Colin Boyd
Juan González Nieto (Eds.)

LNCS 5594

Information Security and Privacy

14th Australasian Conference, ACISP 2009
Brisbane, Australia, July 2009
Proceedings

Springer

Lecture Notes in Computer Science 5594

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

Colin Boyd Juan González Nieto (Eds.)

Information Security and Privacy

14th Australasian Conference, ACISP 2009
Brisbane, Australia, July 1-3, 2009
Proceedings

Springer

Volume Editors

Colin Boyd
Juan González Nieto
Queensland University of Technology
Information Security Institute
GPO Box 2434, Brisbane, QLD 4001, Australia
E-mail: {c.boyd, j.gonzaleznieto}@qut.edu.au

Library of Congress Control Number: 2009930749

CR Subject Classification (1998): E.3, K.6.5, D.4.6, C.2, E.4, F.2.1, K.4.1

LNCS Sublibrary: SL 4 – Security and Cryptology

ISSN 0302-9743
ISBN-10 3-642-02619-2 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02619-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12707515 06/3180 5 4 3 2 1 0

Preface

The 2009 Australasian Conference on Information Security and Privacy was the 14th in an annual series that started in 1996. Over the years ACISP has grown from a relatively small conference with a large proportion of papers coming from Australia into a truly international conference with an established reputation. ACISP 2009 was held at Queensland University of Technology in Brisbane, during July 1–3, 2009.

This year there were 106 paper submissions and from those 30 papers were accepted for presentation, but one was subsequently withdrawn. Authors of accepted papers came from 17 countries and 4 continents, illustrating the international flavor of ACISP. We would like to extend our sincere thanks to all authors who submitted papers to ACISP 2009.

The contributed papers were supplemented by two invited talks from eminent researchers in information security. Basie von Solms (University of Johannesburg), currently President of IFIP, raised the question of how well dressed is the information security king. L. Jean Camp (Indiana University) talked about how to harden the network from the friend within. We are grateful to both of them for sharing their extensive knowledge and setting challenging questions for the ACISP 2009 delegates.

We were fortunate to have an energetic team of experts who formed the Program Committee. Their names may be found overleaf, and we thank them warmly for their considerable efforts. This team was helped by an even larger number of individuals who reviewed papers in their particular areas of expertise. A list of these names is also provided which we hope is complete. We would like to express our thanks to Springer for continuing to support the ACISP conference and for help in the conference proceedings production.

We are delighted to acknowledge the generous financial sponsorship of ACISP 2009 by the Research Network for a Secure Australia (funded by the Australian Research Council). The conference was hosted by the Information Security Institute at Queensland University of Technology, who provided first-class facilities and material support. The excellent Local Organizing Committee was led by the ACISP 2009 General Chair, Ed Dawson, with key contributions from Elizabeth Hansford and Christine Orme. We made use of the iChair electronic submission and reviewing software written by Thomas Baignères and Matthieu Finiasz at EPFL, LASEC.

July 2009

Colin Boyd
Juan González Nieto

Organization

General Chair

Ed Dawson	Queensland University of Technology, Australia

Program Co-chairs

Colin Boyd	Queensland University of Technology, Australia
Juan González Nieto	Queensland University of Technology, Australia

Program Committee

Michel Abdalla	École Normale Supérieure, France
Tuomas Aura	Microsoft Research, UK
Feng Bao	Institute for Infocomm Research, Singapore
Lynn Batten	Deakin University, Australia
Mike Burmester	Florida State University, USA
Andrew Clark	Queensland University of Technology, Australia
Marc Dacier	Symantec Research Labs Europe, France
Sabrina De Capitani di Vimercati	Università degli Studi di Milano, Italy
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Spain
Alain Durand	Thomson, France
Pierre-Alain Fouque	Ecole Normale Supérieure, France
Steven Galbraith	Royal Holloway, UK
Dieter Gollman	Hamburg University of Technology, Germany
Maria Isabel González Vasco	Universidad Rey Juan Carlos, Spain
Kwangjo Kim	Information and Communication University, Korea
Lars Knudsen	Technical University of Denmark, Denmark
Pil Joong Lee	Pohang University of Science and Technology, Korea
Xuejia Lai	Shanghai Jiaotong University, China
Mark Manulis	TU Darmstadt, Germany
Chris Mitchell	Royal Holloway, UK
Atsuko Miyaji	JAIST, Japan
Paul Montague	DSTO, Australia
Yi Mu	University of Wollongong, Australia
Eiji Okamoto	Tsukuba University, Japan
Pascal Paillier	Gemalto, France
Kenny Paterson	Royal Holloway, UK
Josef Pieprzyk	Macquarie University, Australia

Matt Robshaw	Orange Labs, France
Carsten Rudolph	Fraunhofer SIT, Germany
Mark Ryan	University of Birmingham, UK
Rei Safavi-Naini	University of Calgary, Canada
Palash Sarkar	Indian Statistical Institute, India
Ron Steinfeld	Macquarie University, Australia
Douglas Stinson	University of Waterloo, Canada
Willy Susilo	University of Wollongong, Australia
Jorge Villar	Universitat Politècnica de Catalunya, Spain
Huaxiong Wang	Nanyang Technological University, Singapore
Duncan Wong	University of Hong Kong, China

External Reviewers

Davide Alessio	Shaoquan Jiang	Agusti Solanas
Myrto Arapinis	Marc Joye	Rainer Steinwandt
Tomoyuki Asano	Stefan Katzenbeisser	Pairat Thorncharoensri
Mina Askari	Angelos Keromytis	Leonie Simpson
Man Ho Au	Sun Young Kim	Bo Qin
Julia Borghoff	Izuru Kitamura	Rolando Trujillo Rasua
Joo Yeon Cho	Divyan M. Konidala	Jae Woo Seo
Imsung Choi	Gregor Leander	Michal Sramka
Carlos Cid	Hyunrok Lee	Xiaorui Sun
Nico Doettling	Corrado Leita	Tomas Toft
Ming Duan	Benoit Libert	Olivier Thonnard
Dang Nguyen Duc	Xibin Lin	Sungmok Shin
Orr Dunkelman	Hans Loehr	Sren S. Thomsen
Sungwook Eom	Xianhui Lu	Damien Vergnaud
Martin Gagné	Yi Lu	Nguyen Vo
Praveen Gauravaram	Atefeh Mashatan	Zhongmei Wan
David Galindo	Toshihiko Matsuo	Hongjun Wu
Zheng Gong	Krystian Matusiewicz	Mu-En Wu
Choudary Gorantla	Jörn Müller-Quade	Jiang Wu
Fuchun Guo	Hyeran Mun	Qianhong Wu
Jian Guo	Kris Narayan	Wei Wu
Kyusuk Han	Kazuto Ogawa	Zhongming Wu
Francisco Rodríguez	Kazumasa Omote	Xiaokang Xiong
Henríquez	Hyewon Park	Yeon-Hyeong Yang
Matt Henricksen	Vijayakrishnan	Myunghan Yoo
Javier Herranz	Pasupathinathan	Kazuki Yoneyama
Jonathan Hoch	Axel Poschmann	Tsz Hon Yuen
Dennis Hofheinz	Angel L. Perez del Pozo	Greg Zaverucha
Qiong Huang	Thomas Plantard	Lei Zhang
Xinyi Huang	Siamak Shahandashti	Liangfeng Zhang
Tibor Jager	Ben Smyth	Huafei Zhu

Table of Contents

Invited Lecture

Network Security

Symmetric Key Encryption

Hash Functions

Invited Lecture

Public Key Cryptography

Protocols

Implementation

Is the Information Security King Naked?

Basie von Solms

University of Johannesburg
Johannesburg
South Africa
basievs@uj.ac.za

Abstract. As children we probably all often listened to the fable of the king who rode nakedly through the street thinking he wore a beautiful new coat created for him by his (rogue) tailor. Nobody wanted to tell him that he was naked, because they all feared him – until a small boy revealed the truth.

This paper asks whether the IT industry is not in the same position – Information Security wise totally naked - although everyone tells everyone else how secure our IT systems are.

Keywords: Information Security, Information Security Governance, Insecure systems, Naked, Viruses, Social engineering, Ethics.

1 How Well Is the Information Security King Dressed?

In the fable, the main reason why his subordinates did not tell the king that he was naked, was because they wanted to keep him happy – for whatever reason and under all circumstances. They knew perfectly well that what they told him was not true, but that was less important than keeping him happy. Eventually when the truth was exposed – that he was actually naked – probably all his advisors disappeared or claimed ignorance!

This paper tries to investigate the present situation as far as the way the Information Security king is dressed. In this paper we envisaged the Information Security king wanting to attend certain events (functions) and for every function he needs a dress.

The Information Security king is seen as representative of those who rely on us as Information Security professionals to ensure that IT systems are secure – they include customers, clients, patients, bosses, business owners etc. The functions the king wants to attend represent the IT systems used by the king, and the dress is representative of the Information Security capabilities of IT systems developed to be used by the king. We, as information security professionals and practitioners, are representative of the advisors to the king about the quality of his dress for a specific function.

Of course there are many Information Security kings, and many events, and many dresses, but in this paper we investigate the issue in general.

Does the possibility exist that we, as Information Security practitioners and professionals truly believe that the Information Security king is (always) properly

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 1–7, 2009.

(suitably) dressed, without realizing that he is maybe already naked or nearly naked?

Even worse, does the possibility exist that we, as Information Security practitioners and professionals know very well that the Information Security king is already naked, or nearly naked, but we choose not to advise him about this fact?

It is important that we think about these questions, because the Information Security king relies on our advice!

As Information Security professionals we must all agree on one basic fact – perfect Information Security is not possible. In no way can any operational IT system on this planet be perfectly secured. Anyone claiming that is advising the king that he is always regally dressed – fit for a king - while knowing perfectly well that he is at the most well dressed, or even only casually dressed or dressed in a swimming pants or even a g-string!

The real question is then – how good can we secure systems, meaning how well can we actually dress the king?

I trust that this paper will stimulate the growing discussion about the security of IT systems, and the role that we as Information Security professionals should play in ensuring the security of such systems.

Let us investigate some reasons why many, including the author, have a very worrying feeling that at the present time, for many important functions, the Information Security king is (often) very, very scantily dressed.

2 Should There Be a Worry about the Information Security King's Dress?

Do we have reason to be worried? Why is the question, comprising the title of this paper, at all necessary?

Let us investigate a few quotes from publications over the last few years to give us some clue:

'Computers around the world are systematically being victimized by rampant hacking. This hacking is not only widespread, but is being executed so flawlessly that attackers compromise a system, steal everything of value and completely erase their tracks within 20 minutes' [3]

'On the Internet you are always living in a bad neighborhood...' [4]

'Security remains one of the hottest topics in IT today – one that keeps users baffled and petrified, boards of directors tied up in discussions and the poor IT manager awake at night...' [4]

'Consumer Reports recently conducted a survey of more than 3200 US home computer users with Internet access. Results show that these users have a one-in three likelihood of experiencing damage to their computers, financial loss or both because of a computer worm, virus, Trojan horse or spyware.' [5]

'An intruder gained unauthorized access into a ... system, potentially exposing more than 33000 officers to identity theft' [5]

'A computer tape from a Connecticut bank containing personal data on 90,000 customers, including names, addresses, Social Security numbers and checking account numbers, was lost in transit recently.' [6]

'Personal information for 55,000 customers, including bank data and Social Security numbers, has been stolen from a database at the upmarket Atlantis Resort in the Bahamas.' [7]

'In November alone, the Anti-Phishing Working Group (APWG) received 16,882 unique reports of phishing attacks that attempted to fool consumers with 93 different hijacked brands, according to a report the group released this week. November's phishing attacks marked an all-time high, climbing from the 15,820 the group received notice of in October, and doubling the number of attacks recorded in November 2004, according to the report.' [8]

'Since early 2005, more than 150 million personal records have been exposed.' [9]

'RBS WorldPay (formerly RBS Lynk), the U.S. payment processing arm of The Royal Bank of Scotland Group, today announced that its computer system had been improperly accessed by an unauthorized party. ..... Certain personal information of approximately 1.5 million cardholders and other individuals may have been affected and, of this group, Social Security numbers of 1.1 million people may have been accessed.' [10]

'Late last week, Jefferson County Clerk Jennifer Maghan said she unveiled a new online search tool that enabled residents and business professionals to access nearly 1.6 million documents that are stored in her office via their home computers.' [10]

The quotes above are just a very small number of those which are mentioned daily and weekly concerning Information Security issues – there are many more!

We, as Information Security professionals, know (or should know) this. We also know that the Internet as a system has now become so complex and sophisticated, that it is basically impossible to properly secure any IT system using this Internet.

We know that the 'window of vulnerability', ie the time between a vulnerability being announced and the first virus exploiting that vulnerability is let loose, is becoming smaller and smaller (zero-day vulnerability).

The question is what do we tell the Information Security king? It is and stays our responsibility to seriously talk to the king about his dress, and where necessary, tell him directly that he runs the risk of being naked.

This basic 'unsafe (naked) at any time' feeling is not only because of the Internet. One of the major threats to the use of any IT systems has always been, and will always be the fact that such systems are used by humans – that is the weakest link.

The growing threat of specialized social engineering, where the trust of the user is misused to gain important access and other IT related information, is becoming extremely pervasive.

'... gurus agree that, generally, user-focused attacks present perhaps the biggest threat today.' [4]

Knowing all this, and taking all this into account, we as Information Security professionals have no other option than to be (very) worried about the king's dress!

Personally this author feels that the Information Security king is often very close to naked, specifically for specific events, and is worried that many advisors still convince him that he is well dressed for such events. This feeling does, of course, not imply that all IT systems are insecure, or that it is impossible to secure any IT system. It does imply that many new IT systems are

- so complex and 'close to the edge', that such systems should rather not be developed or implemented, or
- implemented and put into production without proper and adequate information security features designed into the system, or
- are put into production without comprehensive testing the security features of the system

History has some form of analogy relevant to the issues discussed above. This is briefly discussed in the next paragraph

3 The Strategic Defense Initiative (SDI) and David Parnas

'The Strategic Defense Initiative (SDI), commonly called Star Wars after the popular science fiction series, was a system proposed by U.S. President Ronald Reagan on March 23, 1983 to use space-based systems to protect the United States from attack by strategic nuclear missiles. It was never implemented and research in the field tailed off after the end of the Cold War.' [2] Prof David Parnas, one of the pioneers in the development of Computer Science and Software Engineering, was at that time a consultant to the Office of Naval Research in Washington, and was one of nine scientists asked by the Strategic Defense Initiative Office to serve on the "panel on computing in support of battle management".

Parnas resigned from this advisory panel on antimissile defense, asserting that it will never be possible to program a vast complex of battle management computers reliably or to assume they will work when confronted with a salvo of nuclear missiles.

In his letter of resignation he said that it would never be possible to test realistically the large array of computers that would link and control a system of sensors, antimissile weapons, guidance and aiming devices, and battle management stations. Nor, he protested, would it be possible to follow orthodox computer program-writing practices in which errors and "bugs" are detected and eliminated in prolonged everyday use.

"I believe," Professor Parnas said, "that it is our duty, as scientists and engineers, to reply that we have no technological magic that will accomplish that. The President and the public should know that." [1]

In 1984 the ACM Council passed and published an important resolution. It begins:

Contrary to the myth that computer systems are infallible, in fact computer systems can and do fail. Consequently, the reliability of computer-based systems cannot be taken for granted. This reality applies to all computer-based systems, but it is especially critical for systems whose failure would result in extreme risk to the public. Increasingly, human lives depend upon the reliable operation of systems such as air traffic and high-speed ground transportation control systems, military weapons delivery and defense systems, and health care delivery and diagnostic systems. [1]

Although Parnas's stand was related to nuclear warfare, which may not be so relevant today anymore, the morale of this story is still the same. Parnas and the ACM highlighted the reliability issues of the use of computers, because they were important issues concerning the man in the street. They clearly told the DSI king that he runs the danger of being totally naked! It is important to note that Parnas did not say that all computer systems are unreliable – he just said that this specific initiative was dangerous. The reader may now say that the SDI issues were related to the reliability of nuclear computer systems, and not to the Information Security of commercial systems. My answer to such a reaction is : 'Is the Information Security of general IT systems today, even though they are now much more business focussed, less important, or less complicated?' Just have a look at paragraph 2 above, or ask a person who lost all his/her money through fraud committed using IT systems whether he sees it as a serious issue or not? Following Parnas's quote above, I want to state :

'it is our duty, as Information Security practitioners to make it heard from all platforms that IT systems are becoming so complex that we doubt whether they can still be properly protected in all cases. The public should know that'

Maybe the ACM and other relevant bodies like IFIP [11] should again take a stance on this issue as the ACM took in 1985.

4 What Should We Do?

We must have the willingness to advise, and keep advising the king, and we must refuse to be part of the tailor's creation if we are convinced that the resulting risk will be too high.

The motor manufacturing industry can build cars which can do 300 kilometer per hour, but they realize that for security reasons, they must govern the speed to acceptable safe levels. Coming back to the statement in paragraph 2 above, about users having a one-in-three likelihood of experiencing damage to their computers, financial loss or both because a computer worm, virus, Trojan horse or spyware. Will we buy a car if we know that the there is a one-in-three chance of brake failure?

We know that we can build extremely powerful and useful IT systems, specifically exploiting the power of the Internet, but the core question must be the security and safety of such systems – we owe that to the man in the street who will use such systems.

Are we not at the point where we as Information Security professionals must start raising our voices about more and more new and complex systems being developed and rolled out – systems which are too complex and dangerous for their own good, and where the eventual loser will most probably be the well trusting user of a system?

I challenge my peers in this field, and IT organizations to start an open discussion about the wardrobe of the Information Security king.

However, there may be another issue related to the question stated above. The statement uses the term 'information security professionals'. Let us briefly investigate this aspect.

5 Information Security Professionals

The reasoning in this paper sofar accepts that we, working in the field of Information Security and who act as advisors to the Information Security king, can call ourselves 'Information Security Professionals'!

Can we really do that? Are we really professionals, as the term is understood in other circles like medicine, engineering etc? Do we belong to a professional body which has a defined Body of Knowledge, an Ethics Code or a Disciplinary Code? Can we be held accountable for the advice we give to the Information Security king?

Do we have a 'true' Information Security Profession which is acknowledged internationally? The answer must be 'NO' at this stage!

There are some bodies which do 'certify' people as Information Security Professionals, and that is already a step in the right direction. However, it does not go far enough, and do not create 'true' Information Security Professionals or a 'true' Information Security Profession.

Other initiatives are developing to create a 'true' Information Technology Profession, but that is wider than Information Security. Notable examples of these initiatives are those by IFIP, through the International Information Technology Professional (IITP) and I3P programs, those by some national IT Societies like the Australian, British, Canadian and South African Computer Societies and the IEEE-CS. We must ensure that these initiatives should also cater for a dedicated 'Information Security Professional'.

In the meantime, we as Information Security practitioners (not yet Information Security professionals) should act extremely responsible in our advice roles.

6 Summary

We, as Information Security practitioners (professionals?) must ask ourselves two questions :

1. Is the Information Security king naked? Personally I think it is not (yet) the case, but I very strongly feel that he is NOT well dressed, and I am worried that we are wary and scared to tell him that.

2. How are we going to organize ourselves into a true Information Security Profession?

Let's talk about this matter – we owe it to the king and the users out there.

References

1. The Risk Digest 1(1) (1985), http://catless.ncl.ac.uk/Risks/1.01.html (accessed April 2009)
2. Wikipedia, http://en.wikipedia.org/wiki/Strategic_Defense_Initiative (accessed April 2009)
3. Winkler, I.: Guard against Titan Rain hackers (2005), http://www.computerworld.com/printthis/2005/0,4814,105585,00.html (accessed April 2009)
4. Haggard, B.: iWeek (December 1, 2005), www.iweek.co.za
5. Schultz, E.: Computers and Security 24(8) (2005)
6. Lawson, S.: ComputerWorld (accessed April 2009), http://cwflyris.computerworld.com/t/242864/94135/5963/0/
7. Nicolai, J.: ComputerWorld (2006), http://cwflyris.computerworld.com/t/242864/94135/5965/0/ (accessed April 2009)
8. Garretson, C.: NETWORK WORLD (2006), http://www.computerworld.com/securitytopics/security/story/0,10801,107747,00.html?source=NLT_SEC&nid=107747 (accessed April 2009)
9. Theft Statistics (2007), http://www.absolute.com/EMEA/computer-theft-statistics-details.asp (accessed April 2009)
10. Theft Centre (2008), http://www.idtheftcenter.org/BreachPDF/ITRC_Breach_Report_2008_final.pdf (accessed April 2009)
11. IFIP, http://www.ifip.org

Measurement Study on Malicious Web Servers in the .nz Domain

Christian Seifert, Vipul Delwadia, Peter Komisarczuk, David Stirling, and Ian Welch

School of Engineering and Computer Science
Victoria Univestity of Wellington
P.O. Box 600, Wellington 6140, New Zealand
{cseifert,vipul,peterk,david.stirling,ian}@ecs.vuw.ac.nz

Abstract. Client-side attacks have become an increasing problem on the Internet today. Malicious web pages launch so-called drive-by-download attacks that are capable to gain complete control of a user's machine by merely having that user visit a malicious web page. Criminals that are behind the majority of these malicious web pages are highly sensitive to location, language and economic trends to increase their return on investment. In this paper, a comprehensive measurement study of malicious web servers on the .nz domain is presented. The risk of drive-by-download attacks has been compared with other domains showing no elevated risk for the .nz domain. However, a comprehensive assessment of the .nz domain showed the existence of malicious web pages across a variety of types of web pages. Blacklisting services showed limited success to protect against such malicious web pages. This is primarily attributed to the highly dynamic nature of malicious web pages. Over a period of eight months, the .nz domain was monitored and continuous shifting of malicious behavior of web pages has been observed. The rates observed show that on average 50% of malicious URLs identified change monthly. The rates pose a challenge to blacklisting services as well as a risk to end users with rapid dissemination of zero-day attacks. Frequent scans of the web are required to obtain a good up-to-date view of the threat landscape.

1 Introduction

Broadband connectivity and the great variety of services offered over the Internet have made it an important source of information and entertainment and a major means of communication. In 2009, users are more connected than ever. With connectivity to the Internet, however, come security threats. Because the Internet is a global network, an attack can be delivered anonymously from any location in the world. Security professionals responding to these threats offer a wide range of mitigation strategies and measures.

As attack vectors are barred by defenses, malicious users seek out new, unprotected paths of attack. One of these is the client-side attack, which targets client applications. As the client accesses a malicious server, the server delivers

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 8–25, 2009.

the attack to the client as part of its response. A web server that attacks web browsers is a common example. As the web browser requests content from a web server, the server returns a malicious web page that attacks the browser. These, so-called drive-by-download attacks, are capable of gaining complete control of the user's machine without the user's notice or consent. Merely visiting a web page with a vulnerable browser is sufficient for a successful attack to occur. Traditional defenses, such as firewalls, pose no barrier against these attacks and antivirus detection is currently poor [1].

Malicious web pages are prevalent in many areas of the Internet [2]. As such, any user might be affected and if this problem is not tackled more effectively it might result in a loss of trust in usage of the Internet and therefore negatively impact cyber commerce, banking and e-government efforts.

To increase the effectiveness of the malicious web pages, criminals are highly sensitive to location, language and economic trends [3]. Specific regions, for example, are targeted in, so-called, campaigns. As a result, attacks might differ from country to country [3]. For instance, the majority of malware on Chinese sites might target stealing passwords from online gamers [4,5] whereas malware on Brazilian sites is designed to steal bank account information [5]. With specific campaigns, attackers are capable to increase their return on investment.

Purpose of this study is to assess the threat of drive-by-download attacks within New Zealand. Not only is a New Zealand threat profile compared to other countries, but also a measurement study is conducted over an eight month period. The specific goals are:

- Assess where the threat is located. Not only are physical and network characteristics investigated, but malicious web pages are also inspected for content to assess concentration of malicious web pages on particular types of web pages.
- Comprehensively assess the threat within the New Zealand domain. The goal is to assess how many malicious web pages exist.
- Evaluate protection mechanisms. A variety of common protection mechanisms are evaluated in their effectiveness to protect end users from the threat of drive-by-download attacks.
- Investigate the changes in the threat landscape. The question of whether drive-by-download attacks are an increasing or decreasing phenomena are answered. Phenomena that are observed over the eight months period are investigated and assessed in detail.

The paper is structured as follows. In section 2, related work is presented. Section 3 describes the threats to internal and external validity that experimental designs may pose to measurement studies as presented in this paper. Sections 4,5, and 6 present the measurement study; section 7 concludes.

2 Related Work

Several measurement studies and whitepapers exist that describe the threat landscape. Most security vendors publish white papers on a regular basis. Because

campaigns are part of malware and malicious web sites, the security companies in this space pay special attention to geography. Data from the majority of reports list two countries to be hosting the majority of web-based client-side attacks: The United States of America and China [6,7,5]. McAfee didnt investigate physical location, but rather investigated top level domain names [8]. Their report shows that web sites in the .ro (Romania with 1.119%), .info, and .nu (Niue with 0.514%) contain the highest percentage of malicious web sites. .cn (China with 0.307%) is listed in fourth; .us (United States with 0.075%) in 18th position. New Zealand is not listed in the report. The shortcoming of these white papers is the fact that most are created by commercial entities that lack transparency in their data collection methodology and, as such, it is very difficult to assess external and internal validity. In this paper, we identify and mitigate threats to internal and external validity of the measurement study in a systematic and thorough manner.

China is named repeatedly in the white papers. Malicious web sites and the underground economy of the Chinese web was the focus of an academic study by Zhuge et al. [9]. Measurements on popular Chinese web pages revealed a high percentage of malicious web pages of 1.38%. These pages were sourced by submission of popular query terms to the Baidu search engine. The focus of the study presented in this paper investigates the .nz domain comprehensively: both popular and unpopular web pages; web pages that are indexed by search engines and web pages that are not indexed by search engines. Further, this study investigates the .nz over a period of 8 months, which allows us to investigate trends and other behavior that is influenced by time.

Provos et al. also conducted a measurement study on malicious web pages that were part of Google's index [10]. Over a period of 11 months, Provos et al inspected 66 million URLs. They observed an increasing trend of malicious URLs on their search results page. For each malicious URL, they investigated the physical location of the web page denoted by the URL as well as the server that hosts the exploit. China and the United states were the top two countries in both categories. New Zealand is not listed in their statistics. The work presented in this paper, focuses on the .nz domain. This study is comprehensive because our sample of pages represent a significant proportion of the .nz domain.

We argue that our survey was comprehensive because we inspected 247,198 web servers and we estimate there to be 500,000 web servers based in New Zealand. Note that our estimate is very rough as it is arrived at by assuming 30% of hosts are web servers and there are at least 1.7 million hosts (according to the 2008 CIA Factbook) in New Zealand. Furthermore, the 30% is based upon Internet Systems Consortiums January 2009 (www.isc.org) estimate of 625 million hosts on the Internet and NetCrafts January 2009 (www.netcraft.com) estimate of 185 million web servers on the Internet.

In addition, few studies exist that measure the prevalence of drive-by-download attacks. These studies use crawlers and search engine to generate a list of potentially malicious web sites. Measurements of 0.2% [11], 0.071% [12] and 0.1% [2] were observed. Direct comparison, however, cannot be conducted due to a lack

of information provided by the studies. As part of this work, the methodology of this study is disclosed so the results can be put into context in the future.

3 Threats to Internal and External Validity

The presented measurement study is designed to identify and collect information about malicious web pages with client honeypots. The purpose of this study is to answer the question on magnitude of the problem, where the threat is located, whether the threat is increasing/decreasing, assess protection mechanisms, etc. Further, the output of the measurement study could be used in economic models as illustrated by a variety of security related economic studies [13,14,9,15]. The business models [16,3] of the operation behind the malicious web pages can be used to devise strategies to break the business model. Measurement provides the inputs for accurate business models.

To utilize measurement studies for the purposes mentioned above, they have to have external and internal validity. The study needs conduct measurements correctly, so it measures what it is designed to measure (internal validity) and it needs to be generalizable beyond the experimental setting (external validity.) Threats to external and internal validity are plentiful, such as designs that include confounding variables or attempt inappropriate analysis. The major threats to the internal and external validity we identified are uncontrolled variables. These including our mitigation strategy are presented next. The threats are organized into three groups.

3.1 Apparatus (Client Honeypot)

Several threats were identified around the client honeypot technology used to conduct the measurement. A group of high priority threats were concerned with the functional aspects of correctness of the client honeypot. Questions on whether the client honeypot performs what it is designed to do were raised multiple times. The mitigation strategy identified was functional testing. Functional testing can be supported when the technology is transparent and available to a larger audience, so it can be examined and tested. The open-source community does provide this level of support and is one driver why our client honeypot has been made publically available as an open-source project. We believe that many threats are mitigated through this strategy.

In addition to the correctness of the client honeypot, reliability was a concern. Especially in a setting in which there are several network components involved and attack code is executed, threats that stem from low reliability emerge. Continuous monitoring and error handling appear to be mitigating these threats and have been implemented as part of our client honeypot.

Bias introduced by URL selection and client honeypot configuration were identified to be a threat to external validity. A sample taken from dubious sources, such as links in email SPAM messages, can show very different characteristics in malicious web pages than web pages sourced from search engines in a random

fashion [2]. While difficult to address the bias directly, full disclosure how URLs are sourced or providing the list of URLs used and how client honeypots are configured can prevent generalizing beyond the subjects studied.

3.2 Subjects (Web Page(s))

The subjects, the web pages, also pose a variety of threats to the experimental design of the measurement study. The primary threat is related to connectivity issues in which a web page may not be able to participate in the study because the network components, such as DNS server or/and HTTP server, are temporarily unreachable. This threat may be addressed through retrying retrieval of the web page multiple times and logging any unsuccessful visits of the web pages.

Further, a malicious web page may choose not to participate in the study. It could selectively not launch an attack as part of the study, but do so when accessed by a regular used. This may be caused by anti-forensic techniques employed by the malicious web page. The selective behavior could stem from the fact that the web page somehow identified the client honeypot. Primarily a malicious web page can identify the client honeypot through the means it makes requests; this threat is more closely reviewed in the following section.

3.3 Stimuli (Making the Request)

Stimuli, in our context the means of making the request to retrieve the web page for it to participate in the measurement study, is the final area that could pose threats to the validity of the measurement study. Since the process of making the request is part of the apparatus, similar threats around functional correctness and reliability apply to making the request. Functional testing and monitoring functionality during operation is the primarily mitigation strategy for those threats.

As mentioned in the previous section, a malicious web page may choose not to participate in the study through identification that is part of the study. A malicious web page may do so by analyzing the way requests are made. Several characteristics of the requests may cause this threat: Location, time, deceptive nature, and history.

Location. Location from where requests are made may pose a threat to a measurement studys validity. The two main reason why location is of importance are the campaigns run by attackers and evasion techniques. Campaigns were already described above. For a client honeypot, a campaign may manifest itself in an inability to detect certain malicious web pages that do target clients at a specific location. A client honeypot located in New Zealand accessing a malicious web that only triggers an attack if accessed from a client located in Germany will not be exposed to the attack and therefore fail identification of that malicious web page. An assessment on where a client is located is primarily made by mapping the IP address of the client to a specific location with freely available libraries,

such as MaxMind Geolocation Technology [17]. Web exploitation kits, such as MPack, provide functionality to enable location based triggering of attacks [18].

The geolocation-dependent triggering could easily be extended into a more fine grained triggering mechanism as an evasion technique to avoid specific networks. Since location and locale pose threats to the measurement study, they should be explicitly documented.

Time. Malicious web pages change over time. A malicious web page that exhibits malicious behavior in one month might cease to do so in the following month. The attackers might have abandoned the site or the web master might have detected and subsequently removed the malicious content. Sophos Security threat report 07/2008, for instance, describes that the number of malicious web sites they detect almost tripled [5] from 2007 to 2008. Just in a year, the number of malicious web pages on the Internet appeared to have changed significantly. So the time a measurement study is conducted greatly influences the results and, as a result, should be explicitly documented.

Deceptive Nature. If a malicious web page detects it is part of a study, it may choose not to participate in this study. It could make this decision based on identifying that the requests were made by a client honeypot. As such, the deceptive nature of the client honeypot in making the request needs to be closely aligned with how users make requests.

History. History of requests may pose another threat to a study of malicious web pages. Particular malicious web pages implement a tracking functionality in which the attack is launched only once upon a target. A client honeypot requesting the identical page a second time would not lead to an attack and therefore the malicious web page would be missed. This threat is mitigated through caching and rotation of IP addresses to minimize the chance of the client honeypot to visit a web page twice.

4 Comparative Measurement

The first measurement was designed to compare the risk of drive-by-download attacks from web servers in the .nz domain to other top level domains. Sample of potentially malicious web pages from the .nz, .uk,.au, and .com domain were inspected for malicious web pages with client honeypots. Statistical analysis of the malicious URLs and domains was conducted to assess risk levels at each top level domain. First, the methodology is presented followed by the results.

4.1 Methodology

From the Victoria University network, 664,000 web pages from the Australia, New Zealand, UK, and .com top level domain were inspected with a hybrid client honeypot system. The hybrid client honeypot system was composed of

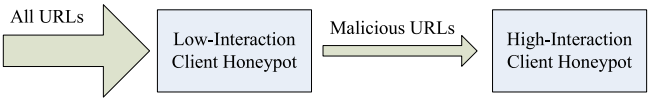


Fig. 1. Hybrid System

low- and high-interaction client honeypot Weta 1.0 and Capture-HPC 2.0 [19]. Each malicious URL was recorded and the number of malicious URLs per domain were analyzed to determine statistically significant differences across domains.

