryptology - AsiaCrypt '98, K. Ohta and D. Pei, eds., Berlin, 1998, Springer-Verlag, pp. 25–34. Lecture Notes in Computer Science Volume 1514. 405
- [3] D. COPPERSMITH, *Finding a small root of a bivariate integer equation; factoring with high bits known*, in Advances in Cryptology - EuroCrypt '96, U. Maurer, ed., Berlin, 1996, Springer-Verlag, pp. 178–189. Lecture Notes in Computer Science Volume 1070. 405, 412
- [4] C. CRÉPEAU AND S. WONG, *The RSA hidden small exponent method*, in <http://crypto.cs.mcgill.ca/~crepeau/RSA>, 2001. 411
- [5] B. DE WEGER, *Cryptanalysis of RSA with small prime difference*, Applicable Algebra in Engineering, Communication and Computing, 13 (2002), pp. 17–28. 406
- [6] M. JOYE, P. PAILLIER, AND S. VAUDENAY, *Efficient generation of prime numbers*, in CHES 2000, Ç. K. Koç and C. Paar, eds., Berlin, 2000, Springer-Verlag, pp. 340–354. Lecture Notes in Computer Science Volume 1965. 405, 415
- [7] A. K. LENSTRA, *Generating RSA moduli with a predetermined portion*, in Advances in Cryptology - AsiaCrypt '98, K. Ohta and D. Pei, eds., Berlin, 1998, Springer-Verlag, pp. 1–10. Lecture Notes in Computer Science Volume 1514. 406, 415
- [8] G. L. MILLER, *Riemann's hypothesis and tests for primality*, J. Comput. System Sci., 13 (1976), pp. 300–317. 405
- [9] R. L. RIVEST AND A. SHAMIR, *Efficient factoring based on partial information.*, in Advances in Cryptology - EuroCrypt '85, F. Pichler, ed., Berlin, 1985, Springer-Verlag, pp. 31–34. Lecture Notes in Computer Science Volume 219. 405
- [10] R. L. RIVEST, A. SHAMIR, AND L. M. ADLEMAN, *A method for obtaining digital signatures and public-key cryptosystems*, Comm. ACM, 21 (1978), pp. 120–126. 405
- [11] A. SLAKMON, *Sur des méthodes et algorithmes de factorisation et leur application en cryptologie*, Master's thesis, Université de Montréal, dépt. IRO, 2000. 406
- [12] S. VAUDENAY, *Private e-mail communication.*, 2 may 2001. 411
- [13] M. WIENER, *Cryptanalysis of short RSA secret exponents*, Information Theory, IEEE Transactions on, 36 (1990), pp. 553–558. 405
- [14] A. YOUNG AND M. YUNG, *The dark side of “black-box” cryptography, or: Should we trust Capstone?*, in Advances in Cryptology - Crypto '96, N. Koblitz, ed., Berlin, 1996, Springer-Verlag, pp. 89–103. Lecture Notes in Computer Science Volume 1109. 406, 412, 413
- [15] ———, *Kleptography: Using cryptography against cryptography*, in Advances in Cryptology - EuroCrypt '97, W. Fumy, ed., Berlin, 1997, Springer-Verlag, pp. 62–74. Lecture Notes in Computer Science Volume 1233. 406
- [16] ———, *The prevalence of kleptographic attacks on discrete-log based cryptosystems*, in Advances in Cryptology - Crypto '97, B. Kaliski, ed., Berlin, 1997, Springer-Verlag, pp. 264–276. Lecture Notes in Computer Science Volume 1294. 406