To compare URLs from the various domains, the malicious nature of URLs from the domains need to be assessed. The number of URLs needed to be of sufficient size to detect any statistically significant differences across the various domains. To achieve this, a large sample of 664,000 URLs needed to be classified. Due to resource constraints, a comprehensive classification of this many URLs using a slow high-interaction client honeypot was not possible. Instead, all URLs were inspected first with a fast-low interaction client honeypot system, which is capable of making an initial classification, and finally with a high-interaction client honeypot for a final classification as shown in Figure 1. The low-interaction client honeypot produces false positives and false negatives. The false positives were addressed by re-inspecting each URL that was classified as malicious with a high-interaction client honeypot. This high-interaction client honeypot filters out all false positives produced by the low-interaction client honeypot. In addition to producing false positives, the low-interaction client honeypot is known to miss attacks. The low-interaction client honeypot approximately misses half of the malicious web pages, but it is assumed that these false negatives apply with equal level across all domains. As such, a comparison is still possible. When we more comprehensively assess the .nz domain described in the later sections, we abandon the low-interaction client honeypot and exclusively inspect all URLs with a high-interaction client honeypot to minimize the level of false negatives.

The low-interaction client honeypot of the hybrid client honeypot system is Weta v1.0. Weta v1.0 is a simulated web browser that imitates Microsoft Internet Explorer 6.0 SP2. It only retrieves the web page denoted by the URL and not any embedded resources, such as iframes and images. It makes a classification based on the static attributes found on the page, such as existence of obfuscated JavaScript, Iframes, etc. Because of the common attributes used by malicious web pages, it is possible to identify malicious pages using this method. The system is very fast and is capable to classify a web page within 0.05 seconds on an Amazon EC2 instance with 1.7GB of RAM, which is equivalent to a CPU capacity of a 1.0-1.2 GHz 2007 Xeon processor, on a 250Mbps connection [20]. The classification mechanism used by this system produces 5.88% false positives and 53.85% true positives. (Note that even though the false positive rate is lower than the true positive rate, the majority of alerts generated by the low-interaction client honeypot will consist of false positives because the majority of web pages in the sample will be benign web pages [21].) The detection mechanism implemented by Weta v1.0 has been described in detail in previous work [22].

Each web page that is classified as malicious by the low-interaction client honeypot system is forwarded to the high-interaction client honeypot for re-inspection. The open-source high-interaction client honeypot used is Capture-HPC v2.0 developed at Victoria University of Wellington [19]. It uses a dedicated, vulnerable computer system to interact with potentially malicious servers. As responses from these servers are consumed by the client honeypot, it monitors the operating system at the kernel level for any unauthorized state changes at the process, file system and registry level (excluding authorized changes to the system state associated with background system processes, such as the creation of temporary cache files). For instance, if a new file appears in the startup folder after the vulnerable browser interacted with a server, the client-honeypot can conclude that the server it just interacted with must have launched a successful attack and placed the file in the startup folder. This approach filters out the false positives the low-interaction client honeypot produces.

Capture-HPC v2.0 was configured with stock installation of Windows XP SP2 and Internet Explorer 6 SP2 within a VMware Server 1.x virtual machine. No additional plug-ins were installed nor was the configuration of the system adjusted to solicit more attacks, such as lowering the browser default security settings. This system was configured with the en-nz locale and the location was set to be New Zealand. The client would be configured to visit the URLs sequentially. After each visitation, the client honeypot would wait 10 seconds to give the web page the opportunity to trigger the attack. If no unauthorized state changes were detected, the client honeypot would record its classification and proceed visiting the next URL. If unauthorized state changes were detected, the client honeypot would record the classification including all the unauthorized state changes that occurred. Before visiting the next URL, the state of the virtual machine would be reset into a clean state prior to doing so.

The data for the comparative study was collected in January and February 2008 from the network at the School of Mathematics, Statistics and Computer Science (now School of Engineering and Computer Science) at Victoria University of Wellington, in Wellington, New Zealand. The URLs from the domains .au, .com, .nz and .uk were sourced from the Yahoo Search engine [23]. Because the domains were domains of countries with the national language English, URLs could be sourced by submitting English queries to the search engine. By submitting the same queries to the search engine for each domain, it is expected that the URLs sourced from the results page are controlled and only differ in the domain they come from. Category bias, which was observed previously [2], such as elevated percentage of adult web pages over news web pages, should be applied consistently across all four domains. The queries were English 5 N-gram queries that randomly selected from the corpus of web pages linked by the DMOZ Open Directory Project [24]. The first 1000 URLs on the results page were used to build the list of 664,000 URLs.

Data Collection and Analysis. All URLs were first inspected with the low-interaction client honeypot Weta v1.0. Each URL that was classified as malicious could be a false positive. As a result, these URLs were forwarded to the

high-interaction client honeypot Capture-HPC v2.0 for a second and final inspection, which would eliminate false positives. The malicious classification of the final inspection by the high-interaction client honeypot was recorded for each URL. A visual verification of the unauthorized state changes ensured that no false positives were used in the data analysis phase.

To assess whether one top level domain contains generally more malicious URLs than others, a simple Chi-Square test was performed on the number of malicious URLs. Because some malicious URLs might have originated from the same host, a second Chi-Square test was performed on the number of unique host names that host at least one malicious URL.

4.2 Results

Inspecting the 664,000 URLs, the low-interaction client honeypot Weta v1.0 produced a total of 11,095 URLs that it considers may be malicious. Since the majority of these URLs were false positives, they were inspected once again with the high-interaction client honeypot Capture-HPC v2.0. A total of 38 malicious URLs from 27 unique hosts were detected.

Of the 168,000 URLs per domain, 26 unique malicious URLs from 16 unique hosts were identified for the .au domain, one URL from one host identified for the .com domain, eight URLs from 7 hosts were identified on the .uk domain, and three URLs from three unique hosts were identified for the .nz domain. The statistical Chi-Square test shows that the difference between the malicious URLs and hosts identified in the .au domain and any of the other domains is statistically very significant (URLs: $p < 0.0036$; Hosts: $p < 0.0092$).

5 Comprehensive Measurement

The first measurement used a hybrid system to compare domains. The second measurement should be more comprehensive. Because the hybrid system is likely to miss attacks, the second measurement is designed to minimize this risk by omitting inspection with the low-interaction client honeypot Weta v1.0. Instread, all URLs were inspected with the more accurate high-interaction client honeypot Capture-HPC v2.1 [19]. In addition, the URLs used for the comprehensive study were not sourced by a search engine, but rather by extracting URLs from the .nz domain file. This allowed us to even inspect URLs that were not indexed by search engines. Each malicious URL identified was recorded and several defensive techniques were evaluated against these malicious URLs.

5.1 Methodology

Each URL is inspected with the open-source high-interaction client honeypot used is Capture-HPC v2.1 developed at Victoria University of Wellington [19]. This is more accurate than the hybrid system used in the first experiment as it is likely to miss less attacks and is able to identify known and unknown attacks.

It produces negligible false positives. It is configured in the identical way as Capture-HPC v2.0 was in the first experiment. The version 2.0 and 2.1 differ primarily in bug fixes.

Several high-interaction client honeypots were used to inspect potentially malicious URLs. For analysis purposes and to counter evasion techniques based on tracking client honeypots, all requests were made via a HTTP/DNS proxy server Squid v2.6 and Pdnsd 1.2.6 [25,26]. These proxy servers were configured to cache more aggressively compared to a standard configuration to counter evasion techniques and support the subsequent analysis [27].

The data for the comprehensive study was collected in April 2008 from the network at the School of Mathematics, Statistics and Computer Science (now School of Engineering and Computer Science) at Victoria University of Wellington, in Wellington, New Zealand. The URLs were obtained by querying DNS records for valid .nz hostnames. Each hostname was checked for the existence of a common web server listening port (TCP port 80) indicating the existence of a web server. For all hostnames that did not indicate the existence of a web server, the hostname was modified with the prefix "www". and existence of a web server checked once again. For all hostnames for which an indication of a web server existed, a URL was generated that pointed to the main entry point of the web server, for example "http://www.vuw.ac.nz/". 247,198 unique URLs were obtained using this method.

Data Collection and Analysis. All URLs were inspected with the high-interaction client honeypot Capture-HPC v2.1, which has a negligible false positive rate. For each URL, the classification, time of inspection as well as any unauthorized state changes, such as a new file that appeared in the startup folder, were recorded. In addition, the network traffic as well as the web page itself was recorded by the HTTP and DNS proxy. A visual verification of the unauthorized state changes ensured that no false positives were used in the data analysis phase.

All malicious URLs were further analyzed. First, they were inspected once again with a fully patched system. This allowed us to assess whether dangerous zero-day exploits exist in the .nz domain. Second, a brief description of the web page and popularity ranking from Alexa, Google Toolbar and SiteAdvisor service [28,29,30] was obtained by visually inspecting the web page to assess whether popularity and particular topic areas show a higher concentration of malicious web pages. Third, each URL was cross-checked against the URL assessment service of Googles Safe Browsing API [31], McAfee SiteAdvisor [30], Stopbadware database [32], and with the HauteSecure browser plugin [33]. This allowed us to evaluate whether knowledge of these pages by leading blacklisting services exist.

5.2 Results

Comprehensive measurement of 247,198 URLs in the .nz domain detected a total of 52 malicious URLs. The sites themselves fill a wide spectrum of topics: From shopping sites, personal sites, to tourist sites. It seems as if sites from various

Fig. 2. Map of hosts

topic areas pose a risk to the end user. Adjusting browser behavior might reduce the risk (e.g. avoid SPAM links and adult content sites), but the risk cannot be eliminated.

Popularity of these sites in general is low; 17 of the URLs were not known to SiteAdvisor, Alexa or Google. Only 2 sites showed a medium popularity. In general, it is expected that in global comparison, sites in the .nz domain will rank low. The fact that several were not tracked by these services poses a risk, because if a site is not known, a security assessment is not likely to take place and therefore the site will not be tagged as malicious by the corresponding blacklisting services.

Several publicly available blacklisting services were used to check whether the malicious sites were known to the service. A combined assessment of Google, Stopbadware.com, and SiteAdvisor showed that only nine out of 52 sites were tagged as suspicious or malicious. Blacklisting based on these services would have protected the end user inadequately.

The browser plug-in by Haute Secure shows more promising results that allows for protection against malicious web pages. Haute Secure was able to detect 40 out of 52 sites, because this software takes a different approach. Instead of merely checking the main URL that is input into the browser, the Haute Secure plug-in also checks for embedded resources on that page. For instance, if a URL foo.com contains an iFrame that points to a centralizied exploit server maliciousSite.com, Haute Secure will detect and block the request to the maliciousSite.com effectively protecting the end user. The approach is more successful than merely blocking the main URL, because many malicious URLs point to few centralized exploit servers as shown in Figure 3.

The physical location of the exploit servers is shown in Figure 4. This Figure shows that the exploit servers are geographically more globally dispersed than the hosts of the .nz domain (as shown in Figure 2.) The hosts of the .nz domain are primarily located in New Zealand and the United States, where many discount hosting providers exist. The exploit servers referenced by these pages, however, are located in additional countries; primarily Unitied States, Russia and China.

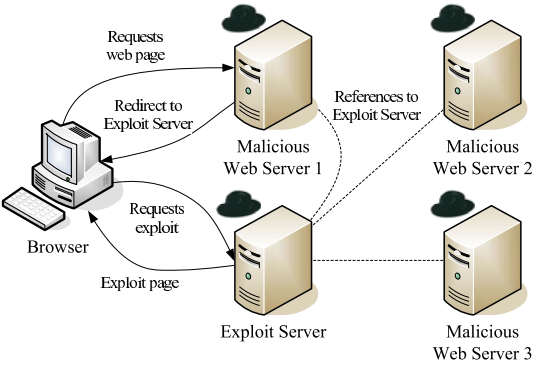


Fig. 3. Centralized Exploit Servers

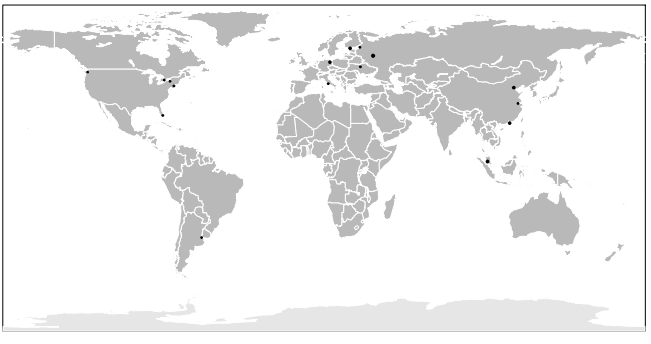


Fig. 4. Map of exploit servers

Besides blacklisting, patching was evaluated. The analysis of inspecting the malicious URLs with a fully patched system resulted in zero successful attacks. As such, patching is a very successful mechanism to defend against these attacks. However, in case just one malicious URL deploys a zero-day exploit, the impact on end users would be tremendous.

6 Term Measurement

In the last and final experiment, a trend assessment of the threat landscape in the .nz domain was made. Over a period of 6 months comprehensive and partial scans of the .nz were conducted repeatedly from the Victoria University network with the high-interaction client honeypot Capture-HPC v.2.1. Changes over this period were analyzed to detect trends and other observations.

6.1 Methodology

The client honeypots were identically setup and configured as described in section 5 High-interaction client honeypots were used exclusively because of their better detection accuracy.

The repeated comprehensive measurements were taken from June 2008 to November 2008 from the network at the School of Mathematics, Statistics and Computer Science (now School of Engineering and Computer Science) at Victoria University of Wellington, in Wellington, New Zealand. The URLs used for the comprehensive assessment were also used for the repeated measurements over the 6 month study period. Because attackers are likely to record the IP addresses of the clients and alter their behavior accordingly, the external IP address of the system was changed monthly.

Data Collection and Analysis. All URLs were inspected with the high-interaction client honeypot Capture-HPC v2.1 monthly. For each URL, the classification, time of inspection as well as any unauthorized state changes were recorded. In addition, the network traffic as well as the web page itself were recorded by the HTTP and DNS proxy. After the monthly scan, the HTTP and DNS proxy cache were cleared. A visual verification of the unauthorized state changes ensured that no false positives were used in the data analysis phase.

The monthly measurements were used to investigate trends in the attack landscape of the .nz domain. Further, monthly scans were analyzed to assess the extent of how dynamic the attack landscape is. The lifetime of malicious behavior of malicious web pages detected in June was determined. For the month of July to November those web pages were inspected for malicious behavior. This data allowed us to assess how quickly and to what extend malicious URLs are turning benign. To determine how quickly and to what extent previously benign URLs turn malicious, a percentage based upon newly discovered URLs was calculated.

6.2 Results

The 247,198 URLs were repeatedly inspected over a period of eight months. A total of 291 unique malicious URLs, about 0.12%, were identified. Results of the monthly inspection of these URLs are shown in Figure 5 (Note that no monthly scan was conducted in May 2008). Over the eight month period, no increasing nor decreasing trend can be detected. While there are fluctuations between 52 (April 2008) and 97 (July 2008) malicious URLs can be observed, significant increases and decreases appear to occur throughout the period of eight months. On average, 73.7 malicious URLs were detected.

Of the 291 malicious URLs identified, a majority of URLs only exhibited malicious behavior once as shown in Figure 6. No malicious URLs that exhibited malicious behavior constantly over the seven monthly scans. Only about 8% of malicious URLs exhibited malicious behavior more than half of the seven scans.

Of the malicious URLs identified over the eight month period, a considerable portion of the malicious URLs were newly classified as malicious compared to

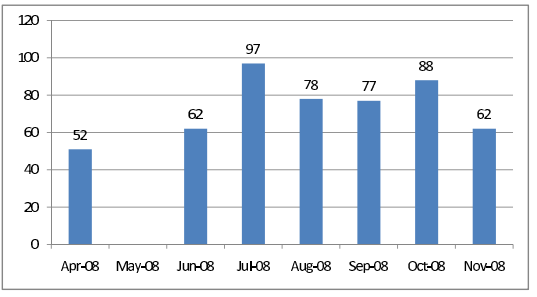


Fig. 5. Monthly Scan Results

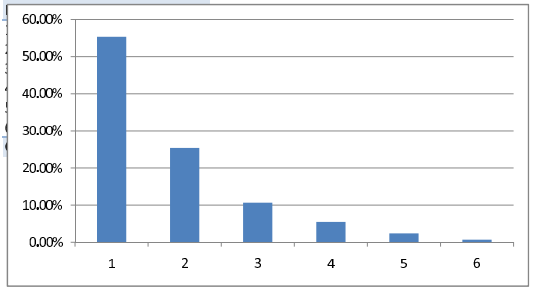


Fig. 6. Number of Times a Malicious URL exhibited over the Seven Monthly Scans

the previous month as shown in Figure 7. In July, for instance nearly 80% of the malicious URLs were newly classified as malicious compared to June. Over the following four months, the percentage of newly classified malicious URLs decreased continuously. In November, about 34% of the malicious URLs were newly classified as malicious compared to October. On average, 50% of malicious URLs were newly classified as malicious compared to the previous month.

In addition to assess URLs that turn malicious, malicious URLs that turn benign were observed. Figure 8 shows malicious URLs identified in June 2008 over time. In June 2008, 62 malicious URLs were observed. Of those 62 malicious URLs, about 32% were identified also in July 2008. The percentage continuously decreases over time. Of the 62 malicious URLs identified in June 2008, approximately 6% were also identified in November 2008. This decay appears to be decreasing over time with an initial half-life of 0.6 months and a half-life of 1.2 months 5 months later. An average half-life of 0.94 months is observed.

In summary, a constant threat level on the .nz domain was observed. However, while the number of malicious URLs identified does not indicate an upward or downward trend, a highly dynamic nature of malicious URLs has been observed. As malicious URLs disappear, they reappear elsewhere. The rates observed show that on average 50% of malicious URLs identified change monthly. This figure applies to the appearance of new and disappearance of existing malicious behavior. Appearance of malicious behavior can be explained by the continuous

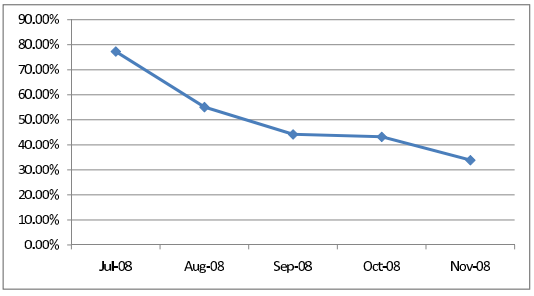


Fig. 7. Percentage of Newly Classified Malicious URLs From Previous Month

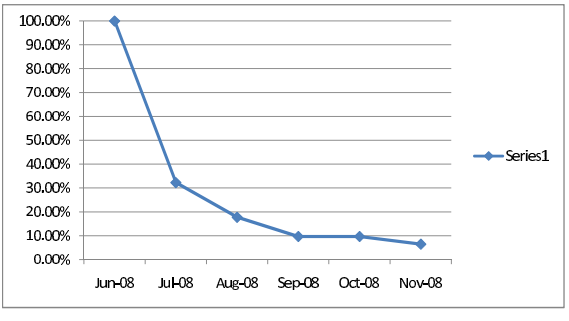


Fig. 8. Malicious URLs Identified in June 2008 Over Time (monthly)

malicious activity which target legitimate web sites. These web sites, which usually expose vulnerabilities, are attacked and modified to serve malicious code. Its estimated that 75-80% of all malicious web sites are hacked [5,34]. The web sites identified mostly seem to fall into this category. The do not appear to have been setup with the intent to attack visitors of that site, but are rather legitimate sites that are abused by a third party.

Malicious URLs that turn benign is a behavior that is a more difficult to explain. Potentially, this behavior could be attributed to the measures taken by the major search engines Google, Yahoo, and Live Search [35,36,37] which all now search for malicious web pages and either remove them (Yahoo) or tag them as malicious on the results page (Google and Live Search.) As malicious URLs are tagged, the webmasters is notified either directly or indirectly by the impact such a warning has on the traffic to her website, are now inclined to take action to remove the malicious content from their site, and secure their site so a repeated attack cannot occur [38]. Further, take-down notices by various entities in the security space could lead to shutting down of the responsible exploit servers. And finally, the malicious web pages might naturally exhibit sporadic malicious behavior. Even immediate interaction with a malicious page would result in benign behavior. This could be due to fast flux networks [39], malicious advertisements that are only shown occasionally [10], or intentional randomization by attackers to counter detection technology.

7 Conclusion

Over the last 10 months, the threat of drive-by-download attacks in the .nz domain was assessed. While the comparative measurement between the .au, .uk, .com and .nz domain has not revealed an elevated risk for the .nz domain, several hundred malicious URLs in the .nz were identified. These malicious URLs appear to be primarily sites that have been hacked by a malicious third party and then manipulated to serve exploit code. While the malicious URLs of the .nz were primarily hosted in New Zealand and the United States, exploit servers which host the actual attack code, were primarily located in the USA, Russia and China.

The malicious URLs identified in the month of April 2008 were cross-checked against known bad site of the Stopbadware, Google index and McAfee SiteAdvisor. Very few URLs were known by these three sources. This is an indication how difficult it is to identify malicious pages on the Internet and provide defensive intelligence to end-users. The Haute Secure browser plug-in was more successful in detection of the malicious content, because it instruments the browser and is able to view and check the commonly used centralized exploit servers. However, some URLs identified did not make use of an exploit server. Rather the exploit code was hosted directly on the page; such pages were often missed by Haute Secure browser plug-in. Identification of malicious web sites is overall difficult.

Patching was a successful method to defend against malicious web pages. None of the pages that were detected as part of this study successfully attacked a patched system. But even with patching being a good defensive strategy, it assumes that users do patch, which is not a given [40]. In addition, even patched systems can be at risk if a zero-day attack appears which continued to happen throughout 2008.

Analyzing the results from the monthly scans conducted from April 2008 to November 2008, the number of malicious URLs that were identified remained fairly constant. However, a very dynamic nature of malicious web pages were observed. Most of the malicious URLs identified only exhibited malicious behavior one month of the seven monthly scans performed. URLs that were identified to exhibit malicious behavior one month might cease to do so in the following month. At the same time, URLs that did not exhibit malicious behavior one month might do so in the following month. A monthly rate of change about 50% has been observed. Few malicious URLs that previously exhibited malicious behavior but are no longer exhibiting malicious behavior today could be attributed to the inaccessibility of the central exploit server that hosts the exploit code. Whether these exploit servers were purposefully abandoned by the attacker or taken down as part of a take-down notice could not be determined.

In conclusion, the attack landscape of the .nz and the likely the Internet as a hole is highly dynamic. The rate of change poses a challenge to blacklisting services as well as a risk to end users with the potential for rapid dissemination of zero-day attacks. Frequent scans of the web are required to obtain a good up-to-date view of the threat landscape.

8 Future Work

In this paper, a comprehensive measurement of the threat landscape in the .nz domain was presented. A highly dynamic nature of the threat landscape was identified. However, the reason for the dynamic natures could not be determined. Additional measurement on web servers, increasing the frequency of scans and development of tools to collect information on why web pages change in their malicious nature will be part of future work.

Acknowlegement

This work was funded by InternetNZ. URLs used for the .nz survey can be provided on request to allow replication of this work.

References

1. Seifert, C., Welch, I., Komisarczuk, P., Narvaez, J.: Drive-by-downloads (February 2008)
2. Seifert, C., Steenson, R., Holz, T., Bing, Y., Davis, M.A.: Know your enemy: Malicious web servers (2007)
3. Finjan: Web security trends report - Q2/2008 (2008)
4. Microsoft Corporation: Microsoft security intelligence report (2008)
5. Sophos: Sophos threat report (July 2008)
6. ScanSafe: Global threat report (2008)
7. Stopbadware.org: Badware websites report 2008 (2008)
8. McAfee, Inc.: Mapping the mal web, revisited (2008)
9. Zhuge, J., Holz, T., Song, C., Guo, J., Han, X., Zou, W.: Studying malicious websites and the underground economy on the chinese web. Technical report, University of Mannheim (2007)
10. Provos, N., Mavrommatis, P., Rajab, M.A., Monrose, F.: All your iframes point to us (2008)
11. Moshchuk, A., Bragin, T., Gribble, S.D., Levy, H.M.: A crawler-based study of spyware on the web. In: 13th Annual Network and Distributed System Security Symposium, San Diego, The Internet Society (2006)
12. Wang, Y.-M., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S., King, S.: Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In: 13th Annual Network and Distributed System Security Symposium, San Diego, Internet Society (2006)
13. Thomas, R., Martin, J.: The underground economy: Priceless (2006)
14. Kanich, C., Kreibich, C., Levchenko, K., Enright, B., Voelker, G., Paxson, V., Savage, S.: Spamalytics: An empirical analysis of spam marketing conversion. In: CCS, Alexandria, ACM, New York (2008)
15. Holz, T., Gorecki, C., Rieck, K., Freiling, F.: Measuring and detecting fast-flux service networks. In: 15th Annual Network & Distributed System Security Symposium, San Diego (2008)
16. Finjan: Web security trends report - Q1/2008 (2008)
17. MaxMind: MaxMind GeoLite Country (2002)

18. Seifert, C.: Know your enemy: Behind the scenes of malicious web servers (2007)
19. Seifert, C., Steenson, R.: Capture - honeypot client (2006)
20. Amazon, Inc.: Amazon Elastic Compute Cloud (Amazon EC2) (2006)
21. Axelsson, S.: The base-rate fallacy and its implications for the difficulty of intrusion detection. In: 6th ACM Conference on Computer and Communications Security, Singapore. ACM Press, New York (1999)
22. Seifert, C., Komisarczuk, P., Welch, I.: Identification of malicious web pages with static heuristics. In: Austalasian Telecommunication Networks and Applications Conference, Adelaide. IEEE, Los Alamitos (2008)
23. Filo, D., Wang, J.: Yahoo! search engine (1994)
24. Netscape Communications Corporation: DMOZ open directory project (1998)
25. Wessels, D., Nordstroem, H., Rousskov, A., Chadd, A., Collins, R., Serassio, G., Wilton, S., Francesco, C.: Squid web proxy cache (1996)
26. Moestl, T., Rombouts, P.: Pdnsd - proxy DNS server (2000)
27. Seifert, C., Endicott-Popovsky, B., Frincke, D., Komisarczuk, P., Muschevici, R., Welch, I.: Justifying the need for forensically ready protocols: A case study of identifying malicious web servers using client honeypots. In: 4th Annual IFIP WG 11.9 International Conference on Digital Forensics, Kyoto (2008)
28. Google Inc.: Google toolbar (2000)
29. Alexa Internet, Inc.: Alexa toolbar (1996)
30. McAfee, Inc.: Mcafee siteadvisor (2005)
31. Google Inc.: Google Safe Browsing API (2007)
32. Stopbadware.org: Home page (2006)
33. HauteSecure: Home page (2007)
34. Websense Inc.: State of internet security, Q1 - Q2, 2008 (2008)
35. Yahoo! Inc.: A safer way to search (2008)
36. Google Inc.: Putting a stop to spyware (2006)
37. Seifert, C.: Live search: Battling the plague of the web (2008)
38. Day, O., Palmen, B., Greenstadt, R.: Reinterpreting the disclosure debate for web infections. In: 7th Workshop on the Economics of Information Security, Hanover, New Hampshire (2008)
39. The Honeynet Project: Know your enemy: Fast-flux service networks (2007)
40. Frei, S., Duebendorfer, T., Ollman, G., May, M.: Understanding the web browser threat: Examination of vulnerable online web browser populations and the "insecurity iceberg" (2008)

A Combinatorial Approach for an Anonymity Metric

Dang Vinh Pham[1] and Dogan Kesdogan[1,2]

[1] Siegen University, Siegen, Germany
pham@fb5.uni-siegen.de
[2] NTNU-Norwegian University of Science and Technology, Trondheim, Norway
kesdogan@fb5.uni-siegen.de, dogan.kesdogan@q2s.ntnu.no

Abstract. A number of papers are suggested with the goal to measure the quality of anonymity of a given anonymity system. Most of them use the anonymity set as the basis for developing, reasoning about and applying measure. In this paper we argue that these approaches are premature. In this work we suggest to use the so called hypothesis set – a term derived from possibilistic information flow theory. Investigating the hypothesis set, it is possible to make the "protection structure" explicit and also define well known terms from measurement theory like scale and metric. We demonstrate our approach by evaluating the hypothesis set of the classical Chaumian Mix.

1 Introduction

One of the most important values in the information society is the information itself. Therefore, the protection of this precious good is a crucial task. A lot of research attention has been devoted to protecting the information contained *within* messages. This can be easily achieved today, by using encryption techniques. Here we focus not on the protection of such content data but rather on how to ensure the confidentiality of *traffic data*, i.e. information about communication relations. Protection of traffic data usually results in some form of anonymity. Such confidentiality is important, since third parties' unrestricted access to traffic data is considered an unacceptable invasion of both private and business lives. Several techniques are known to protect traffic data. However, there is still a lack of models to evaluate the level of protection these techniques can provide. There are two reasons to determine the level of protection. First, the quality of protection can be made visible to the user. Second, the mathematic model gives designers insight into the protection task.

In this paper we will study a specific system - the Mix system [1] that exposes the basis for an understanding of the abstract problem. Consequently, our focus is to investigate the abstract problem and the well known model of anonymity systems, the *anonymity set*, which can be used to model all traffic protection techniques [2]: Anonymity is the state of being not identifiable within a set of subjects, called the anonymity set.

All practical anonymity techniques leak some information about the users' communication peers in the anonymity set. As the number of observed anonymity sets increases, the uncertainty about the peer partners usually decreases, eventually reaching 0. At this point, there is enough information to determine the peer partners uniquely. According to this observation and inspired by the metric *Mean Time To Failure* (MTTF)

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 26–43, 2009.

[3], Shannon's *unicity distance* [4] and measurement theory [3], we define the metric *Mean Time To Deanonymization* (MTTD).

In [5] an attack algorithm called the *Hitting Set Attack* (HS-Attack) is proposed that can be used to measure MTTD for the MIX model. It was proven in [5] that HS-Attack requires the least number of observations that is necessary to uncover the peer partners of a user given a global passive attacker, who can observe any communication link from the sender to the Mix and from the Mix to the receivers. In this work we will investigate the HS-Attack for new structures to better understand how the attack evolves with increasing observations. We will mathematically approximate sharply this evolution from secure state into insecure state as a homomorphism of the "reality". Thereby we will provide mathematic answers to the following main questions:

- What is the average number of observations to disclose all of a user's peer partners?
- What is the average number of observations to disclose at least one of the user's peer partners?
- How likely is it to find a particular fraction of a user's peer partners in a random hypothesis after a given number of observations?

Question one was first considered with respect to inherent structures of the Mix- and attacker- model in [5]. They measured the mean number of observations to reach a necessary condition (called exclusivity) to disclose all of a user's peer partners. In contrast to [5], our approach is more comprehensive and granular, since it enables to directly model the number of observations to disclose any number of a user's peer partners.

In answering question two, we are the first to suggest the anonymity metric MTTD-1 that measures the time point, when the mix system's anonymity function leaks the first unambiguous information about a user's peer. This peer can be revealed by HS-Attack and we provide a mathematic measurement of the number of observations that the attacker needs to succeed. Each of a user's communication relationship is information theoretic anonymous, if it can be avoided that the attacker gains MTTD-1 observations. MTTD-1 extends the traditional measurement of anonymity that until now only considers the time point when all of a user's peer is disclosed.

Finally question three applies to the situation, when it is not possible to definitively identify any user's peer. Our mathematic model gives the probability that a random hypothesis contains a certain number of the user's contacts. The latter point shows that our approach opens the door to further analysis beyond the scope of unambiguous identification of peers and we are also the first to address this issue.

This paper is organised as follows. Section 2 will provide basic background information, related works and the used Mix meta model. Section 3 will describe the inherent structures and properties of hypotheses sets and we will contribute a precise mathematic model to describe them. Based on this model, we will derive analytical formulas to expose distinct anonymity states with respect to the Mix parameters and the observations of the attacker in Sect. 4. We will measure the mean number of observations to identify arbitrary subsets of a user's peer partners, the size of the hypothesis set and the probability to find a particular fraction of a user's partners in a randomly computed hypothesis. Section 5 will finally summarise our results and outline future works.

2 Background

Over the last years a handful of "anonymity metrics" have been proposed [6, 7, 8, 9, 10, 11, 12] that measure the information flow to the attacker. If information flows to the attacker, then it reduces the uncertainty of the attacker, which can be measured with some variant of Shannon's entropy. All entropy measurements follow here a similar scheme: information flow occurs if the attacker is really uncertain[1] and afterwards fairly certain.

This kind of "macroscopic" measurement can easily be applied to different anonymity systems, since only the probability distributions have to be known. The paradigm "information flows when uncertainty decreases" is problematic as discussed in [13, 11]. Our approach is more microscopic. The attacker and his knowledge (his interpretation) is incorporated in the model. Thus, direct application of our approach to other anonymity techniques (e.g. Crowds [14]) is not given, i.e. it has to be adapted or rather redeveloped.

Again, we model the whole path from secure to insecure state where the uncertainty with each observation decreases (i.e. number of hypotheses decreases). Indeed, in each step we can measure the uncertainty by using entropy as suggested in the literature. This would be a concomitant measurement. However, we think that the other way around is not possible (without formal proof), since suggested entropy based approaches measure only the actual information flow but not how the security evolves with time. The reason for our approach is that we think that the goal of a security strength metric is *to quantify the attackers efforts* that are required to break a system's security. Therefore, with respect to the general paradigm "the harder the successful attack the stronger the system" the actual entropy metric suggestions are premature[2].

Consequently, we define anonymity in terms of the attacker's knowledge, using the well known trace-based approach [15]. An attacker observes and accumulates input and output events of the anonymity system. The measurement of anonymity depends on the knowledge (i.e. interpretation of the observations) of the attacker that we model with a set of hypotheses. The only assumption we make is that the user keeps the set of peer partners. Unlike the theoretic works in this area (see e.g. [16, 17, 18]), we are not interested to give a formal specification of the anonymity problem.

2.1 The Mix Model

We consider the Mix Model that was described in [5]. The *Mix* technique was proposed by David Chaum in 1981 [1]. Figure 1 shows the basic ingredients of this technique which consist of a set of senders S, a set of recipients R, and a Mix node. Note that S and R can be equal. All senders in S are connected to the Mix and the Mix itself is connected to all recipients in R by a communication network with reliable secure channels. A reliable secure channel does not

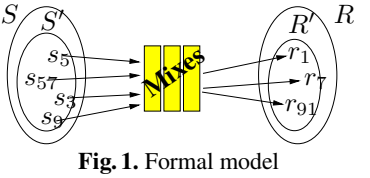


Fig. 1. Formal model

[1] E.g. a priori distribution is the uniform distribution.

[2] The draw back of our approach (as mentioned above) is the necessity of modelling explicitly the anonymity system and the attackers knowledge.

result in loss or duplication of transmitted messages, and guarantees authenticity, integrity, and confidentially of transmitted messages. The users and the Mix transmit messages by using the following protocols:

User Protocol: Users prepare their messages to be of constant length either by splitting long messages or by padding short messages to the specified length. Each message is encrypted twice with one time pads: first the message is encrypted using a shared secret between the sender and the intended recipient, and then it is encrypted using a shared secret between the sender and the Mix. The users send twice encrypted messages to the Mix.

Mix Protocol: A Mix collects b messages (called a *batch*) from distinct users, decrypts the messages, and outputs the decrypted messages in a batch in a different order than the order in that they were received (lexicographically sorted or randomly delayed). The output is broadcasted to all recipients. Furthermore, any incoming packet is compared with formerly received messages (i.e. by locally caching formerly received messages) in order to reject any duplicate messages.

The basic Mix technique described above can perfectly hide the communication relationships between senders and recipients of messages from everybody but the Mix and message senders. Even the act of sending or receiving can be perfectly hidden if the above protocol is applied in fixed time slots, and if every user supplies a fixed number of messages (perhaps some or all of them being dummy messages) to each slot and the whole output batch in a time slot is distributed to every user [19, 1, 20]. Pfitzmann [20] states that the Mix technique provides information-theoretic anonymity and unobservability based on complexity-theoretic secure cryptography.

The pure Mix technique. The "perfect" anonymity solution discussed above uses dummy messages and broadcasting. This solution is not followed widely in large networks such as the Internet, as justified in [5]. As a consequence, most current implementations and solutions use a variant of the perfect Mix solution by neither using dummy messages nor the broadcasting function. We refer to this kind of Mix techniques by the term *pure* Mix technique. Our pure Mix (also called threshold Mix) model is quite general and also Pool-Mixes can be mapped on it as shown in [10].

We consider a *global passive attacker* who is capable of observing all communication links simultaneously as described in [5]. This attacker model is also known as the Dolev-Yao model in the literature. Based on this attacker, we will use the following formal model of a pure Mix and information leakage therein for our analysis.

Formal Model of the Pure Mix Technique

– A communication system consists of a *set of senders* S, and a *set of recipients* R, and a Mix node (see Fig. 1). If a sender $s \in S$ communicates with a recipient $r \in R$, then we say that s and r are peer partners. If the roles of sender and receiver need to be distinguished, then we say that s is a peer sending partner of r and r is a peer recipient partner of s.

- In each *communication round*[3] a subset $S' \subseteq S$ of all senders S send a message to their peer partners. Let $R' \subseteq R$ be the set of intended recipients. The act of sending or receiving a message is not hidden among dummy messages.
- The size of the *sender anonymity set* is $|S'| = b$, where $1 < b \leqslant |S| = n$.
- The size of the *recipient anonymity set* is $|R'| \leqslant b$ since each sender sends exactly one message and several senders may communicate with the same recipient. The size of the recipient set is $|R| = N$.
- The information leakage X available to an attacker in a communication round consists of the pair (S', R') of peer senders and receivers.

2.2 The Hitting-Set Attack

The hitting-set attack (HS-Attack) introduced by [21, 5] is a global passive attack. The goal of the attack is to compute all possible peer recipient sets of a target sender $Alice \in S$ that are called *hypotheses*. Alice's peer recipient set is $\mathcal{H}_A$ and its size is $m = |\mathcal{H}_A|$. We will denote recipients $r \notin \mathcal{H}_A$ by the term *non-peers*. If HS-Attack can find only one hypothesis of size m, then Alice's peer set is uniquely identified. The adversary is interested in Alice's peers, he therefore only observes those pairs (S', R'), where Alice participates as a sender, i.e. $Alice \in S'$. Under this condition we denote the corresponding recipient set R' by the term *observation* $\mathcal{O}$ and the set of observations collected during t rounds is the *observation set* $\mathcal{OS} = \{\mathcal{O}_1, \ldots, \mathcal{O}_t\}$. For each hypothesis $\mathcal{H} \neq \mathcal{H}_A$, it is unlikely that whenever Alice sends a message, also a receiver of $\mathcal{H}$ is contacted. The number of hypotheses therefore decreases with increasing number of observations as illustrated in Example 1.

Example 1. Let $\mathcal{H}_A = \{8, 9\}$ and the observations at time 1,2,3 be $\mathcal{O}_1 = \{8, 5\}$, $\mathcal{O}_2 = \{9, 4\}$, $\mathcal{O}_3 = \{8, 6\}$. Alice contacted peer 8 in the first and third observation and peer 9 in the second observation. At time point 2 the attacker only sees $\mathcal{O}_1, \mathcal{O}_2$, therefore $\mathcal{H} = \{5, 4\}$ is a hypotheses, since these peers could also be contacted by Alice. But at time point 3 $\mathcal{H}$ is excluded, since neither 4 nor 5 is contacted in $\mathcal{O}_3$.

To mount the HS-Attack, the attacker starts with the set $\mathcal{L}_0$ that contains all $\binom{N}{m}$ possible subsets of cardinality m of N recipients, which is called the *hypothesis set*. We assume in this paper that m is known[4], because we are interested in analysing how long Alice can keep a constant set $\mathcal{H}_A$ of m peer partners anonymous. Since Alice has m peer partners, exactly one subset in $\mathcal{L}_0$ is the set of all peer partners of Alice. Let $\{\mathcal{O}_1, \mathcal{O}_2, \mathcal{O}_3, \ldots, \}$ be the observations in the successive communication rounds in which Alice participates. Since Alice has a peer partner in $\mathcal{O}_1$, a set in $\mathcal{L}_0$ that has an empty intersection with $\mathcal{O}_1$ cannot be the set of all peer partners of Alice. Thus upon observing $\mathcal{O}_1$, the attacker obtains a new hypothesis set $\mathcal{L}_1$ by discarding all recipients sets in $\mathcal{L}_0$ that have an empty intersection with $\mathcal{O}_1$. The attacker repeats this process to

[3] A communication round consists of the following events: The Mix node collects messages from a fixed number of distinct senders, and after applying the "Mix" protocol, it forwards the collected messages to their intended recipients.

[4] All attacks shown in this paper are also applicable if m is unknown. See [22] for a justification.

generate hypotheses sets $\mathcal{L}_2, \mathcal{L}_3, \ldots$ after observing recipient sets $\mathcal{O}_2, \mathcal{O}_3, \ldots$ respectively, until the hypothesis set $\mathcal{L}_t$ has only one subset in it. The last remaining subset in the hypothesis set $\mathcal{L}_t$ has to be the set $\mathcal{H}_A$ of all peer partners of Alice, hence the algorithm *fully discloses* Alice's peer set. Note that HS-Attack finds the *unique minimal* hitting set of all observations. A *hitting set* is a set that intersects with all given sets [5]. The hitting set is called *minimal*, if no proper subset of it is a hitting set, otherwise it is called *non-minimal*. All hitting sets addressed in this paper are of size less or equal m. Also note that HS-Attack requires the least number of observations to disclose Alice's peer set under a global passive attacker as proved in [5].

3 Properties of Minimal Hitting Sets

Peers of any hitting sets $\mathcal{H} \neq \mathcal{H}_A$, where $\mathcal{H} \leq m$ are unlikely to be always contacted whenever Alice communicates and this becomes unlikelier, the smaller the size of $\mathcal{H}$ is. After some observations, all hitting sets of size m must therefore be minimal. From now on these minimal hitting sets (of cardinality m) are called *hypotheses* and the *hypothesis set* is the set of all hypotheses. Each non-minimal hitting set $\mathcal{H}$ is a superset of a minimal hitting set $\mathcal{H}'$ of cardinality $m' < m$. Minimal hitting sets therefore represent the common peers of all non-minimal supersets thereof.

A peer can be identified, if it is common to all hitting sets. It is therefore straight forward and without loss of generality to restrict our analysis to minimal hitting sets.

The HS-Attack [21, 5] in Sect. 2.2 does not focus on minimal hitting sets. It simply removes non hitting sets from the set of all $\binom{N}{m}$ possible sets of size m. In contrast to this the ExactHS attack introduced by Pham [23] is the first work that reveals precise structures and quantities of minimal hitting sets. The ExactHS attack is a structured variant of the minimal hitting set attack that requires the same (number of) observations to disclose Alice's peer set as the HS-Attack.

We will extend Pham's work [23] and show how to derive a mathematic model for the evolution of the minimal hitting sets by observations. The obtained model will be elementary, since it enables us to determine the probability, the number and the structure of the minimal hitting sets after any number of observations with respect to the parameters N,b,m of the Mix. In particular we can determine the number of observations, such that a particular number of Alice's peers is disclosed, which is our new metric MTTD.

3.1 Number of Minimal Hitting Sets

The ExactHS attack [23] is a minimal hitting set attack that computes all minimal hitting sets of size lower or equal m with respect to a given observation set (representing the observations of the attacker). It therefore allows us to prove the following claim about the number of minimal hitting sets.

Claim. Let N, b, m be the Mix parameters and $\mathcal{H}_A$ be Alice's peer set of size m. For a given observation set $\mathcal{O}S$, the maximal number of possible minimal hitting sets of cardinality less or equal to m in $\mathcal{O}S$ is b^m. This bound is *sharp*, if $mb \leq N$. For $mb > N$ this is still an upper bound, but it is not sharp.

[5] In our case these sets are the observations $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_t$.

ExactHS Algorithm. Before the first invocation of Alg. 1 the set of all minimal hitting sets $\mathcal{L}$ and the candidate set $\mathcal{H}$ are empty, and the observation set $\mathcal{OS}$ consists of all observations collected by the attacker. The computation of the minimal hitting sets is initially invoked by calling $ExactHS(\mathcal{OS}, m, \mathcal{H})$. We refer to this observation set by the term *initial observation set*, as $\mathcal{OS}$ will be changed during the processing of ExactHS. In each recursion level $\mathcal{H}$ is extended by exactly one peer r, chosen in Line 7 from a *designated observation* $\mathcal{O} \in \mathcal{OS}$ determined in Line 5, where $\mathcal{OS}$ is the actual observation set. At the invocation of the next recursion level to determine the next peer to be added to $\mathcal{H} \cup \{r\}$ in Line 8, ExactHS is applied to a modified copy of the actual observation set that contains no observations intersecting with $\{r\}$. This step of removing is important to avoid non-minimal hitting sets, as it allows us to focus on adding only those peers to the actual set $\mathcal{H} \cup \{r\}$ that definitively intersect with observations not intersected by $\mathcal{H} \cup \{r\}$. Finally, if Line 2 detects that no observation of the actual observation set remains that is not intersected by $\mathcal{H}$, then $\mathcal{H}$ is a hitting set. In this case it will be added to the set $\mathcal{L}$ in Line 3. After a selection of r has been done in a recursion level, we remove r from all observations of the actual observation set in Line 8 and from the designated observation $\mathcal{O}$ in Line 9. This way the algorithm can repeat the extension of $\mathcal{H}$ with a new peer r not chosen before in Line 7.

Algorithm 1. ExactHS

```
1: procedure ExactHS(OS, m', H)
2:     if OS = {} then
3:         L ← L ∪ {H}                          ▷ H is a hitting set, add it to hypothesis set L
4:     else if m' ≥ 1 then                      ▷ add a peer to H, if H contains less than m peers
5:         choose O ∈ OS
6:         while (|O| > 0) ∧ ({} ∉ OS) do
7:             choose r ∈ O                     ▷ r will become element of H
8:             ExactHS(OS \ {O_i ∈ OS | r ∈ O_i}, m' − 1, H ∪ {r}) ▷ select remaining (m' − 1) peers of H
9:             OS ← ⋃_{O_i ∈ OS} {O_i \ {r}}    ▷ remove r in all observations of OS
10:            O ← O \ {r}                      ▷ do not choose r in this recursion level again
11:        end while
12:    end if
13: end procedure
```

Bound of the Number of Minimal Hitting Sets. ExactHS creates a hitting set $\mathcal{H}$ by starting with an empty set $\mathcal{H} = \{\}$ and adding a recipient to $\mathcal{H}$ in each choice phase represented by the lines 6–11. The number of recursive invocation of the choice phases in Line 8 is restricted by m, since we are interested in computing hitting sets $\mathcal{H}$ with at most m recipients. In each choice phase we only have at most b possible choices of a recipient r_i, because only recipients $r_1, \ldots, r_b$ of a fixed observation $\mathcal{O}$ are selected. From the restriction on the number of recursive invocations of the choice phase and the number of choices in each phase, we can conclude that the algorithm computes at most b^m minimal hitting sets. A formal proof that ExactHS is sound and complete with respect to the computation of all minimal hitting sets can be found in [23].

To show that the bound b^m of the algorithm is tight, we construct m pairwise disjoint observations $\mathcal{O}_1, \ldots, \mathcal{O}_m$, such that $\mathcal{O}_i \cap \mathcal{O}_j = \emptyset$ and $|\mathcal{O}_i| = |\mathcal{O}_j| = b$ for distinct $i, j \in \{1, \ldots, m\}$.

Let us consider a concrete example with the parameters $m = 2, b = 2$, the victim peer set $\mathcal{H}_A = \{1, 2\}$ and the observations $\{1, 3\}, \{2, 4\}$. A short glance shows that there are $b^m = 4$ minimal hitting sets, namely: $\{1, 2\}, \{1, 4\}, \{3, 2\}, \{3, 4\}$.

3.2 Structuring Minimal Hitting Sets

This section shows the classification and quantification of minimal hitting sets introduced by [23].

We partition the minimal hitting sets into $(m + 1)$ disjoint classes $\mathfrak{H}_0, \ldots \mathfrak{H}_m$. A minimal hitting set $\mathcal{H}$ belongs to the class $\mathfrak{H}_i$ (written $\mathcal{H} \in \mathfrak{H}_i$), if and only if it contains exactly $(m - i)$ distinct peer partners of Alice.

For sets A, B and integer i we define $A^{\overline{i}} = \bigcup_{j=0}^{i} A^j$, where A^0 is a neutral element, such that $A^0 \times B^k = B^k$. For example the class $\mathfrak{H}_2$ might contain the (minimal hitting) sets $\mathcal{H}_2 = \{r_{2_1}, r_{2_2}, a_{2_3} \ldots, a_{2_m}\}$ and $\mathcal{H}'_2 = \{r'_{2_1}, a'_{2_3} \ldots, a'_{2_m}\}$, where each r_{i_j} represents a non-peer and each a_{i_k} represents an Alice's peer. All peers within a set must be disjoint.

$$
\begin{aligned}
\mathfrak{H}_0 &= \{\mathcal{H}_A\} \\
\mathfrak{H}_1 &\subseteq (R \setminus \mathcal{H}_A)^{\overline{1}} \times \mathcal{H}_A^{m-1} \\
\mathfrak{H}_2 &\subseteq (R \setminus \mathcal{H}_A)^{\overline{2}} \times \mathcal{H}_A^{m-2} \\
&\vdots \\
\mathfrak{H}_m &\subseteq (R \setminus \mathcal{H}_A)^{\overline{m}} \ . \qquad (1)
\end{aligned}
$$

Bounds of Minimal Hitting Set Classes. The last section derives the bound of b^m for the number of minimal hitting sets. Based on ExactHS (2) represents refined bounds for each of the minimal hitting set classes $\mathfrak{H}_0, \ldots, \mathfrak{H}_m$ as proved in [23].

$$|\mathfrak{H}_i| = \binom{m}{m-i}(b-1)^i = \binom{m}{i}(b-1)^i \ . \qquad (2)$$

Note that this bound is again tight and we can use the same construction of m disjoint observations as in Sect. 3.1 to prove its tightness. It is also clear that the sum of the cardinality of each class results in b^m, i.e. $\sum_{i=0}^{m} |\mathfrak{H}_i| = \sum_{i=0}^{m} \binom{m}{i}(b-1)^i = b^m$.

Probability Property of Classes. To model the probability of excluding a particular hypothesis of a class $\mathfrak{H}_i$, we assume that Alice chooses her recipient in each round uniformly distributed from the set of m recipients $\mathcal{H}_A = \{a_1, \ldots, a_m\}$. Similarly the remaining $b - 1$ senders of a batch are assumed to select their receivers uniformly from the set R of N receivers. A (former) hitting set $\mathcal{H}$ is *excluded* by an observation $\mathcal{O}$, if and only if $\mathcal{H}$ does not intersect with $\mathcal{O}$, i.e. if $\mathcal{H} \cap \mathcal{O} = \emptyset$. A hitting set $\mathcal{H}$ is *excludable* with respect to an observation set $\mathcal{OS}$, if and only if $\mathcal{H}$ is a hitting set in $\mathcal{OS}$ and there exist an observation $\mathcal{O} \notin \mathcal{OS}$, such that $\mathcal{H}$ would be excluded by $\mathcal{O}$.

Suppose that a hypothesis $\mathcal{H} \in \mathfrak{H}_i$ is given. According to Pham [24] the probability that this particular hypothesis is excluded by the next observation $\mathcal{O}$ is:

$$p_{inv}(N, b, m, i) = \frac{i}{m}(1 - \frac{m}{N})^{b-1} \ . \qquad (3)$$

Thereby the first factor $\frac{i}{m}$ is the probability that Alice chooses to communicate with any of the i recipients not covered by $\mathcal{H}$ in the observation $\mathcal{O}$. The second factor represents the probability that the remaining $(b - 1)$ senders do not choose to contact any of the recipients in $\mathcal{H}$.

3.3 Extensive Hypotheses

Extensive hypotheses combine the knowledge about the minimal hitting sets of size $\leq m$, which are computed by ExactHS with the knowledge about the structure of hypotheses. That way we can predict which hypotheses will be computed in the future.

Table 1 shows the the Minimal hitting sets $\mathcal{M}_i$, the extensive hypothesis set $\mathcal{L}_i$, and the excluded sets that result from analysing the observation set $\mathcal{OS}_i = \{\mathcal{O}_1, \ldots, \mathcal{O}_i\}$.

For $i = 0$ there is no observation and no $\mathcal{M}_0$, but we have knowledge about the initial hypothesis set represented by the classes (1) in $\mathcal{L}_0$. Each element $\mathcal{H} \in \mathcal{L}_i$ is called an *extensive class*. We will address these classes by their order from left to right and from top to bottom, i.e. $\mathcal{H}_u$ is the u-th class in the hypothesis set. An extensive class is *unspecified*, if it contains a variable x (standing for any variable x_u^v with indexes), otherwise it is *specified*. A specified extensive class is called a *specified extensive hypothesis*. The only specified extensive hypothesis in $\mathcal{L}_0$ is $\mathcal{H}_1 = \{1, 2, 3\}$. Each variable x represents any $(b-1)$ *unspecified* non-peers $r \in R \setminus \mathcal{H}_A$, thereby only distinct peers can be assigned to $x_u^v, x_u^w \in \mathcal{H}_u$ for $v \neq w$. Peers that are explicitly mentioned are called *specified*. Newly specified peers are bold highlighted. Note that writing $\mathcal{H} \subseteq \mathcal{L}_0$ would be more appropriate than $\mathcal{H} \in \mathcal{L}_0$, since $\mathcal{H}$ is a class. But for the sake of reducing formalisms and simplifying explanations, we use the element-notation and -operations.

For $i = 3$ we can see that extensive hypotheses also visualise exclusions of implicit hypotheses (e.g. $\{2, 3, 4\}$ and $\{3, 4, 5\}$). A hypothesis is *implicit*, if it has not been computed by ExactHS as a minimal hitting set yet, otherwise it is *explicit*. Finally all extensive hypotheses will be explicit and equal to minimal hitting sets as seen in $i = 4$.

Table 1. Evolution of minimal hitting sets and extensive hypothesis set

i	$\mathcal{O}_i$	Minimal hitting sets $\mathcal{M}_i$	Extensive hypothesis set $\mathcal{L}_i$	Exclusion
0			$\mathfrak{H}_0 : \{1,2,3\}$; $\mathfrak{H}_1 : \{1,2,x_2^1\}, \{1,3,x_3^1\}, \{2,3,x_4^1\}$; $\mathfrak{H}_2 : \{1,x_5^1,x_5^2\}, \{2,x_6^1,x_6^2\}, \{3,x_7^1,x_7^2\}$; $\mathfrak{H}_3 : \{x_8^1,x_8^2,x_8^3\}$	
1	$\{1,4\}$	$\{1\}, \{4\}$	$\mathfrak{H}_0 : \{1,2,3\}$; $\mathfrak{H}_1 : \{1,2,x_2^1\}, \{1,3,x_3^1\}, \{2,3,\mathbf{4}\}$; $\mathfrak{H}_2 : \{1,x_5^1,x_5^2\}, \{2,\mathbf{4},x_6^1\}, \{3,\mathbf{4},x_7^1\}$; $\mathfrak{H}_3 : \{\mathbf{4},x_8^1,x_8^2\}$	
2	$\{2,5\}$	$\{1,2\}, \{1,5\}, \{4,2\}, \{4,5\}$	$\mathfrak{H}_0 : \{1,2,3\}$; $\mathfrak{H}_1 : \{1,2,x_2^1\}, \{1,3,\mathbf{5}\}, \{2,3,4\}$; $\mathfrak{H}_2 : \{1,\mathbf{5},x_5^1\}, \{2,4,x_6^1\}, \{3,4,\mathbf{5}\}$; $\mathfrak{H}_3 : \{4,\mathbf{5},x_8^1\}$	
3	$\{1,6\}$	$\{1,2\}, \{1,5\}, \{4,2,6\}, \{4,5,6\}$	$\mathfrak{H}_0 : \{1,2,3\}$; $\mathfrak{H}_1 : \{1,2,x_2^1\}, \{1,3,5\}$; $\mathfrak{H}_2 : \{1,5,x_4^1\}, \{2,4,\mathbf{6}\}$; $\mathfrak{H}_3 : \{4,5,\mathbf{6}\}$	$\{2,3,4\}, \{3,4,5\}$
4	$\{3,4\}$	$\{1,2,3\}, \{1,2,4\}, \{1,5,3\}, \{1,5,4\}, \{4,2,6\}, \{4,5,6\}$	$\mathfrak{H}_0 : \{1,2,3\}$; $\mathfrak{H}_1 : \{1,2,\mathbf{4}\}, \{1,3,5\}$; $\mathfrak{H}_2 : \{1,5,\mathbf{4}\}, \{2,4,6\}$; $\mathfrak{H}_3 : \{4,5,6\}$	

$\mathcal{L}_i$ is constructed with respect to the observation set $\mathcal{OS}_i = \{\mathcal{O}_1, \ldots, \mathcal{O}_i\}$ and the minimal hitting sets $\mathcal{M}_i$ for $i \geq 1$. Let $\mathcal{H} \in \mathfrak{H}_j$ be an extensive class of size m and $\mathcal{H}^- = \mathcal{H} \setminus \mathcal{H}_A \setminus \{x\}$, then $\mathcal{H} \in \mathcal{L}_i$, if and only if $\mathcal{H}^-$ is a minimal hitting set with respect to $\mathcal{OS}'_i = \{\mathcal{O} \setminus \mathcal{H}_A \mid \mathcal{O} \in \mathcal{OS}_i, \mathcal{O} \cap \mathcal{H} \cap \mathcal{H}_A = \emptyset\}$[6]. Thus for each $\mathcal{H}^-$, there is a minimal hitting set $\mathcal{H}_i$ with respect to $\mathcal{OS}_i$, such that $|\mathcal{H}_i| \leq m$, $\mathcal{H}^- = \mathcal{H}_i \setminus \mathcal{H}_A$ and $\mathcal{H}_i \subseteq \mathcal{H}$ An extensive class $\mathcal{H}$ that complies to these conditions is called *minimal*, hence an extensive hypothesis set consists of only minimal classes. This defines a surjective

[6] The set $\mathcal{OS}'_i$ results from removing all Alice's peers from those observations in $\mathcal{OS}_i$ that do not contain any of Alice's peers of $\mathcal{H}$.

mapping of the extensive hypothesis set $\mathcal{L}_i$ to the minimal hitting sets with respect to $\mathcal{OS}_i$. We can define $\mathcal{L}_i$ for $i \geq 1$ recursively as follows:

1. Let $\mathcal{L}_i = \{\}$ before the start of its construction below.
2. For each extensive class $\mathcal{H} \in \mathcal{L}_{i-1}$, let $\{r_1, \ldots, r_k\} \subseteq \mathcal{H}$ be the set of all specified peers in $\mathcal{H} \in \mathfrak{H}_j$, where $k \leq m$. Apply either 3. or 4. to each $\mathcal{H}$.
3. If $\{r_1, \ldots, r_k\} \cap \mathcal{O}_i \neq \emptyset$ then add $\mathcal{H}$ to $\mathcal{L}_i$, i.e. $\mathcal{L}_i = \mathcal{L}_i \cup \{\mathcal{H}\}$, because $\mathcal{H}$ is not excluded by $\mathcal{O}_i$.
4. Else if $\{r_1, \ldots, r_k\} \cap \mathcal{O}_i = \emptyset$ and $k < m$ then add for each non-peer $r \in \mathcal{O}_i \setminus \mathcal{H}_A$ the extensive class $\mathcal{H}' = \{r_1, \ldots, r_k, r, x^1, \ldots, x^{m-k-1}\} \in \mathfrak{H}_j$ to $\mathcal{L}_i$, if $\mathcal{H}'$ is a minimal class in $\mathcal{OS}_i$.

We generalise from this example that all extensive classes will become specified apart from the exceptions discussed below. The exclusion probability of a specified extensive hypothesis $\mathcal{H} \in \mathfrak{H}_i$ is exactly $p_{inv}(N, b, m, i)$, even if $\mathcal{H}$ is implicit.

The number of specified extensive hypotheses resulting from $\mathcal{L}_0$ is strictly bounded by b^m. This is due to the fact that $\mathcal{L}_0$ and its extensions are defined according to the classes $\mathfrak{H}_i$ (1) and their class sizes (2).

Exceptions. An *exception* can only arise in point 4. of the computation of $\mathcal{L}_i$ and consists of following cases:

- The extensive class $\mathcal{H}' \in \mathfrak{H}_j$ resulting from specifying a peer in $\mathcal{H} \in \mathfrak{H}_j$ is not minimal in $\mathcal{OS}_i$.
- There are less than $(b-1)$ non-peers in $\mathcal{O}_i$.

We now analyse the effect of exceptions on the number of the sets that will be specified. Let Alice's peers be $\mathcal{H}_A = \{1, 2, 3\}$, $b = 3$ and the considered extensive class be $\mathcal{H} = \{1, 4, x\} \in \mathfrak{H}_2$. If the next observation is no exception (e.g. $\mathcal{O}_i = \{2, 7, 8\}$), then $(b-1) = 2$ specified sets of $\mathfrak{H}_2$ would result from extending $\mathcal{H}$. These sets are $\mathcal{H}' = \{1, 4, 7\}$ and $\mathcal{H}'' = \{1, 4, 8\}$. Assume that $\mathcal{H}''$ is not minimal than only $\mathcal{H}'$ would be the extension of $\mathcal{H}$. Similarly, if the next observation would contain less than $(b-1)$ non-peers, e.g. $\mathcal{O}_i = \{2, 3, 6\}$, respectively $\mathcal{O}_i = \{2, 7\}$. Only one specified set $\mathcal{H}' = \{1, 4, 6\}$ respectively $\mathcal{H}' = \{1, 4, 7\}$ would result from extending $\mathcal{H}$.

Let k be the number of specified peers in $\mathcal{H}$ from point 4., then for each missing non-peer in the next observation $\mathcal{O}_i$, the number of sets that will be specified decreases by at most b^{m-k-1}. The same decrease is caused, if the class $\mathcal{H}'$ resulting from specifying a peer in $\mathcal{H}$ is not minimal in $\mathcal{OS}_i$.

Note that the preconditions for exceptions imply that sets excluded by exceptions are unspecified and implicit before the exclusion. We observe that most extensive hypotheses become specified very fast and logically at least as fast as minimal hitting sets reach the size m. The impact of exclusions by exceptions on the size of the extensive hypothesis set is therefore moderate in comparison to normal (non-exceptional) exclusions. For the sake of simplicity, we only mathematically model the normal exclusion of extensive hypotheses from the initial b^m extensive hypotheses.

The main result of this section is that we can map the inconvenient exclusion process of minimal hitting sets on the exclusion process of the extensive hypothesis set. This again can be simplified to the exclusion process of specified extensive hypothesis set, where the exclusion probability of each set is known. From now on hypotheses and classes are always addressed in terms of specified extensive hypotheses and classes.

4 Modelling Anonymity States

Section 3.3 justified that the evolution of the minimal hitting sets can be modelled by the evolution of the extensive hypothesis set. This section will introduce formulas to describe the deployment of the size and structure of the (extensive) hypothesis set for distinct number of observations and distinct Mix parameters N, b, m. In particular we will answer the following questions:

- How many observation are required to disclose all of Alice's peers?
- How likely is it to find $k \leq m$ of Alice's peers in a random minimal hitting set after t observations?
- What is the average number of observation to disclose at least one peer of Alice?

4.1 Full Disclosure

The *full disclosure* of Alice's peer set is the unambiguous identification of all of Alice's peer recipients, i.e. the identification of $\mathcal{H}_A$.

Mean Number of Minimal Hitting Sets. In this section, we derive closed formulas for the mean number of hypotheses after t observations for distinct classes.

Let V_i be a random variable, where $V_i = 1$ is the event that a particular hypothesis $\mathcal{H}$ of the class $\mathfrak{H}_i$ remains valid after t observations, while $V_i = 0$ denotes the inverse event. The probability of $V_i = 1$ corresponds to t stochastically independent Bernoulli trials, where the outcome of each of the t trials shows that the minimal hitting set remains valid. Thereby a Bernoulli trial corresponds to the outcome, whether $\mathcal{H}$ remains valid at the next collected observation. Since each observation appears stochastically independently from those in the past and in the future, we have a natural mapping of the "remaining valid event" of $\mathcal{H}$ on the independent Bernoulli trials, hence:

$$P(V_i = 1) = [P(\mathcal{H} \textit{ remains valid at next observation})]^t$$

is the probability that $\mathcal{H}$ remains a hypothesis after t observations, which is by (3) exactly $(1 - p_{inv}(N, b, m, i))^t$.

Let $\mathfrak{H}_i = \{\mathcal{H}_1, \ldots, \mathcal{H}_{|\mathfrak{H}_i|}\}$ be a minimal hitting set class containing only hypotheses and V_{i_j} be the event that the hypothesis $\mathcal{H}_j \in \mathfrak{H}_i$ remains valid after t observations. The expectation E of the number of hypotheses in $\mathfrak{H}_i$ after t observations is thus the expectation of the convolution of the random variables $V_{i_1}, \ldots, V_{i_{|\mathfrak{H}_i|}}$. This expectation is represented by (4). Thereby we benefit from the additivity of the expectation function by splitting the complex expectation on the left side to a sum of expectation of single V_{i_j} events on the right side. The probability of the outcome $P(V_{i_j} = 1)$ is identical for each fixed hypothesis $\mathcal{H}_j \in \mathfrak{H}_i$ (i.e. $P(V_{i_j} = 1) = P(V_i = 1)$), hence the right side of the former equation can be simplified to (5).

$$E(V_{i_1}, \ldots, V_{i_{|\mathfrak{H}_i|}}) = \sum_{j=1}^{|\mathfrak{H}_i|} E(V_{i_j}) \tag{4}$$

$$= \sum_{j=1}^{|\mathfrak{H}_i|} P(V_{i_j} = 1)$$

$$= |\mathfrak{H}_i| P(V_i = 1) \ . \tag{5}$$

Note that the events V_{i_j}, V_{i_k} for $i, k \in \{1, \ldots |\mathfrak{H}_i|\}$ and $j \neq k$ are not stochastically independent, hence the probability $P(V_{i_j}) = P(V_i)$ respectively $P(V_{i_k}) = P(V_i)$ only holds, if we consider single events, as on the right side of the equation below.

To clarify that (5) depends on the parameter N, b, m and t we use the more elaborate formulation:

$$E_{|\mathfrak{H}_i|}(N, b, m, t) = |\mathfrak{H}_i|(1 - p_{inv}(N, b, m, i))^t = \binom{m}{i}(b-1)^i(1 - \frac{i}{m}(1 - \frac{m}{N})^{b-1})^t \quad (6)$$

for the expected number of remaining hypotheses of class $\mathfrak{H}_i$.

Formula (4) can be easily extended to cover the mean number of observations for any combination of classes including the consideration of all classes. The expectation of the remaining hypotheses with respect to the initial hypothesis set $\mathfrak{H}$ is:

$$\begin{aligned} E_{|\mathfrak{H}|}(N, b, m, t) &= \sum_{i=0}^{m} \binom{m}{i}(b-1)^i(1 - \frac{i}{m}(1 - \frac{m}{N})^{b-1})^t \\ &\leq ((b-1)e^{-\frac{t}{m}(1-\frac{m}{N})^{b-1}} + 1)^m \quad . \end{aligned} \quad (7)$$

Time to Reduce Hypothesis Set to a Threshold. The expectations $E_{|\mathfrak{H}_i|}$ and $E_{|\mathfrak{H}|}$ of the number of hypotheses after t observations can be easily reformulated to derive the number of observations, such that a hypotheses remains on average.

By a transformation of (6), where a denotes the left side of the equation, we obtain:

$$t_{\mathfrak{H}_i} = \frac{\ln a - \ln \binom{m}{i} - i \ln (b-1)}{\ln (1 - \frac{i}{m}(1 - \frac{m}{N})^{b-1})} \quad \text{for } a > 0 \quad . \quad (8)$$

This equation represents the number of observations, such that at most a hitting sets remain on average in the class $\mathfrak{H}_i$ for $i \geq 1$.

Similarly we reformulate (7) to obtain the number of observations $t_{\mathfrak{H}}$, such that there are on average less than a minimal hitting sets left from the initial hypothesis set $\mathfrak{H}$. Alice's peer set $\mathcal{H}_A$ always remains in $\mathfrak{H}$, therefore $a > 1$.

$$t_{\mathfrak{H}} \leq \frac{m(\ln (b-1) - \ln (a^{1/m} - 1))}{(1 - \frac{m}{N})^{b-1}} \quad \text{for } a > 1 \quad . \quad (9)$$

Comparison to Simulation. This section visualise the precision of the function $t_{\mathfrak{H}}$ of Sect. 4.1 by comparing it with the mean time of full disclosure obtained by our hitting set simulations. The simulation applies the HS-Attack on simulated observations, until Alice's peer set can be uniquely identified. This simulation is run several times to obtain a confidence interval of 95% on the mean number of observations to identify Alice's peer set. The observations are generated under the assumption of a uniformly distributed communication of Alice and the other senders. That is Alice chooses one of her m peers with probability $\frac{1}{m}$ and each of the other senders chooses its peer from all N receivers with the probability $\frac{1}{N}$ at each round. This distribution complies to the distribution used in our formulas.

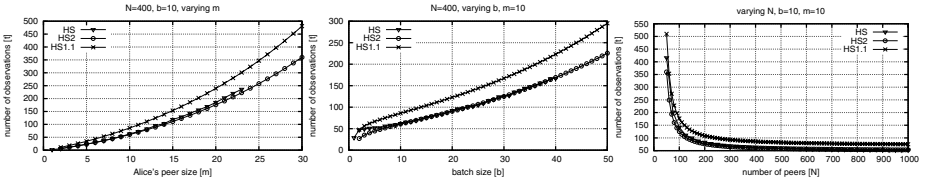


Fig. 2. Number of observations: Full disclosure by simulation (HS), reduction of hypothesis set to size below 2 (HS2) and below 1.1 (HS1.1)

The plots in Fig. 2 compare the mean number of observations for full disclosure obtained by the simulation (HS) with the number of observations to reduce the initial hypothesis set to a cardinality less than 2 (HS2) respectively less than 1.1 (HS1.1) using (9). The y-axes of the plots shows the number of observations, while the x-axes vary one of the parameters N,b, or m.

Note that the mean number of observations for full disclosure is not necessary equal to the number of observation to reduce the hypothesis set to a particular size, although these two values are strongly related to each other. Let μ_{dis} be the mean number of observations for the full disclosure, then the mean number of hypotheses after μ_{dis} observations is obviously larger than 1. Depending on the variance of the number of hypotheses around μ_{dis}, more than μ_{dis} observations are necessary to keep the number of hypotheses closed to 1. This is shown by the Fig. 2. The HS2 curve is almost identical to the HS curve, whereas the HS1.1 curve is noticeably above the HS curve.

Parallel to us [25] suggested a "lower bound" t^* for the mean number of observations for full disclosure using the same Mix and attacker model. They compute for each of the $\binom{N}{m}$ "normal" (not extensive) hypotheses the probability that it is excluded after t observations and claim that t^* is the lower bound of the time when only one hypothesis remains. Figure 3 shows that t^* is far away from being a lower bound. Firstly, there are at most $\binom{N}{m} - \binom{N-b}{m}$ "normal" hypotheses after the first observation due to the Mix model, but this restriction is not in their mathemtic model and causes a significant overestimation. Secondly, even if this would be corrected, the evolution of the "normal" hypothesis set depends on the distribution of the hypotheses' structures remaining after the first observation and those succeeding that. This is not mathematically modeled and might be very difficult to do.

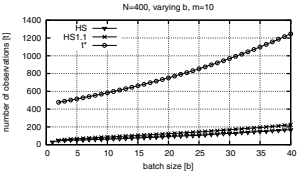


Fig. 3. Comparison: Simulation (HS), reducing hypothesis set below 1.1 (HS1.1), t^* (t*)

4.2 Partial Disclosure

The *partial disclosure* is the unambiguous identification of a subset $\mathcal{H}_{A'} \subseteq \mathcal{H}_A$ of Alice's peer set. The full disclosure in Sect. 4.1 is a special case of the partial disclosure.

Probability to Identify k Particular Peers. The probability to identify k particular peers $\mathcal{H}_{A'} \subseteq \mathcal{H}_A$ of Alice after at most t observations is the probability that all hypotheses are excluded that do not contain all of these $k = |\mathcal{H}_{A'}|$ peers after at most

t observations. This probability is a discrete distribution with respect to t and we will address it by the term f_{id}. The probability to exclude a particular hypothesis depends on its class, therefore we will first determine the number of hypotheses of each class $\mathfrak{H}_i$ that have to be excluded.

Number of Exclusions in a Class. We remember from (2) that the size of $\mathfrak{H}_i$ is $|\mathfrak{H}_i| = \binom{m}{i}(b-1)^i$. In this class i of the m peers of Alice are replaced by non-peers. Therefore $\binom{m-k}{i}(b-1)^i$ is the number of hypotheses in the class $\mathfrak{H}_i$, where the k of Alice's peers $\mathcal{H}_{A'}$ are not replaced by non-peers. The number of hypotheses in $\mathfrak{H}_i$ that have to be excluded to enable the identification of $\mathcal{H}_{A'}$ is therefore:

$$exNo_i(b,m,k,i) = \left(\binom{m}{i} - \binom{m-k}{i}\right)(b-1)^i \quad . \tag{10}$$

Note that we distinguish between the to be excluded hypotheses with respect to their class membership, because the probability to exclude a hypothesis depends on its membership as shown in Sect. 3.2. Also note that we only know the probability with respect to the exclusion of a single hypothesis. If two hypotheses $\mathcal{H}_1$ and $\mathcal{H}_2$ are considered, then there could be a stochastic dependency between them, i.e. if $\mathcal{H}_1$ is excluded, then $\mathcal{H}_2$ might be excluded, too. For that reason we make the simplifying assumption that all minimal hitting sets are stochastically independent. This enables us to unrestrictedly apply p_{inv} to describe the exclusion of hypotheses. The following equation derives the distribution f_{id} with respect to the parameters N, b, m, t and the number $k = |\mathcal{H}_{A'}|$ of Alice's peers that should be identified.

$$\begin{aligned} f_{id}(N,b,m,k,t) = &\prod_{i=1}^{m-k} (1-(1-p_{inv}(N,b,m,i))^t)^{(\binom{m}{i}-\binom{m-k}{i})(b-1)^i} \\ &\prod_{i=m-k+1}^{m} (1-(1-p_{inv}(N,b,m,i))^t)^{\binom{m}{i}(b-1)^i} \quad . \end{aligned} \tag{11}$$

Probability to Identify at Least k Peers. Based on the function f_{id} of the last section, we will derive the probability distribution $f_{id_{any}}$ that at least k of Alice's peers can be identified after at most t observations. In contrast to the previous section we are not focusing on disclosing particular peers, but on the probability to disclose a certain number of peers.

Let Y^k be a random variable denoting the event that particular k peers of Alice's peer set $\mathcal{H}_A$ are identified, i.e. $Y^k = 1$ if the designated peers are identified else $Y^k = 0$ for the inverse case. To simplify the notation we will abbreviate the probability $P(Y^k = 1)$ by the term $P(Y^k)$.

Let $Y_1^k, \ldots, Y_{\binom{m}{k}}^k$ be $\binom{m}{k}$ distinct random variables. Each of this variable represents the event that distinct subsets of $\mathcal{H}_A$ of cardinality k are identified. In order to compute the probability that at least k of Alice's peers can be disclosed, we have to determine the probability that any of these Y_i^k events, for $i \in \{1, \ldots, \binom{m}{k}\}$ takes place. Thereby it would be imprecise to simply sum up the probabilities $P(Y_i^k)$ for $i \in \{1, \ldots, \binom{m}{k}\}$,

because the events Y_i^k are not stochastically independent. We can solve this problem by applying the inclusion-exclusion-formula:

$$P(Y_1^k \vee \ldots \vee Y_{\binom{m}{k}}^k) = P(Y_1^k) + \ldots + P(Y_{\binom{m}{k}}^k) \\ - P(Y_1^k, Y_2^k) - \ldots - P(Y_{\binom{m}{k}-1}^k, Y_{\binom{m}{k}}^k) \ldots + \ldots - \ldots \ . \tag{12}$$

Assume that $\{a_{i_1}, a_{i_2}\}$ and $\{a_{j_1}, a_{j_2}\}$ are those peers that are identified by the event Y_i^k respectively Y_j^k (for $k = 2$). The above term $P(Y_i^k, Y_j^k)$ is the probability that all peers of the joint set $\{a_{i_1}, a_{i_2}, a_{j_1}, a_{j_2}\}$ are identified. Let us denote the joint event by the term $Y^{k'}$, where $k' = |\{a_{i_1}, a_{i_2}, a_{j_1}, a_{j_2}\}| \leq 2k$, then $P(Y_i^k, Y_j^k) = P(Y^{k'})$ can be computed by (11). It is also obvious that this transformation can even be applied to an arbitrary number of joints of events, i.e. we can transform $P(Y_1^k, \ldots, Y_z^k)$ to $P(Y^{k'})$ for any $z \geq 1$ accordingly.

The next formula is an elaborate formulation of (12) for the special case of $k = 1$. It is the probability to identify at least one of Alice's peers after at most t observations.

$$f_{id_{any}}(N, b, m, t, 1) = \sum_{s=1}^{m} \left((-1)^{s-1} \binom{m}{s} f_{id}(N, b, m, s, t) \right) \ . \tag{13}$$

The general probability distribution for arbitrary values of k, where $k \leq m$ is the least number of peers that are to be disclosed is:

$$f_{id_{any}}(N, b, m, t, k) = \sum_{i=1}^{\binom{m}{k}} (-1)^{i-1} \sum_{j_1=1}^{\binom{m}{i}-(i-1)} \cdots \sum_{j_i=j_{i-1}+1}^{\binom{m}{i}-(i-i)} f_{id}(N, b, m, |\bigcup_{z=1}^{k} Y_{j_z}|, t) \ ,$$

where $\bigcup_{z=1}^{k} Y_{j_z}$ is the union of the set of peers identified by each Y_{j_z}.

Given the distribution $f_{id_{any}}(N, b, m, t, k)$, the probability that at least k peers can be identified after exactly t observation is:

$$p_{id_{any}}(N, b, m, t, k) = f_{id_{any}}(N, b, m, t, k) - f_{id_{any}}(N, b, m, t-1, k) \ .$$

Mean Time to Deanonymization. We are now able to provide the first existing formula to compute the mean number of observations to unambiguously identify at least k of Alice's peer, which we call MTTD-k.

$$E_{id_{any}}(N, b, m, t, k) = \sum_{t=1}^{\infty} t \, p_{id_{any}}(N, b, m, t, k) \ . \tag{14}$$

Note that in particular $E_{id_{any}}(N, b, m, t, 1)$, which is the mean number of observations needed to identify at least one of Alice's peers (MTTD-1) should be considered as a more appropriate measurement of the lower bound of the anonymity provided by Mix systems. This is justified by the fact that MTTD-1 measures the time point, when the attacker gains the first unambiguous information about Alice's communication partners and thus breaks the anonymity function of the Mix. In contrast to this, full disclosure,

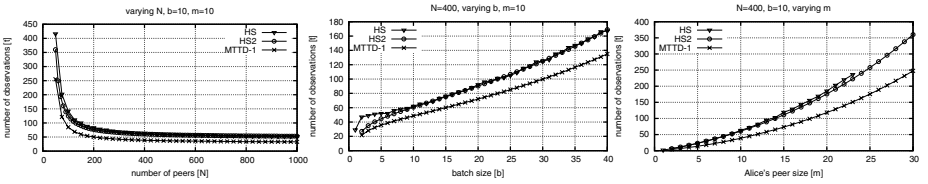


Fig. 4. Disclosure of at least one peer (MTTD-1), simulated full disclosure (HS), reduction of hypothesis set size below 2 (HS2)

or the number of observations to reduce the hypothesis set below a particular size a can not expose this threat.

Figure 4 compares the expected number of observations to disclose at least one peer (MTTD-1) by using $E_{id_{any}}(N, b, m, t, 1)$ with the simulation result for the mean number of observations for full disclosure (HS) and the mean number of observation to reduce the hypothesis set to a size below 2 (HS2) computed by (9). The comparison is with respect to different parameters N, b, m. We can see that the partial disclosure (MTTD-1) appears noticeable earlier than full disclosure (HS) and before the hypothesis set is reduced to a size below 2. This difference increases, the more observations are required for full disclosure and shows that full disclosure alone is insufficient to measure anonymity.

4.3 Beyond Unambiguous Identification of Peers

Our model also enables us to consider the number of Alice's peers contained in any computed minimal hitting set after a particular number of observations.

Figure 5 plots the number of observations to reduce each minimal hitting class $\mathfrak{H}_i$ to a size less than a by using (8). The figure shows this for $a = 1$ by the HS1 curve and for $a = 0.1$ by the HS0.1 curve for the Mix parameters $N = 400, b = 10, m = 10$.

We can see that minimal hitting set classes $\mathfrak{H}_i$ containing less of Alice's peers are reduced earlier than those containing more of Alice's peers. Thus after a particular number of observations t, the number of hypotheses containing less than k Alice's peers are negligible, since $E_{|\mathfrak{H}_i|}(t) < a$ for $i > (m - k)$. In particular our plot shows that after about $t = 40$ observations, the attacker will unlikely find a minimal hitting set containing less than 7 of Alice's peers. Thus any minimal hitting set computed by the attack contains with a high probability at least 70% of the peers of Alice. If we assume that minimal hitting sets are excluded stochastic independently from each other, then the probability to find at least k of m peers of Alice after at most t observations is:

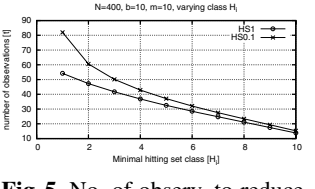


Fig. 5. No. of observ. to reduce size of $\mathfrak{H}_i$ below 1 (HS1) and 0.1 (HS0.1).

$$f_{id_k}(N, b, m, k, t) \geq \prod_{i=m-k-1}^{m} \left(1 - \left(1 - p_{inv}(N, b, m, i)\right)^t\right)^{|\mathfrak{H}_i|} .$$

5 Conclusions

In this work, we investigated the fundamental structures for anonymity that we identified as the hypothesis set. The analysis of the hypothesis set is made under the assumption of a uniformly distributed communication of the senders and of static peer sets. This assumption is chosen in a way, such that we obtain a conservative consideration of anonymity, which can be considered as a lower bound of the anonymity provided by Mixes in the real world.

Based on the ExactHS [23], we derived a comprehensive mathematic model to probabilistically describe the inherent properties of the hypothesis set in detail. In particular we estimated in Sect. 4 the size of the hypothesis set, the structure of hypotheses in it, the size of those structures and the probability that particular hypotheses are included in it, with respect to the parameters N,b,m at any number of observations. This information enabled a fine granular measurement of anonymity that also measures those protection states before the point of full disclosure. In particular MTTD-k introduced in Sect. 4.2 determined the mean number of observations to disclose from one to all Alice's peers. The evaluations of MTTD -1 (see Fig. 4) showed that the first unambiguous knowledge about one of Alice's peers can be gained noticeably before full disclosure. It is therefore not sufficient to solely consider full disclosure (which is the focus of all existing hypothesis set based approaches e.g. [5, 23, 25]) for anonymity measurement.

Furthermore, our model even enabled an analysis granularity beyond the scope of unambiguous identification of peers. This is shown in Sect. 4.3, which provided the probability to find a certain number of Alice's peers in randomly computed hypotheses. This insight opens the door for a new refined metric, which also covers unambiguous information in the anonymity consideration.

We also showed that our mathematic model and the resulting measurements were precise and meaningful by comparison to simulations. All results were in reasonable scopes and reflected the right relations to the Mix parameters N, b, m and the number of observations t.

The elementary model and analyses of our work are important for designers and users of Mix networks. It enables Alice to estimate how much information she is going to leak about her peers with each of her communications. That way she knows when to stop communicating, or to add dummy traffic to remain information theoretic anonymous, such that even attackers with unlimited computing power cannot reveal her peers.

In the future we intend to integrate the leakage of unambiguous and ambiguous information about Alice's peers in one metric. In conjunction with this, we will analyse the uncertainty caused by dummy traffic. Finally we will refine our model to expand the analysis beyond the uniform communication assumption to get closer to the real world.

References

[1] Chaum, D.L.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–88 (1981)

[2] Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 1–9. Springer, Heidelberg (2001)

[3] Eusgeld, I., Freiling, F.C., Reussner, R. (eds.): Dependability Metrics. LNCS, vol. 4909. Springer, Heidelberg (2008)
[4] Shannon, C.E.: Communication theory of secrecy systems. Bell Syst. Tech. J. 28, 656–715 (1949)
[5] Kesdogan, D., Agrawal, D., Pham, V., Rauterbach, D.: Fundamental Limits on the Anonymity Provided by the Mix Technique. In: IEEE Symposium on Security and Privacy (May 2006)
[6] Clauß, S., Schiffner, S.: Structuring Anonymity Metrics. In: DIM 2006: Proceedings of the second ACM workshop on Digital identity management, pp. 55–62 (2006)
[7] Deng, Y., Pang, J., Wu, P.: Measuring Anonymity with Relative Entropy. In: Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S. (eds.) FAST 2006. LNCS, vol. 4691, pp. 65–79. Springer, Heidelberg (2007)
[8] Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards Measuring Anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
[9] Edman, M., Sivrikaya, F., Yener, B.: A Combinatorial Approach to Measuring Anonymity, 356–363 (2007)
[10] Serjantov, A., Danezis, G.: Towards an Information Theoretic Metric for Anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 259–263. Springer, Heidelberg (2003)
[11] Tóth, G., Hornák, Z., Vajda, F.: Measuring Anonymity Revisited. In: Proceedings of the Ninth Nordic Workshop on Secure IT Systems, pp. 85–90 (November 2004)
[12] Zhu, Y., Bettati, R.: Anonymity vs. Information Leakage in Anonymity Systems. In: ICDCS 2005: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, pp. 514–524 (2005)
[13] Clarkson, M.R., Myers, A.C., Schneider, F.B.: Belief in Information Flow. In: Proceedings of the 18th IEEE workshop on Computer Security Foundations, pp. 31–45 (2005)
[14] Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security 1, 66–92 (1998)
[15] Mantel, H.: A Uniform Framework for the Formal Specification and Verification of Information Flow Security. PhD thesis, Universität des Saarlandes (July 2003)
[16] Schneider, S., Sidiropoulos, A.: CSP and Anonymity. In: Martella, G., Kurth, H., Montolivo, E., Bertino, E. (eds.) ESORICS 1996. LNCS, vol. 1146, pp. 198–218. Springer, Heidelberg (1996)
[17] Halpern, J.Y., O'Neill, K.R.: Anonymity and Information Hiding in Multiagent Systems 13, 483–514 (2005)
[18] Hughes, D., Shmatikov, V.: Information Hiding, Anonymity and Privacy: a Modular Approach. J. Comput. Secur. 12, 3–36 (2004)
[19] Padlipsky, M.A., Snow, D.W., Karger, P.A.: Limitations of End-to-End Encryption in Secure Computer Networks. Technical Report ESD-TR-78-158 (August 1978)
[20] Pfitzmann, A.: Diensteintegrierende Kommunikationsnetze mit teilnehmerüberprüfbarem Datenschutz. Informatik-Fachberichte, vol. 234 (1990)
[21] Kesdogan, D., Pimenidis, L.: The Hitting Set Attack on Anonymity Protocols. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 326–339. Springer, Heidelberg (2004)
[22] Kesdogan, D., Agrawal, D., Penz, S.: Limits of Anonymity in Open Environments. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 53–69. Springer, Heidelberg (2003)
[23] Pham, V.: Analysis of the Anonymity Set of Chaumian Mixes. In: 13th Nordic Workshop on Secure IT-Systems (October 2008)
[24] Pham, D.V.: Analysis of Attacks on Chaumian Mixes (Analyse von Angriffen auf Chaummixen). Master's thesis, RWTH-Aachen (April 2006)
[25] O'Connor, L.: Entropy Bounds for Traffic Confirmation. Cryptology ePrint Archive (2008), http://eprint.iacr.org/2008/

On Improving the Accuracy and Performance of Content-Based File Type Identification

Irfan Ahmed[1], Kyung-suk Lhee[1], Hyunjung Shin[2], and ManPyo Hong[1]

[1] Digital Vaccine and Internet Immune System Lab
Graduate School of Information and Communication,
Ajou University, South Korea
{irfan,klhee,mphong}@ajou.ac.kr
[2] Department of Industrial and Information Systems Engineering,
Ajou University, South Korea
shin@ajou.ac.kr

Abstract. Types of files (text, executables, Jpeg images, etc.) can be identified through file extension, magic number, or other header information in the file. However, they are easy to be tampered or corrupted so cannot be trusted as secure ways to identify file types.In the presence of adversaries, analyzing the file content may be a more reliable way to identify file types, but existing approaches of file type analysis still need to be improved in terms of accuracy and speed. Most of them use byte-frequency distribution as a feature in building a representative model of a file type, and apply a distance metric to compare the model with byte-frequency distribution of the file in question. Mahalanobis distance is the most popular distance metric. In this paper, we propose 1) the cosine similarity as a better metric than Mahalanobis distance in terms of classification accuracy, smaller model size, and faster detection rate, and 2) a new type-identification scheme that applies recursive steps to identify types of files. We compare the cosine similarity to Mahalanobis distance using Wei-Hen Li et al.'s single and multi-centroid modeling techniques, which showed 4.8% and 13.10% improvement in classification accuracy (single and multi-centroid respectively). The cosine similarity showed reduction of the model size by about 90% and improvement in the detection speed by 11%. Our proposed type identification scheme showed 37.78% and 31.47% improvement over Wei-Hen Li's single and multi-centroid modeling techniques respectively.

Keywords: file type identification, byte frequency distribution, cosine similarity, Mahalanobis distance, linear discriminant, cluster analysis.

1 Introduction

File type identification is an important task for many security applications to work efficiently. For example, filtering email attachments may require blocking inbound attachment types that may contain malicious contents. Virus scanners may be configured to skip some file extensions; in Norton Antivirus 2008 for

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 44–59, 2009.

example, we have an exclusion option to specify file types to skip for virus scan [1]. Some applications raise alerts before opening unrecognized (therefore suspicious) file extensions; for instance, Windows Media player raises an alert when user tries to open a file of unrecognized extension. Some steganalysis programs also rely on file type detection; for example, stegdetect [2], which detects steganographic contents in images, uses libmagic1 package [3] to determine file type using magic numbers. However, they are easy to be tampered or corrupted so cannot be trusted as secure ways to identify file types. In the presence of adversaries, analyzing the file content may be a more reliable way to identify file types. Solutions that analyze file content usually build a representative model of a file type by averaging the byte frequency distributions of sample files, and use a measure to find out the consistency between the model and the byte frequency distribution of the file in question. Moreover, mahalanobis distance is a popular measure to compare byte frequency distributions such as it is used in PAYL [4], Fileprint [6], [5], [7]. However, existing approaches of file type analysis still need to be improved in terms of accuracy and speed.

In this paper, we propose two schemes to improve the analysis of file content for type identification. First, we show that the cosine similarity [8] is a better metric than mahalanobis distance in terms of classification accuracy, smaller model size and faster detection rate. Secondly, we propose a new type identification scheme that recursively refines the type of a file in question.

We use Wei-Hen Li et al [6]'s modeling technique to compare the cosine similarity (CS) and Mahalanobis distance (MD). Wei-Hen Li's work is a representative modeling technique in that it uses MD and its single centroid model is also used by other schemes [9], [10], [11]. Its multi-centroid model is comparable to our recursive scheme.

In our proposed type identification scheme, we group files with similar byte-frequency pattern irrespective of their file types (whereas Wei-Hen Li's technique groups files with the same type). Then we apply the linear discriminant analysis [12] to identify the type of each file. Our proposed scheme shows better result than Wei-Hen Li's single and multi-centroid modeling technique (considering both CS and MD as distance metric).

This paper is organized as follows. Section 2 presents the related work. Section 3 describes the assumptions that are required for robust techniques of byte-frequency analysis, and explains the dataset used in this paper. Section 4 explains our experiment results on cosine similarity and Mahalanobis distance. Section 5 discusses the weaknesses of Wei-Hen Li's modeling technique. Section 6 presents our recursive file-type identification scheme. Section 7 analyzes the proposed scheme. In section 8, we compare the Wei-Hen Li's single and multi-centroid models (using both CS and MD) with the proposed scheme, and Section 9 shows the conclusions and future work.

2 Related Work

This section discusses the previous techniques for file type identification. Some of these techniques are commonly used by operating systems where the file type

information is explicitly written either in the file header or in the file name. There are other techniques which analyze the file content to identify the file type.

The most common and the simplest way to identify file type is to look at the file's extension [13], but this can be easily spoofed by users with malicious intent. Novice users could also unintentionally change the file extension while renaming the file name. Malwares could also easily hide themselves by having file extensions that virus scanners skip.

Another method to identify file type is to look at magic numbers in the file [14]. For example, GIF files begin with ASCII representation of either GIF87a or GIF89a depending on the standard. ZIP files always have the magic number "PK" at the beginning of the file. However, only binary files have magic numbers so it is not applicable to text files. As with the file extension, it can also be easily spoofed. For instance, malcodes can be hidden using code obfuscation and alphanumeric encoding [15], [16], [17], [18].

McDaniel and Heydari [19] introduce three algorithms to identify file types by analyzing file content. In byte frequency analysis algorithm (BFA), they calculate byte-frequency distribution of different files and generate "fingerprint" of each file type by averaging byte-frequency distribution of their respective files. They also calculate correlation strength as another characterizing factor. They take the difference of the same byte in different files. If the difference gets smaller, the correlation strength increases towards 1 or vice versa. In byte-frequency cross-correlation algorithm, they find the correlation between all byte pairs. They calculate the average frequency between all byte pairs and correlation strength similar to the BFA algorithm. In file header/trailer algorithm, the file headers and trailers are byte patterns that appear in a fixed location at the beginning and end of a file. They maintain an array of 256 for each location and the array entry corresponding to the value of the byte is filled with correlation strength of 1. They construct the fingerprint by averaging the correlation strength of each file. In these algorithms, they compare the file with all the generated fingerprints to identify its file type.

Wei-Hen Li et al. identify file types using n-gram analysis. They calculate 1-gram frequency distribution of files and build 3 different models of each file type: single centroid (one model of each file type), multi-centroid (multiple models of each file type), and exemplar files (set of files of each file type) as centroid. They refer them as "fileprint". In single and multi-centroid models, they calculate mean and standard deviation of 1-gram frequency distribution of files, and use Mahalanobis distance to compare these models with 1-gram distribution of given file to find the closest model. In exemplar file model, they compare 1-gram distribution of exemplar file with that of given file (there is no variance computed), and Manhattan distance is used instead of Mahalanobis distance. Their solution cannot identify files having similar byte-frequency distributions such as MS Office file formats (such as Word and Excel) but treat them as one group or one abstract file type.

Karresand and Shahmehri [9], [10] proposed the *"Oscar"* method for identifying the types of file fragments. They build the single centroid fileprints [6] but use quadratic distance metric and 1-norm as distance metric to compare the centroid with the byte frequency-distribution of file. Although Oscar identifies any file type, they optimized their algorithm for JPG file using specific byte pairs in the file, and reported 99.2% detection rate with no false positives. They also use rate of change of bytes, i.e. the difference of two consecutive byte values where they consider the ordering information of bytes.

Veenman [11] extracts three features from file content. These features are 1) byte frequency distribution, 2) entropy derived from byte-frequency distribution of files, and 3) the algorithmic or Kolmogorov complexity that exploits the substring order [20]. The Fisher linear discriminant is applied to these features to identify the file type.

Calhoun and Coles [21] extended Veenman's work by building classification models (based on the ASCII frequency, entropy, and other statistics) and apply linear discriminant to identify file types. They also argued that files of same type probably have longer substrings in common than that of different types. Our recursive scheme also uses the byte-frequency distribution as a feature and linear discriminant analysis for classification. However, the main difference of ours from Veenman's scheme is the way we build the classification model for each file type. Veenman computes one discriminant function for each file type using all its sample files. However, our scheme combines the similar byte frequency files in groups irrespective of their file types using clustering, and computes the linear discriminant function for each file type in each group. Hence multiple functions could be computed for each file type. Veenman reported 45% overall accuracy while our scheme shows 77% overall accuracy.

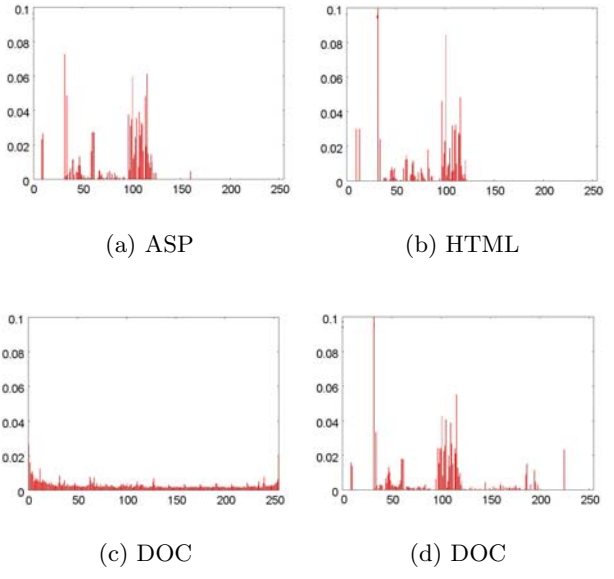
3 Assumptions and Dataset Details

3.1 Assumptions for Byte-Frequency Distributions of File Types

We make two assumptions for byte-frequency distribution of files.

1. Byte-frequency distributions of different file types could be homogeneous. That is, they could have similar byte frequency distributions (Figure 1(a) and 1(b))
2. Byte-frequency distributions of a file type could be heterogeneous. That is, files of the same type could have different byte-frequency distributions (Figure 1(c) and 1(d)).

We believe that a robust solution for file-type identification, which uses byte-frequency distribution to build representative models of file types, should satisfy these assumptions. The proposed recursive scheme is based on these assumptions.



(a) ASP (b) HTML

(c) DOC (d) DOC

Fig. 1. a) and b) show that ASP and HTML can have similar byte-frequency patterns. c) and d) show that two DOC files can have different byte-frequency patterns (X-axis represents the byte patterns and y-axis the relative byte frequency).

3.2 Dataset

In our experiments, we used two datasets to analyze different file-type identification schemes and our scheme. One dataset is used to build the representative models of file types and the other to test the classification accuracy of the identification schemes. 10 file types (JPG, HTML, GIF, EXE, MP3, PDF, TXT, DOC, XLS, and ASP) are included in each dataset, with each type having 100 files. These file types are chosen since they are popularly used and cover a broad range of file contents such as binary files (JPG, GIF, EXE, MP3, PDF), printable character files (TXT, HTML, ASP), and the files that contain binary and characters (DOC and XLS).

4 Cosine Similarity: An Efficient Measure to Compare Byte Frequency Vectors

4.1 Cosine Similarity

Cosine similarity is a measure to compare two vectors. If is a byte-frequency vector of a test file and is the representative model vector of a file type (that is achieved by averaging the byte-frequency distributions of the sample files of the file type), cosine similarity is defined by Eq. 1.

$$cos(x, \bar{y}) = \frac{x.\bar{y}}{|x||\bar{y}|} \tag{1}$$

Where . indicates the vector dot product, $x.\bar{y} = \sum_{k=1}^{n} x_k \bar{y}_k$ and $|x| = \sqrt{\sum_{k=1}^{n} x_k^2} = \sqrt{x.x}$ is the length of vector , The value of cosine similarity lies between 0 and 1. If the cosine similarity is 1, cosine angle between x and $\bar{y}$ is 0, thus x and $\bar{y}$ is the same except the magnitude. If the cosine similarity is 0, cosine angle between x and $\bar{y}$ is 90°, which means they are absolutely dissimilar to each other.

Cosine similarity is quite different from Mahalanobis distance, as it does not use standard deviation but it takes account of the varying size of files. Whereas the byte-frequency distribution of a file needs to be normalized (by dividing each frequency with file size) in order to use Mahalanobis distance, there is no need to normalize the files when using cosine similarity.

In our experimental results presented in next section we found that, unlike Mahalanobis distance, cosine similarity does not lose accuracy even if we truncate the representative model of a file type (so that it contains a subset of highly occurring byte patterns). Thus, using a small number of byte patterns makes it more efficient as it requires less memory and computation without losing accuracy.

4.2 Performance Comparison between Cosine Similarity and Mahalanobis Distance

We use Wei-Hen Li's modeling technique in comparing the classification accuracy of cosine similarity and Mahalanobis distance. The reasons we use their technique are that it uses Mahalanobis distance, its single-centroid model is a representative technique also used by other schemes [9], [10], [11], and its multi-centroid model is comparable to our recursive scheme presented in later section. They build representative model of file type, by combining the byte-frequency distribution of several files of the same type (considering each byte pattern in the distribution as a variable) and averaging their relative frequencies. It is referred as single-centroid model. They also proposed to build multiple representative models of a file type in a similar fashion, but in this case they further group the files of a same type based on their byte-frequency distributions. Such models are referred as multi-centroid model.

We applied, for each file type, multiple truncated models that contain only subsets of byte patterns. For this, we first sorted the byte patterns of a file type (in the order of highest occurring to lowest) and then create multiple models by gradually reducing the number of byte patterns. The rationale behind this is that byte-patterns that rarely occur in a file type may not be representative of that type, but rather be noises. Hence excluding them from the model may be beneficial since it requires smaller memory and computation, and may even increase the classification accuracy in some cases

We also use the simplified Mahalanobis distance formula that is used in Wei-Hen Li's work, which is defined as follows.

$$d(x, \bar{y}) = \frac{\sum_{i=0}^{n-1} (|x_i - .\bar{y}_i|)}{\bar{\delta}_i} \tag{2}$$

where x is a byte-frequency vector of a test file, and $\bar{y}$ and $\bar{\delta}$ are the mean value and standard deviation of the byte-frequency distributions of the sample files (used to build the model of a file type).

Figure 2 (from a to d) shows the classification accuracy of Mahalanobis distance and cosine similarity (on some of the 10 sample file types as mentioned in section 3.2) using different percentages of byte-patterns of single- and multi-centroid models). We used K-means algorithm (as used in Wei-Hen Li's work) to build multi-centroid models. We chose a random value of K (3 in this experiment) as Wei-Hen Li et al. reported that they observed similar results on different values of K. We observed that, in most of the file types, the classification accuracy is degraded as we decrease the number of byte-patterns in the model when using Mahalanobis distance. However using cosine similarity, the classification accuracy either remains the same or shows some improvement.

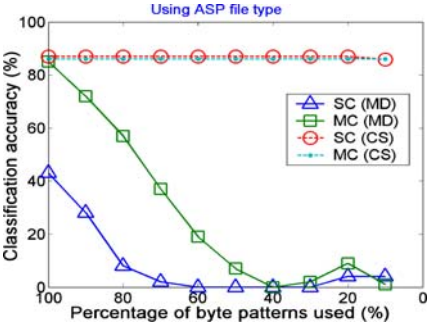
Figure 2 (e) shows the average classification accuracy of Mahalanobis distance and cosine similarity. We can observe that cosine similarity shows better accuracy than Mahalanobis distance, and the multi-centroid model shows better accuracy than single-centroid model. We can also observe that the cosine similarity shows not much difference in classification accuracy even if only 10% of (high frequency) byte-patterns are used. This experiment suggests that we may use a small percentage of high frequency byte-patterns, which reduces the model size and also improve the computation time to compute the cosine similarity values.

Figure 2 (f) shows that by using only 10% of byte-patterns the elapsed time is reduced by approximately 11%. Our experiment considers each byte-pattern as a variable (that is, 1-gram analysis). Since the size of model using 1-gram frequency distribution is small (there are only 256 patterns), there is not much opportunity for improvement on memory space and time. However, models using higher n-grams [22], [23] (i.e. a variable consists of a sequence of multiple bytes) are much bigger since the number of distinct patterns grows exponentially. Therefore the improvement in model size and computation time will be much greater if we use higher n-gram (higher n-gram is out of the scope of this paper).

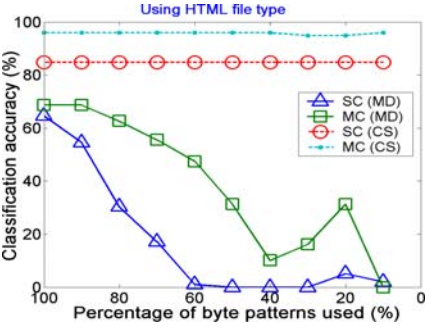
5 Accuracy of Wei-Hen Li et al.'s Single- and Multi-centroid Models

Figure 2 (e) shows that the average classification accuracy using Wei-Hen Li's modeling technique is less than 80%, which may not be satisfactory for many applications. This section presents a discussion of why we think it is so. Our reasoning below is the basis of our proposed scheme in the next section.

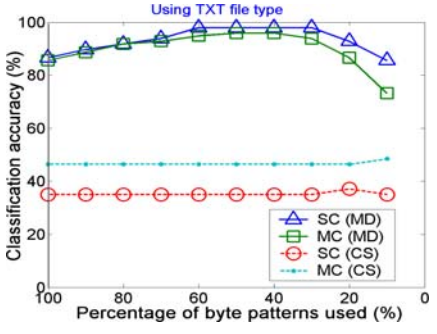
In section 3, we argue that files of a same type could have different byte-frequency patterns. In such case, averaging the different byte-frequency patterns i.e. building a single-centroid model would not yield an accurate representative model (figure 3). In multi-centroid model, a model is built using the files of the same type having similar byte frequency patterns. In section 3, we also argue that files of different types could have similar byte-frequency patterns. Hence it is possible that similar models of different file types are built (figure 4).



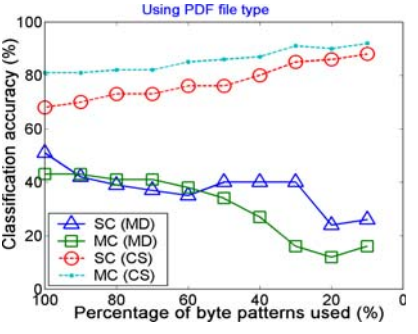
(a) Comparison of MD and CS on SC and MC models using ASP files



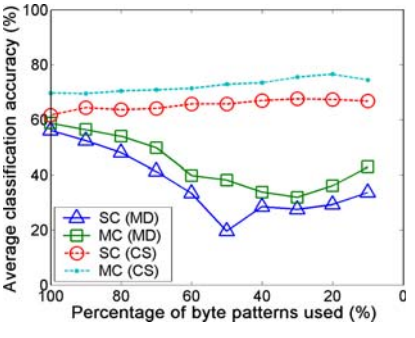
(b) Comparison of MD and CS on SC and MC models using HTML files



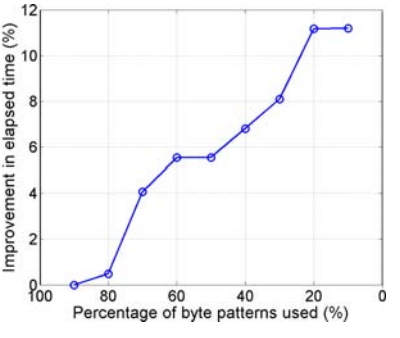
(c) Comparison of MD and CS on SC and MC models using TXT files



(d) Comparison of MD and CS on SC and MC models using PDF files

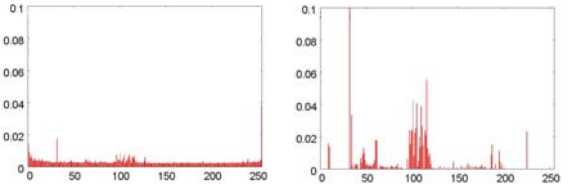


(e) Average classification accuracy of MD and CS on SC and MC models using 10 file types mentioned in figures (a to j)



(f) Improvement in elapsed time of file type detection by processing 1500 files (419MB of total size) on different percentages of byte patterns of representative models.

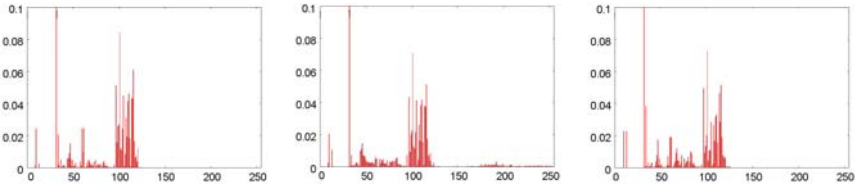
Fig. 2. a) and b) show that ASP and HTML can have similar byte-frequency patterns. c) and d) show that two DOC files can have different byte-frequency patterns (X-axis represents the byte patterns and y-axis the relative byte frequency).



(a) DOC single centroid model

(b) A particular DOC file

Fig. 3. Inaccurate single centroid model of the DOC file type. (x-axis represents byte patterns and y-axis is the relative byte frequency).



(a) HTML (b) TXT (c) ASP

Fig. 4. Similar multi-centroid models of different file types. (x-axis represents byte-patterns and y-axis is the relative byte frequency).

Therefore, when identifying the type of a file, both algorithms (i.e. inaccurate single-centroid model and similar multi-centroid models of different file types) are easily confused with similar nature of other file types (such as TXT, ASP, and HTML) and produces inaccurate results.

6 The Proposed File-Type Identification Scheme

A weakness in Wei-Hen Li's multi-centroid model is that it does not cope well with different types of files having similar byte-frequency distributions. Based on this observation, we propose a scheme that applies recursive steps to classify file types. Our scheme builds a representative model in two steps; we first divide files of varying types (in the sample space) into a few clusters, each of which contain files with similar byte-frequency patterns irrespective of their actual types. Then we apply linear discriminant analysis to find the distinguishing byte-patterns (i.e. a model of a file type) from similar byte-frequency distributions in each cluster.

6.1 Building a Representative Model of a File Type

Clustering is an exploratory statistical procedure to naturally group the data into different clusters. At this stage we do not make any consideration on file types.

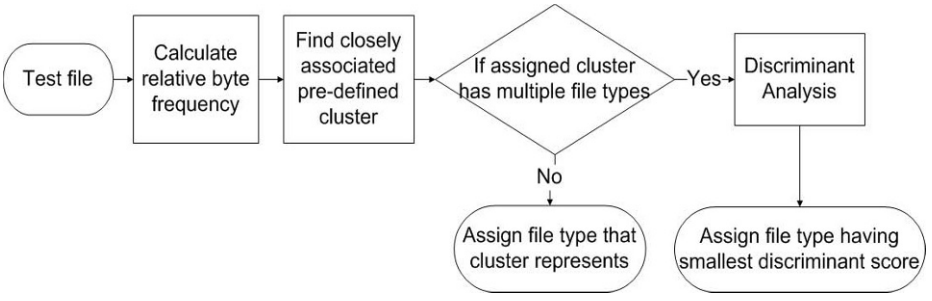


Fig. 5. Identifying a test file using our proposed scheme

After the clusters are built, each cluster could have files with many different types (although the files have similar byte-pattern frequencies). Therefore we need to further divide the type of files in each cluster. For this we perform linear discriminant analysis (LDA), which finds linear combination of byte-patterns to make further distinction of file types. It derives discriminant function for each file type in each cluster. For instance, if two clusters have mp3 files, LDA derives two linear discriminant functions (i.e., one for each cluster). The output of linear discriminant function is called discriminant score, which is supposed to be the least for actual file type.

6.2 Detection Algorithm

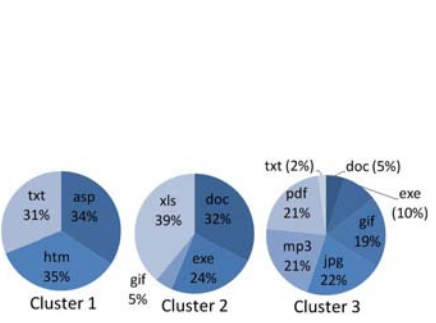
Figure 5 illustrates the detection process of our proposed scheme. When a file enters into the proposed system for type identification, its relative byte frequencies is calculated. The file is then assigned to the cluster having the most similar byte-pattern frequencies. If the assigned cluster represents only one file type, the same type is assigned to the file. Otherwise LDA is used to identify the exact type of the file. The discriminant scores are computed for all file types in the cluster using their respective discriminant function. The file type having least discriminant score is deemed to be the type of file.

7 Evaluation of Our Scheme

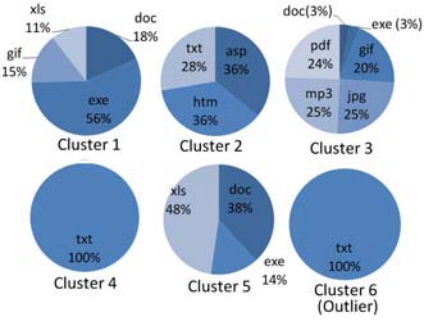
7.1 Cluster Analysis to Group Similar Byte-Frequency Distribution Files

In this section, we describe how files are grouped using clustering. We use Ward's clustering method [24] which forms clusters by minimizing the total within-cluster sum of squares. For instance, if XY is the cluster obtained by combining clusters X and Y, the sum of within-cluster distances (W_{XY}) are

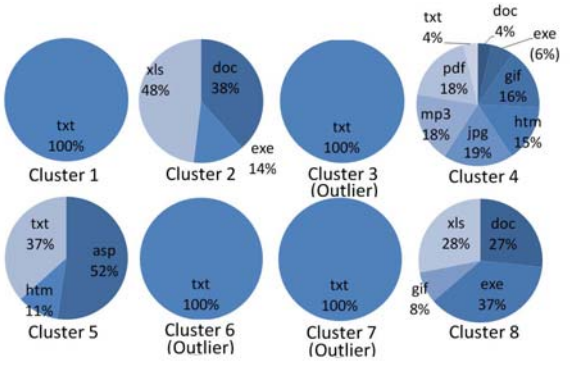
$$W_{XY} = \sum_{i=1}^{n_{XY}} (y_i - \bar{y}_{XY})' (y_i - \bar{y}_{XY}) \tag{3}$$



(a) Grouping file types using 3 clusters



(b) Grouping file types using 6 clusters



(c) Grouping file types using 8 clusters

Fig. 6. Grouping of file types on different number of clusters

where y_i are data points, $\bar{y}_{XY} = \frac{(n_X \bar{y}_X + n_Y \bar{y}_Y)}{(n_X + n_Y)}$ and , n_X , n_Y and $n_{XY} = n_X + n_Y$ are number of data points in X,Y and XY respectively.

Figure 6 illustrates the grouping of different file types and their respective percentage. We group files in 3, 6 and 8 clusters. Figure 6 shows that certain file types are always grouped together. These file types usually have similar content characteristics. For example, ASP, HTML, and TXT files usually contain ASCII characters therefore they are grouped together. Also multiple file types lie in one cluster and same file type in multiple clusters. Such observation supports our assumptions made in section 3, which states that files with different types could have similar byte-frequency patterns and files with the same type could have dissimilar patterns.

We observed some outlier clusters and did not use them in type identification process. We consider them as outlier because each of them only consists of one TXT type file.

7.2 Linear Discriminant Analysis to Classify Each File Type

This section discusses the classification accuracy. In detection phase, the test dataset is used to find out the detection accuracy of the clustering, LDA and overall system. If the type of a test file does not match with any of the cluster file type, it is said to be misclassified. For example, if file f of type X is assigned to cluster Y but cluster Y does not have type X, then file f is considered misclassified. Hence the accuracy (%) of assigning files to their respective clusters is calculated as follows.

$$Accuracy(\%) = \frac{(Total\ number\ of\ files\ -\ misclassified\ files)}{total\ number\ of\ files} \tag{4}$$

The files that are not misclassified in the clustering step are further processed by LDA if the assigned cluster has multiple file type. Each cluster has its own linear discriminant functions for its file types. For example, if the file f is assigned

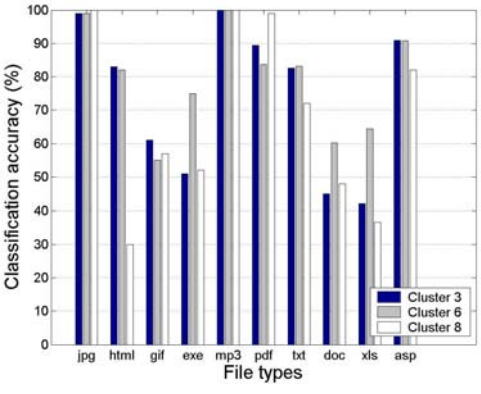


Fig. 7. Accuracy achieved by LDA while detecting 10 file types in different number clusters

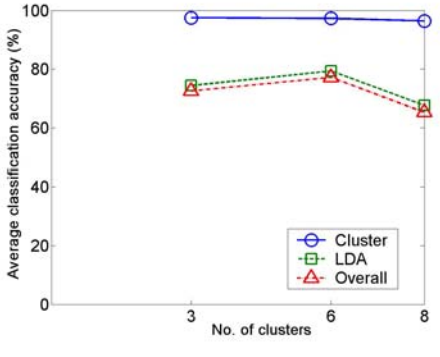


Fig. 8. Average accuracy (%) achieved by clustering and LDA when different number of clusters are used. Overall shows the combined accuracy of clusters and LDA.

to a cluster, linear discriminant score is computed for each file type using its respective function and the file type having least discriminant score is considered as the type of test file f. We use the Eq. 4 to compute the accuracy of LDA for each file type. Figure 7 shows that file types (such as MP3 and JPG) that reside only in one cluster have almost achieved 100% accuracy. However types EXE, DOC, XLS, and GIF reside in multiple clusters (because their byte-pattern frequencies are quite similar) and the results of LDA on them were less accurate. Figure 8 shows the average detection accuracy of the clustering, LDA, and the overall proposed system. Figure 8 also shows the average accuracy of Ward's clustering method when different number of clusters is used. It shows that the accuracy remains almost constant when we increase the number of clusters.

8 Comparison of Single- and Multi-centroid Models with the Proposed Scheme

Figure 9 shows the comparison of the proposed scheme with single- and multi-centroid models (using both cosine similarity and Mahalanobis distance). We

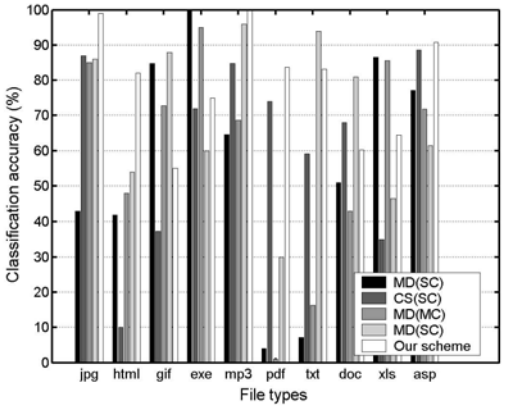


Fig. 9. Classification accuracy (%) achieved by our proposed scheme (6 clusters are used) and single and multiple centroid models using cosine similarity (CS) and Mahalanobis distance (MD) while classifying 10 sampled file types

	Single centroid		Multi centroid		Our proposed scheme (6 clusters)
	MD	CS	MD	CS	
Average accuracy	56.03%	61.59%	58.72%	69.66%	77.20%
Improvement by proposed scheme	37.78%	25.35%	31.47%	10.82%	

Fig. 10. Comparison of average classification accuracy among single- and multi-centroid models and the proposed scheme

took the classification accuracy using all the 256 byte-patterns in single and multi-centroid models (i.e. we did not truncate the models). Figure 9 shows that in most cases our recursive scheme has highest accuracy rate, and on average it achieved substantial improvement in classification accuracy over single- and multi-centroid models (Figure 10).

9 Conclusion and Future Work

This paper presented two approaches to improve the classification accuracy in identifying file types using their byte-frequency distributions. The first approach is to use cosine similarity as a distance metric to find out the consistency between the byte frequency-distribution of a test file and representative model of a file type. Based on our experimental results, we conclude that cosine similarity is a better measure than Mahalanobis distance as it improved 4.8% and 13.10% classification accuracy using Wei-Hen Li's single and multi centroid models respectively, and did not lose accuracy when using a small percentage of highly-occurred byte-patterns in the representative models. Using small percentage of byte patterns results in smaller model size and faster detection rate. Our experiment shows that cosine similarity reduces the detection speed by approximately 11% when only 10% of highly-occurred byte-patterns of the representative model are used. We expect that the improvement in model size and computation time will be much greater if we use higher n-gram.

Wei-Hen Li's multi-centroid model performs better than its single-centroid model but still suffers when building models of different file types showing similar byte patterns. We proposed a recursive scheme which is comparable to Wei-Hen Li's multi-centroid model but is more accurate (figure 10 shows 31.47% improvement).

From cluster analysis, we noticed that files of similar nature file types are always grouped together. For instance, the text-based files (ASP, TXT and HTML) are always grouped together. Also DOC and XLS which contains both text and binary characters are always found in same clusters. The accuracy of linear discriminant in our experiment is still not satisfactory. As our future work, we plan to improve our grouping scheme, especially focusing on identifying text-based file types.

Acknowledgement

This research is supported by the Ubiquitous Computing and Network(UCN) Project, Knowledge and Economy Frontier R&D Program of the Ministry of Knowledge Economy(MKE) in Korea and a result of subproject UCN 09C1-C5-20S.

H.Shin would like to gratefully acknowledge support from Post Brain Korea 21 and the research grant from Korean Government (MOEHRD, Basic Research Promotion Fund, KRF-2008-531-D00032).

References

1. Exclusion option to skip the files for the scanning in Norton antivirus, http://service1.symantec.com/SUPPORT/nav.nsf/0/c829006aa01d540b852565a6007770d8?OpenDocument
2. Stegdetect, http://packages.debian.org/unstable/utils/stegdetect
3. Libmagic1 package, http://packages.debian.org/unstable/libs/libmagic1
4. Wang, K., Stolfo, S.J.: Anomalous Payload-based Network Intrusion Detection. In: Jonsson, E., Valdes, A., Almgren, M. (eds.) RAID 2004. LNCS, vol. 3224, pp. 203–222. Springer, Heidelberg (2004)
5. Ahmed, I., Lhee, K.-s.: Detection of malcodes by packet classification. In: Workshop on Privacy and Security by means of Artificial Intelligence, ARES 2008, pp. 1028–1035 (2008)
6. Li, W.J., Wang, K., Stolfo, S., Herzog, B.: Fileprints: Identifying File Types by n-gram Analysis. In: Workshop on Information Assurance and security (IAW 2005), United States Military Academy, West Point, NY, pp. 64–71 (2005)
7. Srinivasan, N., Vaidehil, V.: Reduction of False Alarm Rate in Detecting Network Anomaly using Mahalanobis Distance and Similarity Measure. In: Proceedings of ICSCN, pp. 366–371 (2007)
8. Tan, P.-N., Steinbach, M., Kumar, V.: Introduction to data mining. Addison-Wesley, Reading (2005)
9. Martin, K., Nahid, S.: Oscar - file type identification of binary data in disk clusters and RAM pages. In: IFIP security and privacy in dynamic environments, pp. 413–424 (2006)
10. Martin, K., Nahid, S.: File type identification of data fragments by their binary structure. In: Proceedings of the IEEE workshop on information assurance, pp. 140–147 (2006)
11. Veenman, C.J.: Statistical disk cluster classification for file carving. In: IEEE third international symposium on information assurance and security, pp. 393–398 (2007)
12. Rencher, A.C.: Methods of Multivariate Analysis. Wiley Interscience, Hoboken (2002)
13. File extensions, http://www.file-extension.com/
14. Magic numbers, http://qdn.qnx.com/support/docs/qnx4/utils/m/magic.html
15. Nachenberg, C.: Polymorphic virus detection module, United States Patent # 5,826,013 (1998)
16. Szor, P., Ferrie, P.: Hunting for metamorphic. In: Proceedings of Virus Bulletin Conference, pp. 123–144 (2001)
17. RIX, Writing IA32 Alphanumeric Shell codes, http://www.phrack.org/issues.html?issue=57&id=15#article
18. Eller, R.: Bypassing MSB Data Filters for Buffer Overflow Exploits on Intel platforms (2003), http://community.core-di.com/~juliano/bypassmsb.txt
19. McDaniel, M., Hossain Heydari, M.: Content Based File Type Detection Algorithms. In: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (2003)
20. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems of Information Transmission, 1–11 (1965)
21. Calhoun, W.C., Coles, D.: Predicting the types of file fragments. Digital Investigation 5(1), 14–20 (2008)

22. Wang, K., Parekh, J.J., Stolfo, S.J.: Anagram: A Content Anomaly Detector Resistant to Mimicry Attack. In: Zamboni, D., Krügel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 226–248. Springer, Heidelberg (2006)
23. Gu, G., Porras, P., Yegneswaran, V., Fong, M., Lee, W.: BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation: in 16th USENIX Security Symposium (2007)
24. Ward, J.H.: Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 235–244 (1963)

Attacking 9 and 10 Rounds of AES-256

Ewan Fleischmann, Michael Gorski, and Stefan Lucks

Bauhaus-University Weimar, Germany
{Ewan.Fleischmann,Michael.Gorski,Stefan.Lucks}@uni-weimar.de

Abstract. The AES-256 has received less attention in cryptanalysis than the 192 or 128-bit versions of the AES. In this paper we propose new attacks on 9 and 10-round AES-256. In particular we present a 9-round attack on AES-256 which has the lowest data complexity of all known 9-round attacks. Also, our 10-round attack has a lower data complexity than all known attacks on AES-256. Also, our attack is the first that uses a key differential with probability below one in combination with a related-key boomerang attack. This leads to better related-key differentials which contain less non-zero byte differences and rounds with zero byte differences in each byte of a subkey difference.

Keywords: block ciphers, AES, differential cryptanalysis, related-key boomerang attack.

1 Introduction

The Advanced Encryption Standard (AES) [5] has become one of the most used symmetric encryption algorithms in the world. Differential cryptanalysis [3] is is the source for essentially all attacks on block ciphers like the AES (i.e. Rijndael). It recovers subkey bits for the first or the last rounds, while using differential properties of the underlying cipher.

The boomerang attack [15] is a strong extension to differential cryptanalysis in order to break more rounds than differential attacks can do, since the cipher is treated as a cascade of two sub-ciphers, using short differentials in each sub-cipher. These differentials are combined in an adaptive chosen plaintext and ciphertext attack to exploit properties of the cipher that have a high probability.

Related-key attacks [1, 14] apply differential cryptanalysis to ciphers using different, but related, keys and consider the information that can be extracted from encryptions under these keys. Ciphers with a weak key schedule are vulnerable to this kind of attack. The idea of related-key differentials was presented in [11], while two encryptions under two related-keys are used. Several combinations of related-key and differential attacks were introduced in the following. Related keys combined with the impossible differential attack [10], the differential-linear attack [8] or the rectangle attack [2, 9, 12, 13]. Biryukov [4] proposed a boomerang attack on the AES-128 which can break up to 5 and 6 out of 10 rounds. The related-key boomerang attack was published first in [2], but was not used to attack the AES.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 60–72, 2009.

Table 1. Existing attacks on round reduced AES-256

Attack	# Rounds	# Keys	Data / Time	Source
Partial Sums	8	1	$2^{128} - 2^{119}$ / 2^{204}	[6]
Partial Sums	9	256	2^{85} / $5 \cdot 2^{224}$	[6]
Related-Key Rectangle	9	4	2^{99} / 2^{120}	[12]
Related-Key Boomerang	9	$2^{15.5}$	2^{67} / $2^{142.83}$	Section 4
Related-Key Rectangle	10	256	$2^{114.9}$ / $2^{171.8}$	[2]
Related-Key Rectangle†	10	64	$2^{113.9}$ / $2^{172.8}$	[12]
Related-Key Boomerang	10	$2^{15.5}$	2^{67} / $2^{182.67}$	Section 5

† : the attack is based on the 10-round attack of Biham et al. [2], which has some flaws.

In this paper we propose a 9-round related-key boomerang attack on the AES-256 which has the lowest data complexity among all attacks presented so far. Furthermore we present the first attack on 10 rounds of the AES-256 described in full detail. There is a similar attack on 7 and 9-round AES-192 proposed by Gorski and Lucks at INDOCRYPT'08 [7]. In addition to their attack, we introduce a key differential that has a probability below one. This allows us to reduce data and time complexity. Up to now, this is the first paper which used this technique on the AES. Table 1 summarizes existing attacks on the AES-256 and our new attacks on 9 and 10 rounds.

The paper is organized as follows: In Section 2 we give a brief description of the AES. In Section 3 we describe the related-key boomerang attack. In Section 4, we present a related-key boomerang attack on 9-round AES-256. We extend this attack to a 10-round attack which is presented in Section 5. We conclude the paper in Section 6.

2 Description of the AES

The AES [5] is a block cipher using data blocks of 128 bits with 128, 192 or 256-bit cipher key. A different number of rounds is used depending on the length of the cipher key. The AES has 10, 12 and 14 rounds when a 128, 192 or 256-bit cipher key is used respectively. The plaintexts are treated as a 4 x 4 byte matrix, which is called *state*. Any single round applies four operations to the state:

- SubBytes (SB) is a non-linear byte-wise substitution applied to every byte of the state matrix.
- ShiftRows (SR) is a cyclic left shift of the i-th row by i bytes, where $i \in \{0, 1, 2, 3\}$.
- MixColumns (MC) is a multiplication of each column by a constant 4 x 4 matrix.
- AddRoundKey (AK) is a XORing of the state and a 128-bit subkey which is derived from the cipher key.

An AES round function applies the SB, SR, MC and AK operation in order. Before the first round, a whitening AK operation is applied and the MC operation is omitted in the last round because of symmetry. We concentrate on the 256-bit version of the AES in this paper and refer to [5] for more details on the other versions. Let W_i be a 32-bit word and $W_{i,j}$ the i-th byte in W_j, then the 256-bit cipher key is represented by $W_0||W_1||W_2||\ldots||W_7$. The 256-bit key schedule algorithm works as follows:

- For $j = 8$ to 59
  - If $j \equiv 0 \mod 8$, then
    - $W_{0,j} = W_{0,j-8} \oplus SB(W_{1,j-1}) \oplus Rcon(j/8)$,
    - For $i = 1$ to 3
      - $W_{i,j} = W_{i,j-8} \oplus SB(W_{i+1 \bmod 4,j-1})$,
  - Else if $j \equiv 4 \mod 8$, then
    - For $i = 0$ to 3 do $W_{i,j} = W_{i,j-8} \oplus SB(W_{i,j-1})$,
  - Else
    - For $i = 0$ to 3 do $W_{i,j} = W_{i,j-8} \oplus W_{i,j-1}$,

where *Rcon* denotes fixed constants depending on its input. The whitening key is $W_0||W_1||W_2||W_3$, the subkey of round 1 is $W_4||W_5||W_6||W_7$, the subkey of round 2 is $W_8||W_9||W_{10}||W_{11}$ and so on. The byte coordinates of a 4 x 4 state matrix are labeled as:

x_0	x_4	x_8	x_{12}
x_1	x_5	x_9	x_{13}
x_2	x_6	x_{10}	x_{14}
x_3	x_7	x_{11}	x_{15}

3 The Related-Key Boomerang Attack

We now describe the related-key boomerang attack [2] in more detail. But first, we have to give some definitions.

Definition 1. *Let P, P' be two bit strings of the same length. The bit-wise xor of P and P', $P \oplus P'$, is called the* difference *of P, P'. Let $'a'$ be a known and $'*'$ an unknown non-zero byte difference.*

Definition 2. *$\alpha \to \beta$ is called a* differential *if α is the plaintext difference $P \oplus P'$ before some non-linear operation $f(\cdot)$ and β is the difference after applying these operation, i.e, $f(P) \oplus f(P')$. The probability p is linked on a differential saying that an α difference turns into a β difference with probability p. The backward direction, i.e., $\alpha \leftarrow \beta$ has probability $\hat{p}$.*

Two texts (P, P') are called a *pair*, while two pairs (P, P', O, O') are called a *quartet*. Regularly, the differential probability decreases the more rounds are included. Therefore two short differentials covering only a few rounds each will be used instead of a long one covering the whole cipher. Related-keys are used to exploit some weaknesses of the key schedule to enhance the probability of the differentials being used. We call such differentials related-key differentials. We

split the related-key boomerang attack into two steps. The *related-key boomerang distinguisher step* and the *key recovery step*. The related-key boomerang distinguisher is used to find all plaintexts sharing a desired difference that depends on the choice of the related-key differential. These plaintexts are used in the key recovery step afterwards to recover subkey bits for the initial round key.

Distinguisher Step. During the *distinguisher step* we treat the cipher as a cascade of two sub-ciphers $E_K(P) = E_K^1(P) \circ E_K^0(P)$, where K is the key used for encryption and decryption. We assume that the related-key differential $\alpha \rightarrow \beta$ for E^0 occurs with probability p, while the related-key differential $\gamma \rightarrow \delta$ for E^1 occurs with probability q, where α, β, γ and δ are differences of texts. The backward direction $E^{0^{-1}}$ and $E^{1^{-1}}$ of the related-key differential for E^0 and E^1 are denoted by $\alpha \leftarrow \beta$ and $\gamma \leftarrow \delta$ and occur with probability $\hat{p}$ and $\hat{q}$ respectively. The related-key boomerang distinguisher involves four different unknown but related-keys $K_a, K_b = K_a \oplus \Delta K^*, K_c = K_a \oplus \Delta K'$ and $K_d = K_a \oplus \Delta K^* \oplus \Delta K'$, where ΔK^* and $\Delta K'$ are chosen cipher key differences. The attack works as follows:

1. Choose a pool of s plaintexts P_i, $i \in \{1, \ldots, s\}$ uniformly at random and compute a pool $P_i' = P_i \oplus \alpha$.
2. Ask for the encryption of P_i under K_a, i.e., $C_i = E_{K_a}(P_i)$ and ask for the encryption of P_i' under K_b, i.e., $C' = E_{K_b}(P_i')$.
3. Compute the new ciphertexts $D_i = C_i \oplus \delta$ and $D_i' = C_i' \oplus \delta$.
4. Ask for the decryption of D_i under K_c, i.e., $O_i = E_{K_c}^{-1}(D_i)$ and ask for the decryption of D_i' under K_d, i.e., $O_i' = E_{K_d}^{-1}(D_i')$.

– For each pair (O_i, O_j'), $i, j \in \{1, \ldots, s\}$

5. If $O_i \oplus O_j'$ equals α store the quartet (P_i, P_j', O_i, O_j') in the set M.

A pair (P_i, P_j'), $i, j \in \{1, \ldots, s\}$ with the difference α satisfies the differential $\alpha \rightarrow \beta$ with the probability p. The output of E_0 is A_i and A_j', i.e., $E_{K_a}^0(P_i) = A_i$ and $E_{K_b}^0(P_j') = A_j'$ have a certain difference $\beta = A_i \oplus A_j'$ with probability p. Using the ciphertexts C_i and C_j' we can compute the new ciphertexts $D_i = C_i \oplus \delta$ and $D_j' = C_j' \oplus \delta$. Let $B_i = E_{K_c}^{1^{-1}}(D_i)$ and $B_j' = E_{K_d}^{1^{-1}}(D_j')$ are the decryption of D_i and D_j' with $E_{K_i}^{1^{-1}}$ $i \in \{c, d\}$. A difference δ turns into a difference γ after passing $E_{K_i}^{1^{-1}}$ with probability $\hat{q}$. Since $\delta = C_i \oplus D_i$ and $\delta = C_j' \oplus D_j'$ we know that $\gamma = A_i \oplus B_i$ and $\gamma = A_j' \oplus B_j'$ with probability $\hat{q}^2$. Since we also know, that $A_i \oplus A_j' = \beta$ with probability p, it follows that $(A_i \oplus B_i) \oplus (A_i \oplus A_j') \oplus (A_j' \oplus B_j') = \gamma \oplus \beta \oplus \gamma = \beta = (B_i \oplus B_j')$ holds with probability $p \cdot \hat{q}^2$. A β difference turns into an α difference after passing the differential $E_{K_i}^{0^{-1}}$ with probability $\hat{p}$. Thus, a pair of plaintexts (P_i, P_j') with $P_i \oplus P_j' = \alpha$ generates a new pair of plaintexts (O_i, O_j') where $O_i \oplus O_j' = \alpha$ with probability $p \cdot \hat{p} \cdot \hat{q}^2$. A quartet containing these two pairs is defined as:

Definition 3. *A quartet* (P_i, P_j', O_i, O_j') *which satisfies*

$$P_i \oplus P_j' = \alpha = O_i \oplus O_j',$$
$$A_i \oplus A_j' = \beta = B_i \oplus B_j',$$

$$A_i \oplus B_i = \gamma = A'_j \oplus B'_j,$$
$$C_i \oplus D_i = \delta = C'_j \oplus D'_j,$$

is called a ***correct related-key boomerang quartet*** *which occurs with probability* $Pr_c = p \cdot \hat{p} \cdot \hat{q}^2$. *A quartet* (P_i, P'_j, O_i, O'_j) *which only satisfies the condition* $P \oplus P'_j = \alpha = O_i \oplus O'_j$ *is called a* ***false related-key boomerang quartet****.*

Figure 1 displays the structure of the related-key boomerang distinguisher step. Any attacker who applies a related-key boomerang distinguisher does not know the internal states A_i, A'_j, B_i, B'_j, since he can only apply a chosen plaintext and ciphertext attack on the cipher. The set M, which is the output of the related-key boomerang distinguisher, therefore contains correct and false related-key boomerang quartets. It is impossible to form another distinguisher which separates the correct and the false related-key boomerang quartets, since the interior differences β and γ cannot be computed.

Key Recovery Step. The second step of the related-key boomerang attack is the *key recovery step*. From now on, an attacker operates on the set M that was stored by the related-key boomerang distinguisher. Let k_a, k_b, k_c and k_d be some key bits of the last round keys derived from the cipher keys K_a, K_b, K_c and K_d. Let $d_k(C)$ be the one round partially decryption of C under the key bits k. The key bits are related as $k_b = k_a \oplus \Delta k^*$, $k_c = k_a \oplus \Delta k'$ and $k_d = k_a \oplus \Delta k^* \oplus \Delta k'$, where Δk^* and $\Delta k'$ are differences of the last round key bits. These differences are derived from the cipher key difference ΔK^* and $\Delta K'$. The key recovery step works as follows:

$$K_b = K_a \oplus \Delta K^*$$
$$K_c = K_a \oplus \Delta K'$$
$$K_d = K_a \oplus \Delta K^* \oplus \Delta K'$$

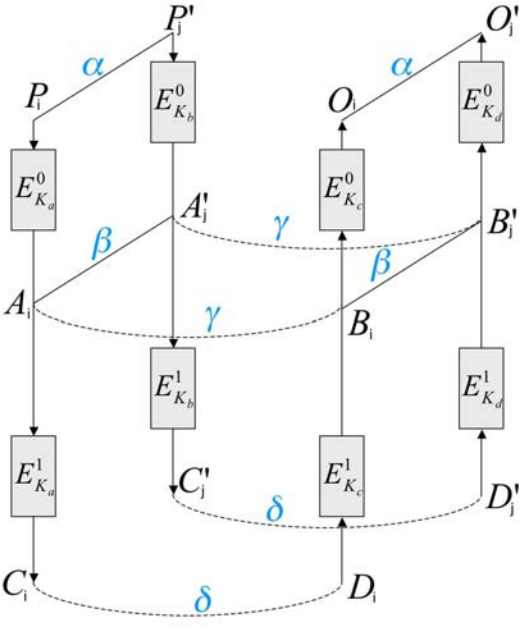


Fig. 1. The related-key boomerang distinguisher

- For each key-bit combination of k_a
  1. Initialize a counter for each key-bit combination with zero.
  - For all quartets (P, P', O, O') stored in M
    2. Ask for the encryption of P, P', O, O' under K_a, K_b, K_c and K_d respectively and obtain the ciphertext quartet C, C', D, D'. Decrypt the ciphertexts C, C', D, D' under k_a, k_b, k_c, k_d, i.e., $\bar{C} = d_{k_a}(C), \bar{C'} = d_{k_b}(C'), \bar{D} = d_{k_c}(D)$ and $\bar{D'} = d_{k_d}(D')$.
    3. Test whether the differences $\bar{C} \oplus \bar{D}$ and $\bar{C'} \oplus \bar{D'}$ have a desired difference an attacker would expect depending on the related-key differential being used. Increase a counter for the used key-bits if the difference is fulfilled in both pairs.
4. Output the key-bits k_a with the highest counter as the correct one.

Four cases can be distinct in Step 3, since M contains correct and false related-key boomerang quartets and the key-bit combination k_a can either be correct or false. A correct related-key boomerang quartet encrypted with the correct key bits will have the desired difference needed to pass the test in Step 3 with probability 1. Hence, the counter for the correct key bits is increased. The three other cases are: a correct related-key boomerang quartet is used with false key bits (Pr_{cK_f}), a false related-key boomerang quartet is used with the correct key-bits (Pr_{fK_c}) or a false related-key boomerang quartet is used with a false key-bit combination (Pr_{fK_f}). We assume that the cipher acts like a random permutation. In these cases we assume that

$$Pr_{cK_f} = Pr_{fK_c} = Pr_{fK_f} =: Pr_{filter}.$$

The probability that a quartet in one of the three undesirable cases is counted for a certain key bit combination is Pr_{filter}. The related-key differentials have to be chosen such that the counter of the correct key bits is significantly higher than the counter of each false key bit combination. If the differentials have a high probability the key recovery step outputs the correct key-bits in Step 4 with a high probability much faster than exhaustive search.

4 Related-Key Boomerang Attack on 9-Round AES-256

In this section we mount a key recovery attack on 9-round AES-256 using $2^{15.5}$ related keys. The cipher is represented as $E = E^1 \circ E^0$. E^0 is a differential containing rounds 1 to 5 and including the whitening key addition as well as the key addition of round 5. E^1 is a differential covering rounds 6 to 9. The notation used in our attack will be defined as:

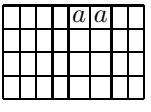
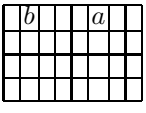
- K_a, K_b, K_c, K_d unknown cipher keys (256 bit).
- $K_{ai}, K_{bi}, K_{ci}, K_{di}$ unknown round keys of round i, where $i \in \{0, 1, 2, \ldots, 12\}$ (128 bit).
- $\Delta K^*, \Delta K'$ chosen cipher key differences (256 bit).
- $\Delta K_i^*, \Delta K_i'$ known subkey differences of round i (128 bit).
- P_i, P_j', O_i, O_j' plaintexts.

- C_i, C'_j, D_i, D'_j ciphertexts.
- a is a known non-zero byte difference.
- b, b' are an output differences of S-Box for the input difference a.
- c, d, e, f are unknown non-zero byte differences.
- $*$ is a variable unknown non-zero byte differences.

The Structure of the Keys. In our attack we use four related but unknown keys K_a, K_b, K_c and K_d. Let K_a be the unknown key an attacker would like to recover. The relation that is required for the attack is:

$$K_b = K_a \oplus \Delta K^*$$
$$K_c = K_a \oplus \Delta K'$$
$$K_d = K_a \oplus \Delta K^* \oplus \Delta K'$$

ΔK^* is the cipher key difference used for the first related-key differential E^0 and $\Delta K'$ is the cipher key difference used for the second related-key differential E^1. An attacker only chooses the differences ΔK^* and $\Delta K'$ but does not know the keys. He chooses the cipher key differences as:

$\Delta K^* =$  and $\Delta K' =$ 

Using the key schedule algorithm of AES-256 we can use the cipher key differences ΔK^* and $\Delta K'$ to derive the round key differences $\Delta K^*_0, \ldots, \Delta K^*_8$ and $\Delta K'_0, \ldots, \Delta K'_8$ respectively.[1] These values are shown in Figure 2 and 3. The key differences $\Delta K^*_0, \Delta K^*_1, \ldots, \Delta K^*_{10}$ occur with probability 1, while the key differences $\Delta K'_0, \Delta K'_1, \ldots, \Delta K'_{10}$ occur with probability 2^{-14}. This is the probability that an a difference will be transformed into a certain b difference by the S-Box two times. The difference b can be one of $2^7 - 1$ values, because of the symmetry of the XOR operation and the fact that an a difference can be one of $2^8 - 1$ differences.

4.1 The Related-Key Differential E^0 for Rounds $1 - 5$

The input difference α of E^0 has a non-zero difference in bytes 0,3,4,5,9,10,14 and 15. After SR_1 all non-zero byte differences are in column 0 and 1. A column with four non-zero byte differences ist transformed into a column having an a difference with fixed position after MixColumns with probability 2^{-32}. This occurs for two of such columns with probability 2^{-64}. The two a differences in bytes 0 and 4 are canceled out by the key addition AK_1. Thus each byte of the state matrix has a zero difference until AK_3 creates an a difference in byte 0, which is transformed to a non-zero difference by SB_4 and to four non-zero byte differences after MC_4. The state matrix has a non-zero differences in bytes 0, 1, 2, and 3. The difference which occurs after AK_5 is called β_{out}, where all bytes

[1] The key difference ΔK^* is also used in [12].

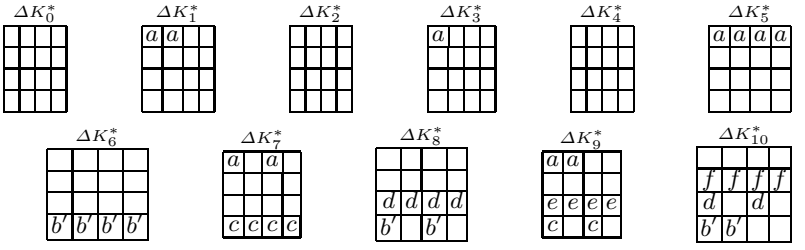


Fig. 2. Round key differences derived from ΔK^*

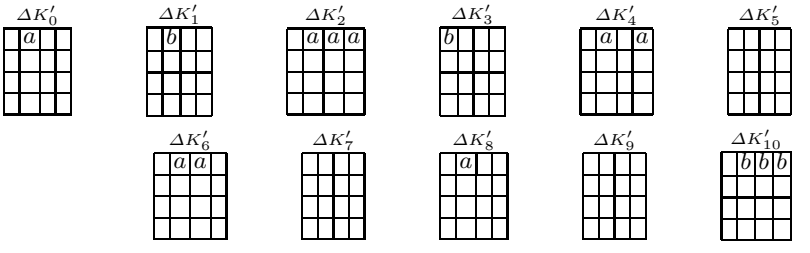


Fig. 3. Round key differences derived from $\Delta K'$

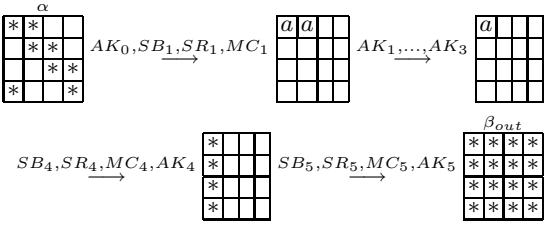


Fig. 4. The related-key differential E^0

are non-zero. The probability of the differential E_0, i.e., the transformation of an α difference into a β_{out} difference is given by

$$Pr(\alpha \rightarrow \beta_{out}) = 2^{-64}.$$

The related-key differential E^0 is shown in Figure 4.

4.2 The Related-Key Differential $E^{1^{-1}}$ for Rounds $9 - 6$

From the bottom up direction of the related-key boomerang distinguisher the related-key differential $E^{1^{-1}}$ is used with the round-key differences of $\Delta K'$. Note that the used key differential $\Delta K'$ has a probability of 2^{-14} to occur, which increases the necessary keys for the attack. The input difference δ consists of

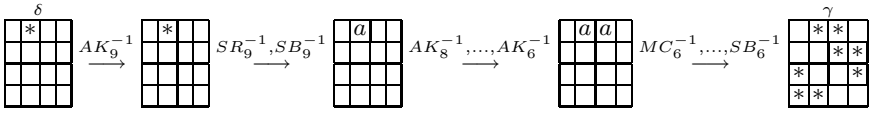


Fig. 5. The related-key differential $E^{1^{-1}}$

a non-zero difference in byte 4. SB_9^{-1} generates an a difference in byte 4 with probability 2^{-8}. If this occurs the text difference after SB_9^{-1} is equal to the subkey difference $\Delta K'_8$. Hence, all bytes have a zero difference after applying AK_8^{-1}. Passing AK_6^{-1} the state matrix has two a differences in bytes 4 and 8. We call γ the text difference remaining after SB_6^{-1}. This text difference has eight non-zero difference in bytes 2,3,4,7,8,9,13 and 14. The probability of $E^{1^{-1}}$ is $\Pr(\gamma \leftarrow \delta) = 2^{-8}$. The related-key differential $E^{1^{-1}}$ is shown in Figure 5.

4.3 The Related-Key Differential $E^{0^{-1}}$ for Rounds $6 - 1$

For the following steps we need that the output difference β_{out} of the related-key differential E^0 is equal to the input difference β_{in} for the related-key differential $E^{0^{-1}}$. Note that β_{in} and β_{out} are not only equal in the same positions of non-zero differences but are also equal in each byte. We will shown how to construct such a case. From the boomerang condition inside the cipher for two differences γ_1 and γ_2 we know that

$$\beta_{out} \oplus \gamma_1 \oplus \gamma_2 = \beta_{in}$$

holds with some probability. Since γ_1 and γ_2 are equal in each byte, we simply write γ. Thus the above condition reduces to :

$$\beta_{out} \oplus \gamma \oplus \gamma = \beta_{out} = \beta_{in} \tag{1}$$

Using the differentials above, the differences β_{in} and β_{out} are equal with probability one. Note that these difference occur only with some probability, which will be given in more detail later.

Let A, A', B, B' be the internal state after SR_5 when encrypting P, P', O, O' under K_a, K_b, K_c, K_d respectively. We use the same notation as in Figure 1. Since MC is linear γ can be expressed as

$$\gamma = K_{a5} \oplus MC_5(A) \oplus K_{c5} \oplus MC_5(B) = \overbrace{K_{a5} \oplus K_{c5}}^{\Delta K'_5} \oplus MC_5(A \oplus B) \tag{2}$$

and as

$$\gamma = K_{b5} \oplus MC_5(A') \oplus K_{d5} \oplus MC_5(B') = \overbrace{K_{b5} \oplus K_{d5}}^{\Delta K'_5} \oplus MC_5(A' \oplus B'). \tag{3}$$

Equation (2) and (3) can be combined, which leaves $A \oplus A' = B \oplus B'$. In other words, the MC_5 operation can be undone with the probability 1 due to the

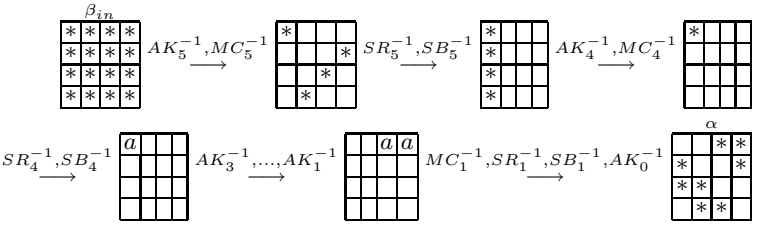


Fig. 6. The related-key differential $E^{0^{-1}}$

boomerang condition (1).To achieve that β_{out} equals β_{in} the differences γ_1 and γ_2 have to be equal. This happens with probability 2^{-56} since an a difference can be one of $2^7 - 1$ values after an S-Box transformation and MixColumns is a linear operation. If this occurs we know from the boomerang condition that $\beta_{out} \oplus \gamma_1 \oplus \gamma_2 = \beta_{out} = \beta_{in}$ holds with some probability and MC_5 can be undone with probability one. This means that we know that a non-zero byte difference occurs after MC_5^{-1} only in the bytes 0,7,10 and 13, while the other bytes are zero. Four non-zero bytes remain after AK_4^{-1} in column 0. With probability 2^{-24} MC_4^{-1} generates a non-zero difference in byte 0 while the remaining bytes are zero. After the next S-Box operation we have an a difference with probability 2^{-8}. The further steps operate such that the output difference of $E^{0^{-1}}$ has non-zero differences in bytes 1,2,6,7,8,11,12 and 13. We call this difference α. The differential $E^{0^{-1}}$ has the probability $\Pr(\alpha \leftarrow \beta_{in}) = 2^{-32}$ and is shown in Figure 6.

4.4 The Attack

1. Choose $2^{42.5}$ structures $S_1, S_2, \ldots, S_{2^{42.5}}$ of 2^{64} plaintexts P_i, $i \in \{1, 2, \ldots, 2^{64}\}$ where the bytes 0, 3, 4, 5, 9, 10, 14, 15 are fixed and the other bytes have all possible values. Ask for encryption of P_i under K_a to obtain the ciphertexts C_i, i.e., $C_i = E_{K_a}(P_i)$.
2. Compute $2^{42.5}$ structures $S'_1, S'_2, \ldots, S'_{2^{49.5}}$ of 2^{64} plaintexts $P'_i = P_i$. Ask for encryption of P'_i under K_b, where $K_b = K_a \oplus \Delta K^*$ to obtain the ciphertexts C'_i, i.e., $C'_i = E_{K_b}(P'_i)$.
3. For each possible value of b compute $\Delta\tilde{K}'$
   - 3.1. Compute $2^{42.5}$ structures $S^*_1, S^*_2, \ldots, S^*_{2^{42.5}}$ of 2^{64} ciphertexts D_i, i.e, $D_i = C_i \oplus \delta$ where δ is a fixed 128-bit value of which byte 4 non-zero while all other bytes are zero. Ask for decryption of D_i under $K_c = K_a \oplus \Delta\tilde{K}'$ to obtain the plaintexts O_i, i.e., $O_i = E^{-1}_{K_c}(D_i)$.
   - 3.2. Compute $2^{42.5}$ structures $S'^*_1, S'^*_2, \ldots, S'^*_{2^{42.5}}$ of 2^{64} ciphertexts D'_i, i.e, $D'_i = C'_i \oplus \delta$ where δ is as in Step 3.1. Ask for decryption of D'_i under $K_d = K_a \oplus \Delta K^* \oplus \Delta\tilde{K}'$ to obtain the plaintexts O'_i, i.e., $O'_i = E^{-1}_{K_d}(D_i)$.
   - 3.3. Store only those quartets (P_i, P'_j, O_i, O'_j) where $O_i \oplus O'_j$ have a zero byte differences in bytes 0, 3, 4, 5, 9, 10, 14 and 15.

3.4. Guess an 8-bit subkey $\bar{k}_{a9}$ of K_{a9} in the positions of byte 4 and compute $\bar{k}_{b9}, \bar{k}_{c9}, \bar{k}_{d9}$ respectively.
  3.4.1. Partially decrypt each quartet (C_i, C'_j, D_i, D'_j) remaining after Step 3.3 under $\bar{k}_{a9}, \bar{k}_{b9}, \bar{k}_{c9}, \bar{k}_{d9}$ respectively.
  3.4.2. Check if $d_{\bar{k}_{a9}}(C_i) \oplus d_{\bar{k}_{c9}}(D_i)$ and $d_{\bar{k}_{b9}}(C'_i) \oplus d_{\bar{k}_{d9}}(D'_i)$ have an a-difference after SB_9^{-1} in byte 4. Record $(\bar{k}_{a9})$ and all the qualified quartets and then go to Step 3.5.

3.5. Guess a 32-bit subkey k'_{a0} of K_{a0} in the positions of bytes 0,5,10,15 and compute $k'_{b0} = k'_{c0} = k'_{d0} = k'_{a0}$ *(ΔK_0^* and $\Delta K'_0$ are zero in these four bytes)*
  3.5.1. Partially encrypt each quartet (P_i, P'_j, O_i, O'_j) remaining after Step 3.4.2 under $k'_{a0}, k'_{b0}, k'_{c0}, k'_{d0}$ respectively.
  3.5.2. Check if $e_{k'_{a0}}(P_i) \oplus e_{k'_{b0}}(P'_j)$ and $e_{k'_{c0}}(O_i) \oplus e_{k'_{d0}}(O'_j)$ have an a difference in byte 0 after MC_1. Record $(\bar{k}_{a9}, k'_{a0})$ and all the qualified quartets and then go to Step 3.6.

3.6. Guess a 32-bit subkey k^*_{a0} of K_{a0} in the positions of bytes 3,4,9,14 and compute $k^*_{b0} = k^*_{a0}$, compute $k^*_{c0} = k^*_{a0} \oplus M_1$, with the 32-bit value $M_1 = (a, 0, 0, 0)$ and compute $k^*_{d0} = k^*_{b0} \oplus M_1$.
  3.6.1. Partially encrypt each quartet (P_i, P'_j, O_i, O'_j) remaining after Step 3.5.2 under $k^*_{a0}, k^*_{b0}, k^*_{c0}, k^*_{d0}$ respectively.
  3.6.2. Check if $e_{k^*_{a0}}(P_i) \oplus e_{k^*_{b0}}(P'_j)$ and $e_{k^*_{c0}}(O_i) \oplus e_{k^*_{d0}}(O'_j)$ have an a difference in byte 4 after MC_1. If there exist more than 2 boomerang quartets passing this test, record $(\bar{k}_{a9}, k'_{a0}, k^*_{a0})$ and all the qualified quartets and then go to Step 4. Otherwise, repeat Step 3.6 with another guessed key. If all the possible keys are tested, then repeat Step 3.5 with another guessed key. If all the possible keys are tested, then repeat Step 3.4 with another guessed key.

4. For a suggested $(\bar{k}_{a9}, k'_{a0}, k^*_{a0})$, do an exhaustive search for the remaining 184 cipher key bits using trial encryption. If a 256-bit cipher key is suggested, output it as the cipher key. Otherwise, go to Step 3 with another guess of b.

4.5 Analysis of the Attack

A pool of 2^{64} plaintexts can be combined to $\frac{\left(2^{64}\right)^2}{2} = 2^{127}$ quartets. Each quartet of structures S_i, S'_i, S^*_i, S'^*, $i \in \{1, 2, \ldots, 2^{42.5}\}$ can be analyzed separately. The data complexity of Step 1, 2, 3.1 and 3.2 is $2^2 \cdot 2^{64} = 2^{66}$ chosen plaintexts, while the time complexity is about 2^{64} encryptions for Step 1 and 2 and about $2^{7.5} \cdot 2^7 \cdot 2^{64} = 2^{78.5}$ for Step 3.1 and 3.2, since Step 3 runs at most 2^7 times and we expect that we need to run the attack with about $2^{7.5}$ different cipher keys. The data complexity of Step 3.3 is $2^2 \cdot 2^{64} = 2^{66}$ plaintexts, since we have a 64-bit filtering condition which leaves $2^{127} \cdot 2^{-64} = 2^{63}$ quartets stored in this step. Step 3.4.1 takes about $2^{7.5} \cdot (1/9) \cdot (1/16) \cdot 2^7 \cdot 2^8 \cdot 2^2 \cdot 2^{63} = 2^{80.33}$ nine round encryptions. The number of remaining quartets after Step 3.4.2 are $2^{63} \cdot 2^{-14} = 2^{49}$, since we have a 7-bit filtering on both pairs of a quartet. The time complexity of Step 3.5.1 is about $2^{7.5} \cdot (1/9) \cdot (4/16) \cdot 2^7 \cdot 2^{32} \cdot 2^8 \cdot 2^2 \cdot 2^{49} = 2^{100.33}$ nine round encryptions. Due to the 32-bit filtering on both pairs we obtain about

$2^{49} \cdot 2^{-64} = 2^{-15}$ quartets after Step 3.5.2. The time complexity of Step 3.6.1 is negligible, while about $2^{-15} \cdot 2^{-64} = 2^{-79}$ quartets remain after this step.

Using $2^{42.5}$ structures we obtain $\#PP \approx 2^{42.5} \cdot 2^{127} = 2^{169.5}$ quartets in total. A correct related-key boomerang quartet occurs with probability

$$\begin{aligned} Pr_c &= \Pr(\alpha \to \beta_{out}) \cdot (\Pr(\gamma \leftarrow \delta))^2 \cdot \Pr(\beta_{out} = \beta_{in}) \cdot \Pr(\alpha \leftarrow \beta_{in}) \\ &= 2^{-64} \cdot (2^{-8})^2 \cdot 2^{-56} \cdot 2^{-32} = 2^{-168}, \end{aligned}$$

since all related-key differential conditions are fulfilled. About $2^{42.5} \cdot 2^{-79} = 2^{-36.5}$ false related-key boomerang quartets remain Step 3.6.2 and are counted with the false key bits.

Using the Poisson distribution we can compute the success rate of our attack. The probability that the number of remaining quartets for each false key bit combination is larger than 1 is $Y \sim Poisson(\mu = 2^{-36.5})$, $\Pr(Y \geq 2) \approx 0$. Therefore the probability that our attack outputs false key bits as the correct one is very low. We expect to have a count of 3 quartets for the correct key bits. The probability that the number of quartets counted for the correct key bits is larger than 1 is $Z \sim Poisson(\mu = 3)$, $\Pr(Z \geq 2) \approx 0.8$.

The data complexity is about $2^{67} = 2^3 \cdot 2^{64}$ adaptive chosen plaintexts and ciphertexts and the time complexity is about $2^{142.83} = 2^{42.5} \cdot 2^{100.33}$ 9-round AES-256 encryptions.

5 Related-Key Boomerang Attack on 10-Round AES-256

The related-key boomerang attack can easily be extended to a 10-round attack, by guessing 32 bits of K_{10} at the bytes 1,4,11 and 14. Since we use related keys we have to take the key differences ΔK_{10}^* and $\Delta K_{10}'$ into account. Therefore we have to guess 8 bits for the unknown values of f in ΔK_{10}^*. The moreover, we change the order of MC_9 and AK_9, which allows us to mount the 9-round attack inside the 10-round attack. The data complexity of this 10-round attack on AES-256 remains at 2^{67} chosen plaintexts and ciphertexts and the time complexity increases to $2^{32} \cdot 2^8 \cdot (9/10) \cdot 2^{142.83} = 2^{182.67}$ 10-round AES-256 encryptions.

6 Conclusion

This paper proposes a new attack on 9 and 10-round AES-256. These are the first attacks using a related-key differential with probability below one. Our 9 and 10-round attacks on AES-256 have the lowest data complexity compared to existing attacks and use about 2^{67} chosen plaintexts and ciphertexts. Nevertheless, the attacks presented in this paper find only some weaknesses for round reduced versions of the AES-256, while the full version of the AES-256 remains unbroken.

Acknowledgements

The authors would like to thank Orr Dunkelman and the anonymous reviewers for many helpful comments.

References

[1] Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. J. Cryptology 7(4), 229–246 (1994)
[2] Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
[3] Biham, E., Shamir, A.: Differential Cryptanalysis of DES-like Cryptosystems. J. Cryptology 4(1), 3–72 (1991)
[4] Biryukov, A.: The Boomerang Attack on 5 and 6-Round Reduced AES. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) AES 2005. LNCS, vol. 3373, pp. 11–15. Springer, Heidelberg (2005)
[5] Daemen, J., Rijmen, V.: The Design of Rijndael: AES - The Advanced Encryption Standard. Springer, Heidelberg (2002)
[6] Ferguson, N., Kelsey, J., Lucks, S., Schneier, B., Stay, M., Wagner, D., Whiting, D.: Improved Cryptanalysis of Rijndael. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 213–230. Springer, Heidelberg (2001)
[7] Gorski, M., Lucks, S.: New Related-Key Boomerang Attacks on AES. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 266–278. Springer, Heidelberg (2008)
[8] Hawkes, P.: Differential-Linear Weak Key Classes of IDEA. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 112–126. Springer, Heidelberg (1998)
[9] Hong, S., Kim, J., Lee, S., Preneel, B.: Related-key rectangle attacks on reduced versions of SHACAL-1 and AES-192. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 368–383. Springer, Heidelberg (2005)
[10] Jakimoski, G., Desmedt, Y.: Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 208–221. Springer, Heidelberg (2004)
[11] Kelsey, J., Schneier, B., Wagner, D.: Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 233–246. Springer, Heidelberg (1997)
[12] Kim, J., Hong, S., Preneel, B.: Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 225–241. Springer, Heidelberg (2007)
[13] Kim, J., Kim, G., Hong, S., Lee, S., Hong, D.: The Related-Key Rectangle Attack - Application to SHACAL-1. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 123–136. Springer, Heidelberg (2004)
[14] Knudsen, L.R.: Cryptanalysis of LOKI91. In: Zheng, Y., Seberry, J. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 196–208. Springer, Heidelberg (1993)
[15] Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)

Cryptographic Properties and Application of a Generalized Unbalanced Feistel Network Structure

Jiali Choy, Guanhan Chew, Khoongming Khoo, and Huihui Yap

DSO National Laboratories
20 Science Park Drive, Singapore 118230
{cjiali,cguanhan,kkhoongm,yhuihui}@dso.org.sg

Abstract. In this paper, we study GF-NLFSR, a Generalized Unbalanced Feistel Network (GUFN) which can be considered as an extension of the outer function FO of the KASUMI block cipher. We prove upper bounds for the differential and linear hull probabilities for any $n+1$ rounds of an n-cell GF-NLFSR. Besides analyzing security against differential and linear cryptanalysis, we provide a frequency distribution for upper bounds on the true differential and linear hull probabilities. We also demonstrate a $(2n-1)$-round impossible differential distinguisher and a $(3n-1)$-round integral attack distinguisher on the n-cell GF-NLFSR. As an application, we design a new block cipher Four-Cell based on a 4-cell GF-NLFSR. We prove the security of Four-Cell against differential, linear, and boomerang attack. Based on the 7-round impossible differential and 11-round integral attack distinguisher, we set the number of rounds of Four-Cell to be 25 for protection against these attacks. Furthermore, Four-Cell can be shown to be secure against other attacks such as higher order differential attack, cube attack, interpolation attack, XSL attack and slide attack.

Keywords: Block Ciphers, Generalized Unbalanced Feistel Network, Differential Probability, Linear Hull Probability.

1 Introduction

In this paper, we examine a family of block ciphers whose structure is modelled after that of a Generalized Unbalanced Feistel Network (GUFN). The GUFN was first suggested by Schneier et al. in [20]. Similar to conventional Feistel networks, unbalanced ones comprise of a concatenation of rounds. In each round, one part of the block controls the encryption of another part of the block. However, the two parts need not be of equal sizes.

The particular GUFN we shall be analyzing is an n-cell extension of the outer function FO of the KASUMI block cipher [24], which is a 2-cell structure. Besides being a GUFN, our structure can also be viewed as an n-cell *NonLinear Feedback Shift Register* (NLFSR). Thus, we call our structure a *Generalized*

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 73–89, 2009.

Feistel-NonLinear Feedback Shift Register (GF-NLFSR). In Section 3, we shall give a detailed description of the GF-NLFSR.

Many GUFN-based block ciphers have been constructed; some examples include the ciphers SMS4 [13] and CLEFIA [21]. While the true differential and linear hull probabilities of these ciphers are not known in the open literature, they have been calculated for other GUFN-like constructions. In [24], these were derived for KASUMI's FO function, which is equivalent to a 2-cell GUFN. Similar analyses have been done in [25] for another GUFN-like round function, and also in [15]. To the best of our knowledge, bounds for the true differential and linear hull probabilities have not been proven for GUFN-based ciphers with n input cells. Analysis of true differentials and linear hulls is required in assessing vulnerability to attacks such as boomerang attack. In light of this, the study in our paper is both novel and useful.

In Sections 4 and 5, we prove that the true differential and linear hull probability of any $n+1$ rounds of the n-cell GF-NLFSR is bounded by p^2 where p is the maximal probability of the nonlinear function. In Section 6, we investigate the frequency distribution of the differential and linear hull probability of any $n+1$ rounds based on different input-output differentials/linear masks. From the frequency distribution, we see that the maximal probability p^2 only holds for a very tiny portion of all differentials/linear hulls. There are also other differentials/linear hulls having probability bounds $p^3, p^4, \ldots, p^n$, but we prove that almost all differentials/linear hulls have probability bound p^n. Furthermore, we compute the expected differential/linear hull probability bound and find this value to be close to $(2^{-B}+p)^n$ where B is the size of each cell in GF-NLFSR. These differential and linear hull probability bounds are achieved when the input differences and mask values are randomly chosen, which is likely when $n+1$ rounds of the n-cell GF-NLFSR is prepended and appended by additional cipher structures. In this case, the security of $n+1$ rounds of n-cell GF-NLFSR, in the sense of differential and linear hull probability bounds, is therefore much better than is typically believed. This motivates our study of the expected bounds in the Section 6.

Other than differential and linear cryptanalysis, in Sections 7 and 8, we also consider the security of GF-NLFSR against impossible differential and integral cryptanalysis. For the former this is done by finding impossible differential characteristics which play the role of a sieve, methodically rejecting the wrong key guesses and leaving the correct key. For GF-NLFSR, the maximum number of rounds for impossible differential characteristics was found to be $2n-1$. On the other hand, in an integral attack, the attacker looks at larger carefully chosen sets of encryptions, in which parts of the input text form a multiset. We studied the propagation of multisets through the cipher and unveiled a $(3n-1)$-round distinguisher for GF-NLSR.

As an application of our results on GF-NLFSR, we design a GUFN-based block cipher Four-Cell in Section 9. It is a 128-bit block cipher based on a 4-cell GF-NLFSR where each cell is 32-bit long. Besides proving practical security against differential and linear cryptanalysis, we are able to bound its true

differential probability by $2^{-55.39}$ and linear hull probability by $2^{-52.96}$. Moreover, we show that with 99.9999% frequency, the differential and linear hull probability bounds are much lower at $2^{-110.78}$ and $2^{-105.91}$ respectively. These facts also allow us to prove its security against boomerang attack. Based on the results in Sections 7 and 8, we can deduce a 7-round impossible differential and an 11-round integral attack distinguisher on Four-Cell. To protect against these attacks, we set the number of rounds of Four-Cell to be 25. Furthermore, we explain why Four-Cell is secure against other cryptanalysis like higher-order differential attack, cube attack, interpolation attack, XSL attack and slide attack.

Like the AES cipher, our Four-Cell block cipher can be proven secure against known block cipher attacks. In principle, it can use the same S-box (SubBytes) and MDS transform (MixColumn) as AES. However, it is more efficient (in hardware) in the sense that it uses less MDS transforms (25 compared to 40) than AES while keeping the number of S-boxes unchanged. Another advantage of the n-cell GF-NLFSR structure is that the nonlinear function in any n rounds can be computed in parallel. Therefore, any four rounds of the nonlinear transforms in our block cipher Four-Cell can be computed in parallel. This is not true for a general GUFN-based block cipher like SMS4 [13].

2 Definitions and Preliminaries

In this paper, we shall study the GF-NLFSR which can be considered as a particular instantiation of the Generalized Unbalanced Feistel Network defined in [20]. In what follows, the "+" symbol is used to denote finite field addition (XOR) over $GF(2)^n$ or ordinary addition, depending on the operands and context.

2.1 Differential Cryptanalysis

As is widely known, differential cryptanalysis [1] is a chosen-plaintext attack in which statistical key information is deduced from ciphertext blocks obtained by encrypting pairs of plaintext blocks with a specific bitwise difference under the target key. It studies the propagation of input differences to output differences in iterated transformations.

Let $f : GF(2)^m \mapsto GF(2)^m$ be a Boolean mapping composed of a number of rounds. The concept of *characteristic* was introduced: a sequence of difference patterns such that the output difference from one round corresponds to the input difference in the next round. On the other hand, in [10,11], the concept of a *differential*, denoted by $\alpha \xrightarrow{f} \beta$, was presented, where the XORs in the inputs and outputs of the intermediate rounds are not fixed. We denote $DP(\alpha \xrightarrow{f} \beta) = Pr(f(x) + f(x+\alpha) = \beta)$, where α, β are fixed input and output differences.

Differential cryptanalysis exploits differential characteristics with high probability. However, even if the maximal differential characteristic probability is low, one cannot conclude that the cipher is secure against differential attack. Instead, one must show that the maximal differential probability of all differentials is low

enough [11]. This property ensures *provable security* against differential cryptanalysis as opposed to *practical security* which simply considers the maximal differential characteristic probability.

Proposition 1. *[11] A block cipher with block length m is resistant against conventional differential attacks under an independent subkey assumption, if there does not exist any differential $\alpha \longrightarrow \beta$, $\alpha \neq 0$, ranging over all but a few rounds, such that $DP(\alpha \longrightarrow \beta) \gg 2^{-m}$.*

For key-dependent functions, we consider the average resistance against differential cryptanalysis, i.e. the average differential probability taken over the entire key set. More formally, let $F : GF(2)^m \times K \mapsto GF(2)^m$ be a key-dependent function. Denote $f_k = F(x, k)$ for each fixed $k \in K$. Let $\alpha, \beta \in GF(2)^m$ be constants. The *differential probability of the differential* $\alpha \xrightarrow{F} \beta$ is defined as $DP(\alpha \xrightarrow{F} \beta) = \frac{1}{|K|} \sum_{k \in K} DP(\alpha \xrightarrow{f_k} \beta)$. The *maximal differential probability* of F is defined as $DP(F_{max}) = \max_{\alpha \neq 0, \beta} DP(\alpha \xrightarrow{F} \beta)$.

2.2 Linear Cryptanalysis

Linear cryptanalysis [14] is a known-plaintext attack that tries to utilize high probability occurrences of linear expressions involving plaintext bits, ciphertext bits, and subkey bits.

As with the differential case, we must also distinguish between a linear characteristic and a linear hull. A *linear characteristic over f* consists of a sequence of mask values such that the output mask values from one round corresponds to the input mask values to the next round. On the other hand, a *linear hull*, denoted by $u \xleftarrow{f} w$, is the set of all linear characteristics with the same initial and terminal mask values. We denote $LP(u \xleftarrow{f} w) = [2 \cdot Pr(u \cdot f(x) = w \cdot x) - 1]^2$, where w, u are fixed input and output mask values.

Linear cryptanalysis takes advantage of linear characteristics with high correlation probability to recover key bits. However, in the evaluation of the strength of a block cipher against linear cryptanalysis, one must consider the linear hulls instead. Having low linear hull probability for all linear hulls will guarantee provable security against linear attacks [17].

Proposition 2. *[17] A block cipher with block length m is resistant against conventional linear cryptanalysis under an independent subkey assumption, if there does not exist any linear hull $u \longleftarrow w$, $u \neq 0$, ranging over all but a few rounds, such that $LP(u \longleftarrow w) \gg 2^{-m}$.*

For key-dependent functions, we consider the average resistance against linear cryptanalysis. Explicitly, let $F : GF(2)^m \times K \mapsto GF(2)^m$ be a key-dependent function. Denote $f_k(x) = F(x, k)$ for each fixed $k \in K$. Let $u, w \in GF(2)^m$ be constants. The *linear hull probability of the linear hull* $u \xleftarrow{F} w$ is defined as

$LP(u \xleftarrow{F} w) = \frac{1}{|K|}\sum_{k\in K} LP(u \xleftarrow{f_k} w)$. The *maximal linear hull probability* of F is defined as $LP(F_{max}) = \max_{w,u\neq 0} LP(u \xleftarrow{F} w)$.

It was proven in [11] and [17] the following result about differential and linear hull probabilities of compositions of key-dependent mappings.

Fact 1. *[11,17] Let $F : GF(2)^m \times GF(2)^m \times K_1$ and $G : GF(2)^m \times GF(2)^m \times K_2$ be key-dependent functions of the type $F(x, k, k') = f(x + k, k')$, $G(x, k, k') = g(x + k, k')$, where $f : GF(2)^m \times K_1 \mapsto GF(2)^m$ and $g : GF(2)^m \times K_2 \mapsto GF(2)^m$ are bijective for all fixed $k_1 \in K_1$, $k_2 \in K_2$. Then $DP(\alpha \xrightarrow{G\circ F} \beta) = \sum_{\xi\in GF(2)^m} DP(\alpha \xrightarrow{f} \xi)DP(\xi \xrightarrow{g} \beta)$ and $LP(u \xleftarrow{G\circ F} w) = \sum_{v\in GF(2)^m} LP(u \xleftarrow{g} v)LP(v \xleftarrow{f} w)$.*

In Sections 4 and 5, we shall be demonstrating provable security of our design structure against differential and linear cryptanalysis by studying its differential and linear hull probabilities. Fact 1 will be required in the proofs of our results later.

3 Description of the Structure

In this section, we will give a description of our design structure, which we call GF-NLFSR. It is essentially a generalization of the outer function, FO, of the KASUMI cipher. The FO function was first suggested by Matsui in [15, Figure 7] as one of the new structures of block ciphers with provable security against differential and linear cryptanalysis. It was then adopted in the design of KASUMI [24]. The following result was proven in the same paper regarding the maximal differential and linear hull probabilities of this function.

Fact 2. *[24, Theorem 2] Let F be the 3-round function shown in Figure 1 of [24] (i.e. a 2-cell GF-NLFSR) where each $F_i : GF(2)^B \times GF(2)^B \times K_i' \mapsto GF(2)^B$ is of the form $F_i(x, k_i, k_i') = f_i(x + k_i, k_i')$ and each $f_i : GF(2)^B \times K_i' \mapsto GF(2)^B$ is bijective for all fixed $k_i' \in K_i'$, where K_i' is the key space for k_i'.*

(1) If $DP((f_i)_{max}) \leq p$ for each i, then $DP(F_{max}) \leq p^2$.
(2) If $LP((f_i)_{max}) \leq q$ for each i, then $LP(F_{max}) \leq q^2$.

This function splits the input block into 2 sub-blocks of equal size. Our block cipher structure generalizes this by splitting the input block into n sub-blocks of equal size. Figure 1 below displays one round of GF-NLFSR. Explicitly, suppose we have a m-bit block cipher, i.e. the input and output blocks are both of size $m = nB$ bits. Let the internal state by denoted by $\mathcal{S} = (S_1, S_2, \ldots, S_n)$ where $S_i \in GF(2)^B$. Therefore the internal state consists of n sub-blocks of B bits each. The round keys of the cipher shall be denoted by k_i, k_i' $(i = 1, \ldots, n + 1)$. Each F_i function is of the form

$$F_i : GF(2)^B \times GF(2)^B \times K_i' \mapsto GF(2)^B$$
$$F_i(x, k_i, k_i') = f_i(x + k_i, k_i')$$

where each $f_i : GF(2)^B \times K_i' \mapsto GF(2)^B$ is bijective for all fixed $k_i' \in K_i'$.

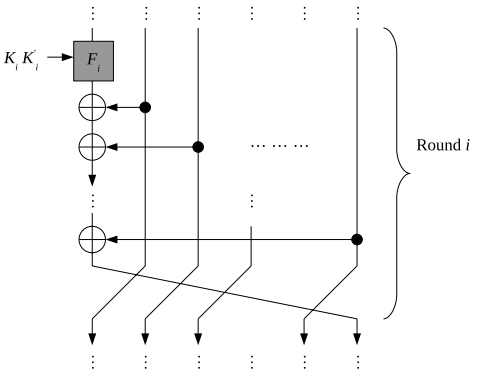


Fig. 1. One round of n-cell GF-NLFSR

The round function R that maps $\mathcal{S}_i$ to $\mathcal{S}_{i+1}$ under the round keys k_i, k_i' is:

$$R : GF(2)^m \times GF(2)^B \times K_i' \mapsto GF(2)^m$$
$$((S_1, S_2, \ldots, S_n), k_i, k_i') \mapsto (S_2, S_3, \ldots, S_n, F_i(S_1, k_i, k_i') + S_2 + S_3 + \ldots + S_n)$$

4 Differential Probability

In this section, we present a result for the differential probability of an n-block GF-NLFSR over $n+1$ rounds which is similar to Fact 2.

Theorem 1. *Let F be the $(n+1)$-round function in Figure 2 (left) of Appendix B where each $F_i : GF(2)^B \times GF(2)^B \times K_i' \mapsto GF(2)^B$ is of the form $F_i(x, k_i, k_i') = f_i(x + k_i, k_i')$ and each $f_i : GF(2)^B \times K_i' \mapsto GF(2)^B$ is bijective for all fixed $k_i' \in K_i'$. If $DP((f_i)_{max}) \leq p$ for each i, then $DP(F_{max}) \leq p^2$.*

Proof. Let the input difference of F be $\alpha = (\alpha_1, \ldots, \alpha_n) \neq 0$ and the output difference be $\beta = (\beta_1, \ldots, \beta_n) \neq 0$, where $\alpha_i, \beta_i \in GF(2)^B$ for $i = 1, 2, \ldots, n$. Also let the output difference of F_1 be ϵ.

In general, the input-output differences for all F_i's in the n-cell GF-NLFSR can be summarized as follows:

$$\begin{aligned}
&\alpha_1 \xrightarrow{F_1} \epsilon \\
&\alpha_2 \xrightarrow{F_2} \epsilon + \alpha_2 && +\beta_1 \\
&\alpha_3 \xrightarrow{F_3} \epsilon + \alpha_2 + \alpha_3 && +\beta_1 + \beta_2 \\
&\quad\vdots \qquad\qquad \vdots && \qquad\vdots \\
&\alpha_n \xrightarrow{F_n} \epsilon + \alpha_2 + \alpha_3 + \ldots + \alpha_n && +\beta_1 + \beta_2 + \ldots + \beta_{n-1} \\
\epsilon + \alpha_2 + \alpha_3 + \ldots + \alpha_n &\xrightarrow{F_{n+1}} && \beta_1 + \beta_2 + \ldots + \beta_{n-1} + \beta_n
\end{aligned} \tag{1}$$

From Fact 1, we have the following:

$$DP(\alpha \xrightarrow{F} \beta) = \sum_{\epsilon \in GF(2)^B} DP(\alpha_1 \xrightarrow{F_1} \epsilon) DP(\alpha_2 \xrightarrow{F_2} \epsilon + \alpha_2 + \beta_1)$$
$$DP(\alpha_3 \xrightarrow{F_3} \epsilon + \alpha_2 + \alpha_3 + \beta_1 + \beta_2) \ldots DP(\epsilon + \alpha_2 + \ldots + \alpha_n \xrightarrow{F_{n+1}} \beta_1 + \ldots + \beta_n). \quad (2)$$

We shall show that at least 2 input differences in Equation 2 are non-zero when $\alpha \neq 0$. This implies that $DP(\alpha \xrightarrow{F} \beta) \leq p^2$. It suffices to prove this fact for the cases where only one of $\alpha_1, \alpha_2, \ldots, \alpha_n$ is non-zero.

(1) Suppose that only $\alpha_1 \neq 0$, then $\epsilon \neq 0$ (otherwise, $DP(\alpha_1 \xrightarrow{F_1} \epsilon = 0)$). Therefore, the input difference of F_{n+1}, i.e. $\epsilon + \alpha_2 + \ldots + \alpha_n = \epsilon$, is non-zero.
(2) Suppose that only $\alpha_2 \neq 0$, then the input difference of F_{n+1}, i.e. $\epsilon + \alpha_2 + \ldots + \alpha_n = \alpha_2$, is non-zero.

$\vdots$

(n) Suppose that only $\alpha_n \neq 0$, then the input difference of F_{n+1}, i.e. $\epsilon + \alpha_2 + \ldots + \alpha_n =$ α_n, is non-zero.

Therefore, at least 2 of the input differences are non-zero and $DP(\alpha \xrightarrow{F} \beta) \leq p^2$. □

5 Linear Hull Probability

We also have a result similar to Fact 2 for the linear hull probability of GF-NLFSR over $n+1$ rounds where the internal state is split into n equally sized blocks.

Theorem 2. *Let F be the $(n+1)$-round function in Figure 2 (right) of Appendix B where each $F_i : GF(2)^B \times GF(2)^B \times K_i' \mapsto GF(2)^B$ is of the form $F_i(x, k_i, k_i') = f_i(x + k_i, k_i')$ and each $f_i : GF(2)^B \times K_i' \mapsto GF(2)^B$ is bijective for all fixed $k_i' \in K_i'$. If $LP((f_i)_{max}) \leq q$ for each i, then $LP(F_{max}) \leq q^2$.*

Proof. Let the output mask value of F be $u = (u_1, \ldots, u_n) \neq 0$ and the input mask value be $w = (w_1, \ldots, w_n) \neq 0$. If the output mask value of F_1 is ϵ, it can be easily derived that we have the following individual round approximations:

$$\begin{array}{lllllll}
\epsilon & & \xleftarrow{F_1} & w_1 & & & \\
u_1 + u_2 & & \xleftarrow{F_2} \epsilon & + w_2 & & & \\
u_2 + u_3 & & \xleftarrow{F_3} \epsilon & & + w_3 & + u_1 + u_2 & \\
\vdots & & & & \vdots & \vdots & \\
u_{n-1} + u_n & \xleftarrow{F_n} \epsilon & & & + w_n + u_1 & + u_{n-1} & \\
u_n & \xleftarrow{F_{n+1}} \epsilon & & & + u_1 & & + u_n
\end{array}$$

Then Fact 1 gives

$$LP(u \xleftarrow{F} w) = \sum_{\epsilon \in GF(2)^B} LP(\epsilon \xleftarrow{F_1} w_1) LP(u_1 + u_2 \xleftarrow{F_2} \epsilon + w_2) LP(u_2 + u_3 \xleftarrow{F_3} \epsilon + w_3 + u_1 + u_2) \ldots LP(u_{n-1} + u_n \xleftarrow{F_n} \epsilon + w_n + u_1 + u_{n-1}) LP(u_n \xleftarrow{F_{n+1}} \epsilon + u_1 + u_n). \quad (3)$$

We shall show that at least 2 output mask values in Equation 3 are non-zero when $u, w \neq 0$. This will then imply that $LP(u \xleftarrow{F} w) \leq q^2$. If all the output mask values are equal to 0, i.e.

$$\epsilon = u_1 + u_2 = u_2 + u_3 = \ldots = u_{n-1} + u_n = u_n = 0,$$

then

$$\begin{aligned} & u_1 = u_2 = \ldots = u_n = 0 \\ \Rightarrow\ & u = 0 \end{aligned}$$

which gives a contradiction. Therefore, at least 1 output mask value is non-zero. Now we show that if only one of them is non-zero, then we will arrive at a contradiction.

(1) Suppose that only $\epsilon \neq 0$. Then $u_1 = u_2 = \ldots = u_n = 0$ which is a contradiction since $u \neq 0$.

(2) Suppose that only $u_1 + u_2 \neq 0$. Note that if $\epsilon = 0$, then $w_1 = 0$; otherwise, $LP(\epsilon \xleftarrow{F_1} w_1) = 0$. If $w_1 = 0$, then for other non-zero values of ϵ, $LP(\epsilon \xleftarrow{F_1} w_1) = 0$.

$$\begin{aligned} & \epsilon = u_2 + u_3 = u_3 + u_4 = \ldots = u_{n-1} + u_n = u_n = 0 \\ \Rightarrow\ & \epsilon + u_1 + u_n = u_1 = 0 \ \text{ (otherwise, } LP(u \xleftarrow{F} w) = 0) \\ & \text{and } \ u_2 = u_3 = \ldots = u_n = 0 \\ \Rightarrow\ & u = 0 \end{aligned}$$

which gives a contradiction.

(3) Suppose that only $u_2 + u_3 \neq 0$. Then

$$\begin{aligned} & \epsilon = u_1 + u_2 = u_3 + u_4 = \ldots = u_{n-1} + u_n = u_n = 0 \\ \Rightarrow\ & \epsilon + u_1 + u_n = u_1 = 0 \ \text{ (otherwise, } LP(u \xleftarrow{F} w) = 0) \\ \Rightarrow\ & u_2 = u_1 = 0 \ \text{ and } \ u_3 = u_4 = \ldots = u_n = 0 \\ \Rightarrow\ & u = 0 \end{aligned}$$

which gives a contradiction.

$$\vdots$$

(n) Suppose that only $u_{n-1} + u_n \neq 0$. Then

$$\begin{aligned} & \epsilon = u_1 + u_2 = u_2 + u_3 = \ldots = u_{n-2} + u_{n-1} = u_n = 0 \\ \Rightarrow\ & \epsilon + u_1 + u_n = u_1 = 0 \ \text{ (otherwise, } LP(u \xleftarrow{F} w) = 0) \\ \Rightarrow\ & u_1 = u_2 = \ldots = u_{n-1} = 0 \ \text{ and } \ u_n = 0 \\ \Rightarrow\ & u = 0 \end{aligned}$$

which gives a contradiction.

$(n+1)$ Suppose that only $u_n \neq 0$. Then

$$\begin{aligned}
&\epsilon = u_1 + u_2 = u_2 + u_3 = \ldots = u_{n-1} + u_n = 0 \\
\Rightarrow\ &u_1 = u_2 = \ldots = u_{n-1} = u_n \\
\Rightarrow\ &w_1 = 0,\ \epsilon + w_2 = w_2 = 0,\ \epsilon + w_3 + u_1 + u_2 = w_3 = 0, \ldots, \\
&\epsilon + w_n + u_1 + u_{n-1} = w_n = 0 \ \text{(otherwise, } LP(u \xleftarrow{F} w) = 0) \\
\Rightarrow\ &w = 0
\end{aligned}$$

which gives a contradiction.

Therefore, at least 2 of the output mask values must be non-zero and $LP(u \xleftarrow{F} w) \leq q^2$. □

6 Frequencies of Differential and Linear Hull Probabilities and Expected Value

Here we calculate the approximate number of input-output differences $(\alpha \longrightarrow \beta)$ or mask values $(u \longleftarrow w)$ with $DP(\alpha \xrightarrow{F} \beta) \leq p^x$ or $LP(u \xleftarrow{F} w) \leq q^x$ respectively $(x = 2, \ldots, n)$. With reference to the sequence of differences and mask values stated in Sections 4 and 5, let $\Delta = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ and $\Omega = \{u_1 + u_2, u_2 + u_3, \ldots, u_{n-1} + u_n, u_n\}$.

Define $N_d(x)$ (respectively $N_l(x)$) as the number of input-output differences (α, β) (respectively input-output masks (w, u)) when there are x non-zero entries in Δ (respectively Ω). From the structure of n-cell, having x non-zero entries in Δ or Ω will ensure $DP(\alpha \xrightarrow{F} \beta) \leq p^x$ or $LP(u \xleftarrow{F} w) \leq q^x$ respectively. The only exception is when $x = 1$, where we still have $DP(\alpha \xrightarrow{F} \beta) \leq p^2$ or $LP(u \xleftarrow{F} w) \leq q^2$ by Theorems 1 and 2.

Various cases for the input-output pairs and their corresponding bounds are shown in Table 1 in Appendix A. When there are x non-zero entries in Δ, the number of possible input-output differences is given by $N_d(x) = \binom{n}{x}(2^B - 1)^x(2^{nB} - 1)$. This is because there are $\binom{n}{x}$ possible input differences with x non-zero entries where each non-zero entry has $2^B - 1$ possibilities, and there are $2^{nB} - 1$ possibilities for the non-zero output difference. We have an identical formula for $N_l(x)$ by a similar reason.

Based on the values $N_d(x)$ and $N_l(x)$, we see that when an attacker uses plaintexts such that the input differences α (output mask values u resp.) are randomly chosen, he is more likely to obtain a bound much lower than p^2 (q^2 resp.) since most of the input differences α (output mask values u resp.) give rise to differential probabilities $DP(\alpha \xrightarrow{F} \beta)$ (linear hull probabilities $LP(u \xleftarrow{F} w)$ resp.) whose bounds are much smaller than p^2 (q^2 resp.). Such a scenario may occur when, for example, the $(n+1)$-round structure is an intermediate portion of a cipher so that the attacker does not have much control over the input differences (output mask values resp.). This motivates our desire to have more practically useful differential and linear hull probability bounds. For this purpose, we make the following definitions:

Definition 1. *The* expected differential probability *is defined as* $E_d = \frac{\sum_{\alpha,\beta\neq 0} DP(\alpha \xrightarrow{F} \beta)}{\#\{(\alpha,\beta)|\alpha,\beta\neq 0\}}$ *and the* expected linear probability *is defined as* $E_l = \frac{\sum_{w,u\neq 0} LP(u \xleftarrow{F} w)}{\#\{(w,u)|w,u\neq 0\}}$

Note that $\sum_{x=2}^{n} N_d(x) = (2^{nB}-1)^2$ which is the total number of differences with both input and output non-zero. We may make a similar observation for the linear case. From this table, we may directly calculate the proportion of input-output differences (mask values resp.) with differential (linear hull resp.) probability $\leq p^x$ (q^x resp.). Denote the approximate proportion of input-output differences with differential probability $\leq p^x$ by $P_d(x) = \frac{N_d(x)}{\#\{(\alpha,\beta)|\alpha,\beta\neq 0\}}$. Likewise, denote the approximate proportion of input-output mask values with linear hull probability $\leq q^x$ by $P_l(x) = \frac{N_l(x)}{\#\{(w,u)|w,u\neq 0\}}$. It can be computed that the statistics are heavily skewed towards the lowest probabilities instead of p^2 or q^2. For example, when $n = 4$, $B = 8$, and when $n = 4$, $B = 16$, we have the following proportions shown in Table 2 in Appendix A.

Using the frequency values in Table 1, we can derive that

$$\begin{aligned} E_d &\leq \frac{1}{(2^{nB}-1)^2}\left[\binom{n}{1}(2^B-1)\cdot(2^{nB}-1)p^2 + \sum_{x=2}^{n}\binom{n}{x}(2^B-1)^x\cdot(2^{nB}-1)p^x\right] \\ &< \frac{1}{(2^{nB}-1)}\left[\binom{n}{1}(2^B-1)p + \sum_{x=2}^{n}\binom{n}{x}(2^B-1)^x p^x\right] \\ &< \frac{1}{(2^{nB}-1)}\left[\sum_{x=0}^{n}\binom{n}{x}(2^B-1)^x p^x\right] \\ &= \frac{1}{(2^{nB}-1)}(1+(2^B-1)p)^n \\ &\approx (2^{-B}+p)^n, \end{aligned} \tag{4}$$

where we have approximated $2^B - 1$ and $2^{nB} - 1$ by 2^B and 2^{nB} respectively because B is usually much larger than 1. Similarly, we have $E_l \leq (2^{-B} + q)^n$.

For example, when $n = 4$, $B = 8$ and $p = 2^{-6}$, the bound in (4) is approximately $2^{-22.7}$, which is much better than the 2^{-12} bound obtained from Theorem 1.

7 Impossible Differential Characteristics

Impossible differential cryptanalysis is a variant of differential cryptanalysis against block ciphers. It was applied against Skipjack to reject wrong key candidates by using input and output difference pairs whose probabilities are zero. It can also be used to attack a 5-round Feistel structure even though the 3-round Feistel structure with bijective round functions are provably secure against differential and linear cryptanalysis.

In impossible differential cryptanalysis, impossible differential characteristics are used to retrieve a subkey material for the first or last several rounds of block ciphers. Thus the security of a block cipher against impossible differential cryptanalysis can be evaluated by impossible differential characteristics [26].

A general tool, called U-method, was introduced by [26] to find the maximum number of rounds for impossible differential characteristics. An algorithm, Algorithm 1, was also provided to compute the maximum length of impossible differential characteristics that can be found by the U-method. Interested readers may refer to [26] for the technicalities. By modifying Algorithm 1, we can determine the impossible differential characteristics of the block cipher structures. The following result for our block cipher n-cell GF-NLFSR is based on the simulation. Here, a r-round impossible differential characteristic is denoted by $\alpha \nrightarrow_r \beta$ where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$.

Proposition 3. *The maximum number of rounds for impossible differential characteristics that can be found by the U-method for n-cell GF-NLFSR is $2n-1$. Generalized impossible differential characteristics are*

$$(0, \cdots, 0, \alpha_n) \nrightarrow_{2n-1} (\beta_1, \beta_2, 0, \cdots, 0), \text{ where } \alpha_n \neq 0, \beta_1 = \beta_2 \neq 0,$$

and,

$$(0, \cdots, 0, \alpha_n) \nrightarrow_{2n-1} (\beta_1, 0, \cdots, 0, \beta_n), \text{ where } \alpha_n \neq 0, \beta_1 = \beta_n \neq 0.$$

In particular, when $n = 4$, a 7-round impossible differential characteristic is $(0, 0, 0, \gamma) \nrightarrow_7 (\gamma, \gamma, 0, 0)$, with the input and output differences to and after each round as follows:

$$(0,0,0,\gamma) \to (0,0,\gamma,\gamma) \to (0,\gamma,\gamma,0) \to (\gamma,\gamma,0,0) \to (\gamma,0,0,\gamma+\delta) \to (0,0,\gamma+\delta,?)$$
$$\to (0,\gamma+\delta,?,?) \neq (0,\gamma,\gamma,0) \leftarrow (\gamma,\gamma,0,0),$$

where γ, δ and ? denote nonzero nonfixed, nonzero fixed, and, nonfixed differences respectively. We can thus use a 7-round impossible differential to conduct an impossible differential attack.

8 Integral Attack

The integral attack is a cryptanalytic technique on block ciphers. It was originally proposed by Knudsen and Wagner in 2002 [9] and has since been adapted to cryptanalyse various ciphers. In this attack, a set of chosen plaintexts is encrypted and the corresponding ciphertexts are decrypted a certain number of rounds using all possible subkey guesses. The plaintext set is chosen such that one part is held constant while another part varies over all possibilities. Using this plaintext set, a distinguisher is produced after a certain number of rounds, which enables the attacker to determine the correct subkey which was used for partial decryption. We shall show that for $n \geq 2$, GUFN has a $(3n-1)$-round distinguisher, where n is the number of blocks.

In this section, we let the n-tuple $(A, c, \ldots, c)$ denote a set of 2^B plaintexts where the leftmost block of B bits vary over all 2^B possibilities, while the other blocks, represented by c, are constants. We shall let uppercase letters represent a set that varies over all values in $GF(2)^B$. Thus, we obtain the set $(c, c, \ldots, c, D+c)$ after a round of encryption. Consequently, we have the following sequence of round-by-round output sets after n rounds:

$$\begin{aligned}(A, c, \ldots, c) &\rightarrow (c, c, \ldots, c, D + c)\\ &\vdots\\ &\rightarrow (c, D + c, D + c, c, \ldots, c)\\ &\rightarrow (D + c, D + c, c, c, \ldots, c).\end{aligned}$$

After another n rounds we have the following:

$$\begin{aligned}(D + c, D + c, c, \ldots, c) &\rightarrow (D + c, c, \ldots, c, D + E + c)\\ &\rightarrow (c, \ldots, c, D + E + c, D + E + G + c)\\ &\rightarrow (c, \ldots, c, D + E + c, D + E + G + c, G + c)\\ &\rightarrow (c, \ldots, c, D + E + c, D + E + G + c, G + c, c)\\ &\vdots\\ &\rightarrow (D + E + c, D + E + G + c, G + c, c, \ldots, c).\end{aligned}$$

The distinguishing property of certain values in the set is subsequently destroyed block by block. We obtain this, after another $n - 1$ rounds:

$$\begin{aligned}(D + E + c, D + E + G + c, G + c, c, \ldots, c) &\rightarrow (D + E + G + c, G + c, c, \ldots, c, ?)\\ &\vdots\\ &\rightarrow (c, ?, ?, \ldots, ?).\end{aligned}$$

The set of ciphertexts after $3n-1$ rounds of encryption will be constant in the leftmost block. This property can be exploited as a distinguisher if the cipher has only slightly more than $3n - 1$ rounds. Although some minor detail of the above proof does not apply for the cases $n = 2$ and $n = 3$, it can be easily verified, using a similar approach, that the $(3n - 1)$-round result still holds for these values of n.

9 Application: New Block Cipher Four-Cell

As an application, we design a new 128-bit block cipher, Four-Cell, with 128-bit key size. It uses the block cipher structure described in Section 3 with four cells where each cell is a 32-bit word. The block cipher has 25 rounds and uses two types of nonlinear functions for round i, defined as follows:

$$f_i(x_i, k_i, 0) = MDS(S(x_i + k_i)), \text{ for rounds } i = 1, 2, \ldots, 5 \text{ and } i = 21, 22, \ldots, 25.$$
$$f_i(x_i, k_i, k_i') = S(MDS(S(x_i + k_i)) + k_i'), \text{ for rounds } i = 6, 7, \ldots, 20.$$

Here, $S : GF(2^8)^4 \rightarrow GF(2^8)^4$ is defined as

$$S(x_1, x_2, x_3, x_4) = (Inv(x_1), Inv(x_2), Inv(x_3), Inv(x_4)),$$

where $Inv : GF(2^8) \rightarrow GF(2^8)$ is affine equivalent to $x \mapsto x^{254}$ on $GF(2^8)$ (e.g., the AES S-box). $MDS : GF(2^8)^4 \rightarrow GF(2^8)^4$ is a 4-byte to 4-byte maximal distance separable transform with optimal branch number 5 (e.g., the MixColumn operation in AES). Note that one subkey and one layer of S-box is used for rounds $1, 2, \ldots, 5$ and $21, 22, \ldots, 25$ while two subkeys and two layers of S-boxes are used for rounds $6, 7, \ldots, 20$. Moreover, we XOR a 128-bit post-whitening key K_{26} to the output after 25 rounds.

We leave the implementation of a secure key schedule open to the reader. One possibility would be to use a similar cipher with 26 rounds as the key schedule. The only difference is that the nonlinear function for all rounds is defined as $f_i(x_i, c_i, 0) = MDS(S(x_i + c_i))$ where the c_i's are distinct randomly generated constants. The input to the key-schedule is the secret key K. The least significant 32 output bits (i.e., nonlinear output) of round i of the key schedule can be used as the i^{th}-round cipher subkey k_i. For rounds $i = 6, 7, \ldots, 20$, we can take the next 32 least significant output bits of round i of the key schedule to be k_i'. The post-whitening key K_{26} is the 128-bit output of round 26 of the key schedule.

In the following section, we demonstrate the security of Four-Cell against a slew of cryptanalytic attacks, in addition to differential and linear cryptanalysis.

9.1 Security of Four-Cell

Based on the cryptographic properties of the inversion S-box and MDS transform, it is easy to see that the differential and linear characteristic probabilities of Four-Cell are at most $2^{-192} < 2^{-128}$. Therefore it is practically secure against differential and linear cryptanalysis. By combining the results of Sections 4 and 5 with the differential and linear probability of a SPN structure [19], we can show that the true differential and linear probabilities of Four-Cell are at most $2^{-55.39}$ and $2^{-52.96}$ respectively.

However, this bound is tight only for a negligible number of input-output differences and masks. From the results of Section 6, the expected differential and linear probabilities are actually $2^{-110.5}$ and $2^{-105.79}$ respectively. Based on the true differential probability, we can show that if we split Four-Cell into two sub-ciphers with true differential probabilities p and q, then $(pq)^2 \leq 2^{-221.57} \ll 2^{-128}$. This will ensure Four-Cell is secure against boomerang attack.

From the results of Section 7, there is a 10-round impossible differential attack based on a 7-round impossible differential distinguisher. In addition, from the results of Section 8, there is a 14-round attack based on an 11-round integral attack distinguisher. However, it is unlikely that these two attacks will work against the full cipher which will require a 21-round impossible differential/integral attack distinguisher.

Furthermore, Four-Cell is secure against higher order differential and cube attacks after 9 rounds, because the algebraic degree of the cipher attains the maximum degree 127. We also see that interpolation attack might not work as the cipher will be a complex multivariable equation over $GF(2^8)$. By analyzing the works in [3,4,5,12], we are able to show that the XSL attack (over $GF(2)$ and $GF(2^8)$) does not work on our cipher. Finally, Four-Cell is secure against slide attack because of its distinct round structures and distinct round subkeys. Full details on the analysis of Four-Cell can be found in the full version of our paper (with the same title) on the IACR cryptology eprint archive.

9.2 Implementation Considerations

The Four-Cell cipher uses 160 S-boxes based on the inversion function on $GF(2^8)$. This is the same as the number of S-boxes used in AES. However only 25 MDS transform are used when compared to AES, which uses 40 MDS transforms. This might make the cipher faster in hardware implementations where the S-box and MDS are not combined into a T-table. Moreover, note that the computation of the nonlinear function in any 4 consecutive rounds of the cipher can be performed in parallel for faster encryption speed, giving it an added advantage over other GUFNs such as SMS4. Thus the Four-Cell cipher which (like the AES cipher) has provable security against existing block cipher attacks can be viewed as a viable alternative.

Also note that although the inverse cipher of Four-cell is distinct from Four-cell itself and therefore coding might potentially take up more space in hardware, it is still useful for modes of operation such as counter mode, output feedback (OFB) mode, and cipher feedback (CFB) mode, where no inverse cipher is required.

Acknowledgement

The authors would like to thank the anonymous reviewer of CT-RSA who pointed out the integral attack on Four-Cell.

References

1. Biham, E., Shamir, A.: Differential Cryptanalysis of the Data Encryption Standard. Springer, New York (1993)
2. Biryukov, A., Wagner, D.: Slide Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 245–259. Springer, Heidelberg (1999)
3. Cid, C., Leurent, G.: An Analysis of the XSL Algorithm. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 333–352. Springer, Heidelberg (2005)
4. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. IACR eprint server 2002/044 (March 2002), http://www.iacr.org
5. Courtois, N., Pieprzyk, J.: Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 267–287. Springer, Heidelberg (2002)

6. Daemen, J., Rijmen, V.: The Design of Rijndael: AES, The Advanced Encryption Standard. Springer, Heidelberg (2002)
7. Dinur, I., Shamir, A.: Cube Attacks on Tweakable Black Box Polynomials, Cryptology Eprint Archive, Report 2008/385
8. Jakobsen, T., Knudsen, L.R.: Attacks on Block ciphers of Low Algebraic Degree. Journal of Cryptology 14, 197–210 (2001)
9. Knudsen, L.R., Wagner, D.: Integral Cryptanalysis. In: Daemen, J., Rijmen, V. (eds.) FSE 2002. LNCS, vol. 2365, pp. 112–127. Springer, Heidelberg (2002)
10. Lai, X.: On the Design and Security of Block Ciphers, Thesis (1992)
11. Lai, X., Massey, J.L., Murphy, S.: Markov Ciphers and Differential Cryptanalysis. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 17–38. Springer, Heidelberg (1991)
12. Lim, C.W., Khoo, K.: An Analysis of XSL Applied on BES. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 242–253. Springer, Heidelberg (2007)
13. Liu, F., Ji, W., Hu, L., Ding, J., Lv, S., Pyshkin, A., Weinmann, R.: Analysis of the SMS4 Block Cipher. In: Pieprzyk, J., Ghodosi, H., Dawson, E. (eds.) ACISP 2007. LNCS, vol. 4586, pp. 158–170. Springer, Heidelberg (2007)
14. Matsui, M.: Linear Cryptanalysis Method for DES Cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994)
15. Matsui, M.: New Structure of Block Ciphers with Provable Security Against Differential and Linear Cryptanalysis. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 205–218. Springer, Heidelberg (1996)
16. Murphy, S., Robshaw, M.: Essential Algebraic Structure within the AES. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 1–16. Springer, Heidelberg (2002)
17. Nyberg, K.: Linear Approximation of Block Ciphers. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 439–444. Springer, Heidelberg (1995)
18. Nyberg, K.: Generalized Feistel Networks. In: Kim, K.-c., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 91–104. Springer, Heidelberg (1996)
19. Park, S., Sang, S.H., Lee, S., Lim, J.: Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES. In: Johansson, T. (ed.) FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer, Heidelberg (2003)
20. Schneier, B., Kelsey, J.: Unbalanced Feistel Networks and Block-Cipher Design. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 121–144. Springer, Heidelberg (1996)
21. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-bit Blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
22. Daemen, J., Knudsen, L., Rijmen, V.: The Block Cipher Square. In: Biham, E. (ed.) FSE 1997. LNCS, vol. 1267, pp. 149–165. Springer, Heidelberg (1997)
23. Wagner, D.: The Boomerang Attack. In: Knudsen, L.R. (ed.) FSE 1999. LNCS, vol. 1636, pp. 156–170. Springer, Heidelberg (1999)
24. Wallen, J.: Design Principles of the KASUMI Block Cipher (June 2008), http://www.tml.tkk.fi/Opinnot/Tik-110.501/2000/papers/wallen.pdf
25. Wu, W., Zhang, W., Lin, D.: On the Security of Generalized Feistel Scheme with SP Round Function. International Journal of Network Security 3(3), 215–224 (2006)
26. Kim, J., Hong, S., Sung, J., Lee, S., Lim, J., Sung, S.: Impossible Differential Cryptanalysis for Block Cipher Structures. In: Johansson, T., Maitra, S. (eds.) INDOCRYPT 2003. LNCS, vol. 2904, pp. 82–96. Springer, Heidelberg (2003)

A Tables

Table 1. Frequencies of differential and linear hull probabilities

Differential probability	Linear hull probability	$N_d(x)/N_l(x)$	# of elements in Δ (or Ω resp.) which are non-zero
$\leq p^n$	$\leq q^n$	$(2^B-1)^n \cdot (2^{nB}-1)$	n
$\leq p^{n-1}$	$\leq q^{n-1}$	$\binom{n}{n-1}(2^B-1)^{n-1} \cdot (2^{nB}-1)$	$n-1$
$\leq p^{n-2}$	$\leq q^{n-2}$	$\binom{n}{n-2}(2^B-1)^{n-2} \cdot (2^{nB}-1)$	$n-2$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\leq p^3$	$\leq q^3$	$\binom{n}{3}(2^B-1)^3 \cdot (2^{nB}-1)$	3
$\leq p^2$	$\leq q^2$	$\binom{n}{2}(2^B-1)^2 \cdot (2^{nB}-1)$	2
$\leq p^2$	$\leq q^2$	$\binom{n}{1}(2^B-1) \cdot (2^{nB}-1)$	1

Table 2. Distribution of proportions

Differential probability	Linear hull probability	x	$P_d(x)/P_l(x)$	
			$n=4, B=8$	$n=4, B=16$
p^4	q^4	4	0.9844663148	0.9999389662
p^3	q^3	3	0.1544260886	0.0000610323
p^2	q^2	2	0.0000910763	$0.1396955440 \times 10^{-8}$

B Figures

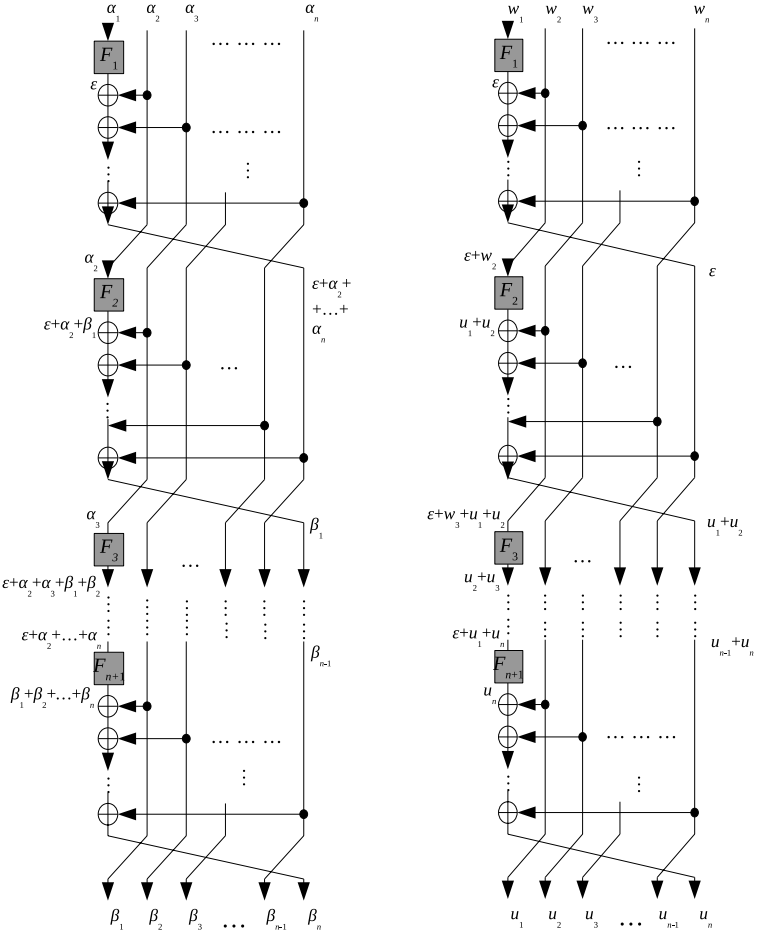


Fig. 2. Sequence of differences(left)/mask values(right) for $n+1$ rounds of GF-NLFSR

Lightweight Block Ciphers Revisited: Cryptanalysis of Reduced Round PRESENT and HIGHT

Onur Özen[1], Kerem Varıcı[2,⋆], Cihangir Tezcan[3], and Çelebi Kocair[4]

[1] EPFL IC LACAL Station 14. CH-1015 Lausanne, Switzerland
onur.ozen@epfl.ch
[2] K.U.Leuven, Dept. of Electrical Engineering, ESAT/SCD/COSIC and IBBT
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
kerem.varici@esat.kuleuven.be
[3] METU, Institute of Applied Mathematics,
Department of Cryptography, 06531 Ankara, Turkey
cihangir@metu.edu.tr
[4] METU, Department of Computer Engineering, 06531 Ankara, Turkey
celebi@ceng.metu.edu.tr

Abstract. Design and analysis of lightweight block ciphers have become more popular due to the fact that the future use of block ciphers in ubiquitous devices is generally assumed to be extensive. In this respect, several lightweight block ciphers are designed, of which PRESENT and HIGHT are two recently proposed ones by Bogdanov *et al.* and Hong *et al.* respectively. In this paper, we propose new attacks on PRESENT and HIGHT. Firstly, we present the first related-key cryptanalysis of 128-bit keyed PRESENT by introducing 17-round related-key rectangle attack with time complexity approximately 2^{104} memory accesses. Moreover, we further analyze the resistance of HIGHT against impossible differential attacks by mounting new 26-round impossible differential and 31-round related-key impossible differential attacks where the former requires time complexity of $2^{119.53}$ reduced round HIGHT evaluations and the latter is slightly better than exhaustive search.

Keywords: PRESENT, HIGHT, Related-Key Attack, Rectangle Attack, Impossible Differential Attack.

1 Introduction

Lightweight cryptography has become very vital with the emerging needs in sensitive applications like RFID (Radio-frequency identification) systems and

⋆ This work was sponsored by the Research Fund K.U.Leuven, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy) and by the European Commission through the ICT Programme under Contract ICT-2007-216676 (ECRYPT II). The information in this paper is provided as is, and no warranty is given or implied that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 90–107, 2009.

sensor networks. For these types of special purposes, there is a strong demand in designing secure lightweight cryptographic modules. After the selection of AES (Advanced Encryption Standard) [1], the research on efficient implementation of AES, especially for such constrained environments, brought special attention in research community. Even though it is highly convenient for such devices, the research on designing and analyzing new lightweight block ciphers that are more efficient than AES on these platforms poses huge challenges. For this purpose, several block ciphers are designed as potential candidates such as HIGHT [2,3], PRESENT [4], mCrypton [5], SEA [6], CGEN [7], DES [8] and DESXL [8].[1]

A recent portfolio[2], which contains four software and three hardware oriented stream ciphers [11], has been announced by ECRYPT as part of eSTREAM project to identify new stream ciphers that might become suitable for widespread adoption including lightweight platforms. As a result, stream ciphers are shown to be highly efficient on both software and hardware implementations comparing to block ciphers. To fill this efficiency gap, PRESENT [4] was proposed by Bogdanov *et al.* at CHES '07 as an ultra-lightweight block cipher with 31 rounds offering as good hardware and software performance as current modern stream ciphers while it is more efficient than many known block ciphers.

Basic security analysis of PRESENT is provided in [4] by showing resistance against known attacks such as differential, linear cryptanalysis and their variants. Recent differential attacks [12,13] on 16 and 19 rounds of PRESENT provide similar results as in the original proposal with some practical evidence of applied characteristics where the latter is an attempt to combine algebraic attacks with differential cryptanalysis. Another type of an attack called *bit-pattern based integral attack* [14] is applicable up to seven rounds of PRESENT. More recently, a new type of attack, called *statistical saturation attack* was proposed in [15] and shown to be applicable up to 24 rounds of PRESENT. Previous results on the analysis of PRESENT are summarized in Table 1.

The security of PRESENT against key schedule weaknesses is provided by showing the resistance against slide [16] and related-key differential attacks [17] where slide attacks are inapplicable because of the round dependent counters in key scheduling algorithm. Related-key differential attacks, on the other hand, are also believed to be inapplicable because of the sufficient non-linearity due to key scheduling algorithm.

HIGHT [2,3] is a South Korean standard encryption algorithm enjoying the use of a low-resource hardware implementation. It is a 32 round block cipher proposed one year before PRESENT at CHES '06 by Hong *et al.* to be used for ubiquitous computing devices. The prominent characteristic of HIGHT is that it makes use of simple byte oriented operations such as exclusive-or, addition modulo 256 and cyclic rotation which offers nice performance on hardware.

[1] TEA [9] and XTEA [10] can also be given as lightweight block ciphers which were designed before AES.

[2] The original hardware-oriented portfolio of eSTREAM contains four hardware-oriented stream ciphers. However, F-FSCR-H has recently been eliminated from the eSTREAM portfolio.

Table 1. Summary of the attacks on PRESENT and HIGHT (CP-Chosen Plaintext, MA-Memory Accesses, PR-Reduced round PRESENT evaluation, HE-Reduced round HIGHT evaluation)

Cipher	Rounds	Key Size	Attack Type	Data Complexity	Time Complexity	Memory Complexity	Reference
PRESENT	24	80	Stat. Sat.	2^{60}CP	2^{20} PR	2^{16} bytes	[15]
	24	80	Stat. Sat.	2^{57}CP	2^{57} PR	2^{32} bytes	[15]
	7	128	Bit-Pat. Int.	$2^{24.3}$CP	$2^{100.1}$ MA	2^{77} bytes	[14]
	17	128	Rel.-Key Rec.	2^{63} CP	2^{104} MA	2^{53} bytes	§3.1
	19	128	Alg.-Dif.	6×2^{62} CP	2^{113} MA	not specified	[13]
HIGHT	18	128	Imp. Dif.	$2^{46.8}$ CP	$2^{109.2}$ HE	not specified	[2]
	25	128	Imp. Dif.	2^{60} CP	$2^{126.78}$HE	not specified	[18]
	26	128	Imp. Dif.	2^{61} CP	$2^{119.53}$HE	2^{109} bytes	§4.1
	26	128	Rel.-Key Rec.	$2^{51.2}$ CP	$2^{120.41}$HE	not specified	[18]
	28	128	Rel.-Key Imp.	2^{60} CP	$2^{125.54}$HE	not specified	[18]
	31	128	Rel.-Key Imp.	2^{64} CP	$2^{127.28}$HE	2^{117} bytes	§4.2

The security of HIGHT is investigated in [2] by showing resistance against differential, linear, truncated differential, boomerang, rectangle, impossible differential attacks and their related-key variants. In [2], the safety margin was shown to be 13 rounds as the best attack covers 19 rounds. Recent serious attacks [19] by Lu on reduced round HIGHT make use of 25, 26 and 28 round impossible differential, related-key rectangle and related-key impossible differential attacks: the last attack is the best attack on HIGHT so far that reduced the safety margin from 13 rounds to four rounds.

In this work, we present the first related-key cryptanalysis of PRESENT. For 128-bit keyed version, we introduce 17-round related-key rectangle attack [20,21,22] which is not explicitly mentioned in the original proposal [4]. Moreover, we further analyze the resistance of HIGHT against impossible differential attacks [23,24]. Firstly, we improve 25-round impossible differential attack of Lu by introducing a new characteristic to 26 rounds and update 28-round related-key impossible differential attack on 31 rounds. To the best of our knowledge, these are the best cryptanalytic results on HIGHT. We provide a summary of our results in Table 1.

The organization of the paper is as follows. In Section 2, we give a brief description of the block ciphers PRESENT and HIGHT. Section 3 introduces the idea behind the related-key attacks on PRESENT and contains related-key rectangle attack on 17-round. In Section 4, we introduce our improved impossible differential and related-key impossible differential attacks on reduced round HIGHT. We conclude with Section 5 and provide supplementary details about the paper in Appendices.

2 The Block Ciphers PRESENT and HIGHT

2.1 Notation

For PRESENT and HIGHT, we use the same notation to denote the variables used in this paper. For the sake of clarity and the parallelism with the previous work

Table 2. Notation

$\oplus$	Bitwise logical exclusive OR (XOR)
$\boxplus$	Addition modulo 2^8
$\lll$i	Left cyclic rotation by i bits
PRESENT-n-r	PRESENT reduced to r-rounds with n-bit secret key
K_i	ith subkey of PRESENT
S_i	ith S-Box of PRESENT
$e_{j_1,\ldots,j_k}$	A word with zeros in all positions but bits $j_1, \ldots, j_k$
HIGHT-r	HIGHT reduced to r-rounds
e_j	A byte with zeros in all positions but bit j ($0 \leqslant j \leqslant 7$)
$e_{j,\sim}$	A byte that has zeros in bits 0 to $j-1$, a one in bit j and indeterminate values in bits $(j+1)$ to 7
$e_{\bar{j},\sim}$	A byte that has zeros in bits 0 to j and indeterminate values in bits $(j+1)$ to 7
?	An arbitrary byte
$X_{i,j}$	jth byte of state variable of round i of HIGHT, ($0 \leqslant j \leqslant 7$) ($0 \leqslant i \leqslant 32$)
MK_i	ith secret key byte of HIGHT
WK_i	ith whitening key byte of HIGHT
SK_i	ith subkey byte of HIGHT

[19], we use exactly the same notation for HIGHT which is provided in Table 2. Throughout the paper, it is assumed that the rounds are numbered from zero and the leftmost bit is the most significant bit in a byte or a word.

2.2 PRESENT

PRESENT is a 31-round (and an output whitening at the end) SPN (Substitution Permutation Network) type block cipher with block size of 64 bits that supports 80 and 128-bit secret key. Round function of PRESENT, which is depicted in Figure 1, is same for both versions of PRESENT and consists of standard operations such as subkey XOR, substitution and permutation: At the beginning of each round, 64-bit input of the round function is XORed with the subkey. Just after the subkey XOR, 16 identical 4×4-bit S-boxes are used in parallel as a non-linear substitution layer and finally a permutation is performed so as to provide diffusion.

The subkeys for each round are derived from the user-provided secret key by the key scheduling algorithm. We provide only the details of the key scheduling algorithm of PRESENT-128 as it is the main target of this paper: 128-bit secret key is stored in a key register K and represented as $k_{127}k_{126}\ldots k_0$. The subkeys K_i $(0 \leq i \leq 31)$ consist of 64 leftmost bits of the actual content of register K. After round key K_i is extracted, the key register K is rotated by 61 bit positions

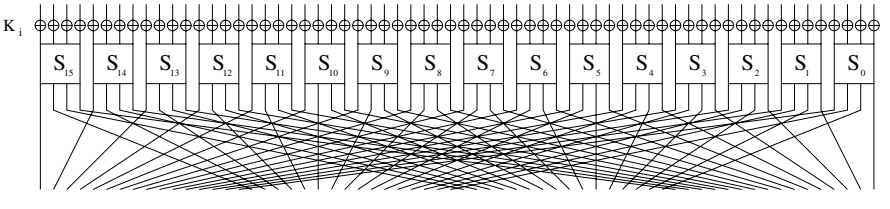


Fig. 1. Round function of PRESENT

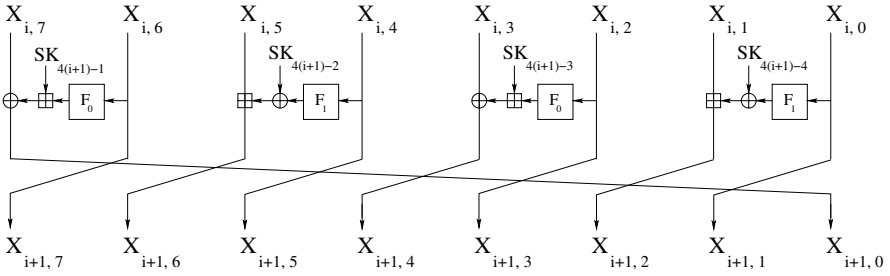


Fig. 2. ith round of HIGHT for $i = 0, \ldots, 31$

to the left, then S-box is applied to the left-most eight bits of the key register and finally the round counter value, which is a different constant for each round, is XORed with bits $k_{66}k_{65}k_{64}k_{63}k_{62}$. Further details about the specification of PRESENT are provided in [4].

2.3 HIGHT

HIGHT is a 32-round block cipher with 64-bit block size and 128-bit user key that makes use of an unbalanced Feistel Network. The encryption function starts with an Initial Transformation (IT) that is applied to plaintexts together with input whitening keys WKs. At the end of 32 rounds, in order to obtain the ciphertexts, a Final Transformation (FT) is applied to the output of the last round together with an output whitening. The byte-oriented round function, shown in Figure 2, uses modular addition, XOR and linear subround functions F_0 and F_1; the latter can be described as follows:

$$\begin{aligned} F_0(x) &= (x \lll 1) \oplus (x \lll 2) \oplus (x \lll 7) \\ F_1(x) &= (x \lll 3) \oplus (x \lll 4) \oplus (x \lll 6) \end{aligned}$$

HIGHT only works with 128-bit secret key MK which is treated as 16 bytes, $(MK_{15}, \ldots, MK_0)$. The key schedule of HIGHT uses additional constants to avoid the self similarity in the key scheduling algorithm which prevents cipher from slide attacks. Input-output whitening keys and round subkeys are obtained by permuting the 16 bytes of the original key and using addition with constants. Table 9 will be extensively used in this paper that shows the relations between the original and the subkey bytes. Namely, each value in a row represents the obtained whitening and subkey bytes once the corresponding byte in the first column of the same row of the original key is known. Further details about the specification of HIGHT are provided in [2,3].

3 The Related-Key Attacks on PRESENT

The idea behind the related-key attacks on PRESENT is to benefit from the slow mixing in the key scheduling algorithm which makes use of only one or two S-box operations (depending on the version) during each iteration. To achieve this

goal, we made an efficient search for related-key differentials of PRESENT which was done by flipping at most two bits of the original key. The crucial part of the key differentials is that we only consider the trivial differentials. More precisely, all reduced round key differentials in our attacks work with probability one.

In the original proposal of PRESENT [4], the resistance against differential and linear attacks are given by the bounds provided by the minimum number of active S-boxes. This approach also works for showing resistance against wide variety of attacks. A recent differential attack [12] uses same idea to attack the cipher by increasing the overall probability of the characteristics more effectively. Although there is no contradiction with the security claims given in [4], the differential attack in [12] provides a practical evidence. In this work, however, our aim is quite different and simple in that we try to decrease the number of active S-boxes (NAS). In order to do so, we cancel the intermediate differences with the subkey differences and construct our differentials by activating at most five S-boxes at the beginning. At the end, we are able to construct related-key differentials having less active S-boxes than given in [12]. As an example, for PRESENT-80, the minimum number of active S-boxes for any five-round differential characteristic is given to be ten in [4,12]. However, we found several five-round related-key differentials with only three active S-boxes.

Although it seems quite promising, as the number of rounds increases, the minimum number of active S-boxes gets closer to the one given in the original proposal [12] and the overall probabilities of the characteristics are not optimal. Still, for less number of rounds the related-key differentials are efficient and the number of possible characteristics are quite high. So, attacks like the related-key rectangle attack are easily applicable.

3.1 The Related-Key Rectangle Attack on PRESENT-128-17

The related-key rectangle attack is the clever extension of differential cryptanalysis. In rectangle-boomerang style attacks, the attacker uses two short differential characteristics instead of one long differential characteristic. The aim is to benefit from the slow mixing in relatively reduced round versions of the attacked cipher. We provide a brief description about the related-key rectangle attack in Appendix A and follow the mounted attack on PRESENT. Throughout the paper, the related-key rectangle attack is assumed to be mounted by using four related keys.

Let E denote the encryption function of PRESENT-n-r. We treat E as a cascade of four subciphers as $E = E_f \circ E_1 \circ E_0 \circ E_b$ where E is composed of a core $E' = E_1 \circ E_0$ covered by additional rounds, E_b and E_f which are the subciphers before and after the core function respectively.

For the related-key rectangle attack on PRESENT-128-17, we use the following decomposition: E_0 starts with the first round and ends just after the subkey XOR in round eight. E_1, on the other hand, commences with the substitution layer in round eight and stops at the end of round 14[3]. Round 0 and round

[3] This decomposition is not unique and can be done in various ways.

Table 3. An example of related-key differential used in E_0

r	Input Difference $\Delta(I)$	Key Difference $\Delta(K)$	$\Delta(I) \oplus \Delta(K)$	Output Difference $\Delta(O)$	NAS	P
1	0000000000000*bb*	0000000000000000	0000000000000*bb*	0003000000000000	2	2^{-4}
2	0003000000000000	0003000000000000	0000000000000000	0000000000000000	0	1
3	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
4	0000000000000000	00000*c*0000000000	00000*c*0000000000	0000000004000000	1	2^{-3}
5	0000000004000000	0000000000000000	0000000004000000	0000004000000040	1	2^{-2}
6	0000004000000040	0000003000000000	0000007000000040	0000000200000202	2	2^{-4}
7	0000000200000202	0000000000000000	0000000200000202	0000010500000105	3	2^{-6}
8	0000010500000105	00000000*c*0000000	00000105*c*0000105			1

$15-16$ serve as the round before and after the distinguisher respectively (E_b and E_f)[4].

All the differentials used in E_0 have the same input difference $\alpha = e_{0,1,3,4,5,7}$ and they all work with the key difference $\Delta K^{12} = e_{118,119}$. There are at least 343 such characteristics with varying differences at the beginning of the seventh round: there exist one characteristics of probability $p = 2^{-19}$, 18 characteristics of probability $p = 2^{-20}$, 108 characteristics of probability $p = 2^{-21}$ and 216 characteristics of probability $p = 2^{-22}$. Therefore the overall probability for E_0 is $\hat{p} = \sqrt{1 \cdot (2^{-19})^2 + 18 \cdot (2^{-20})^2 + 108 \cdot (2^{-21})^2 + 216 \cdot (2^{-22})^2} \approx 2^{-17}$. Table 3 shows one of the characteristics used for E_0.

Given the α difference, the number of active S-boxes in E_b which lead to an α difference in round 1 can be found by applying the inverse permutation to α difference. This leads to six active S-boxes in the first round with varying output differences after the substitution layer. These six S-boxes are used to create the α difference before the core function. Since the output difference is known for all active S-boxes in round 0, namely 1_x, not all of the input differences are possible. The number of possible input differences is only $2^{15.5}$ instead of 2^{24}.

For the second subcipher E_1, we use the fixed output difference $\delta = e_{11,15}$ under the key difference $\Delta K^{13} = e_{117,121}$. The most efficient characteristic is provided in Table 4 with probability $p = 2^{-12}$. As it can be seen from the table, the overall probability for E_1 is $\hat{q} \approx 2^{-12}$. Thus, the probability of the related-key rectangle distinguisher is given by $Pr = 2^{-64}\hat{p}^2\hat{q}^2 \approx 2^{-122}$. Similarly, the subcipher E_f after the core function can be defined by letting δ propagate. In round 15, there exist two active S-boxes with input differences 8_x each leading to six output differences and these outputs are diffused to six different S-boxes after key addition in round 16 which produce $2^{21.22}$ possible output differences in total out of 2^{24}.

To attack 17 round PRESENT, we request 2^{39} structures of 2^{24} plaintexts each. The structures are chosen in such a way that each structure varies all over the possible inputs to the active S-boxes in E_b, while the differences for the other S-boxes are kept zero. Our aim is to get an α difference at the beginning of E_0. This technique for choosing plaintexts lets us 2^{47} pairs in total for each structure in which 2^{23} of them satisfy α difference before E_0. Thus, the total

[4] We exclude the output whitening in our attack.

Table 4. An example of related-key differential used in E_1

r	Output Difference $\Delta(O)$	Key Difference $\Delta(K)$	$\Delta(I) \oplus \Delta(K)$	$P^{-1}(S^{-1}(I \oplus K))$	NAS	P
14	0000000000008800	0000000000008800	0000000000000000	0000000000000000	0	1
13	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
12	0000000000000000	0000000000220000	0000000000220000	0000000000600060	2	2^{-6}
11	0000000000600060	0000000000000000	0000000000600060	0000000008800000	2	2^{-6}
10	0000000008800000	0000000008800000	0000000000000000	0000000000000000	0	1
9	0000000000000000	0000000000000000	0000000000000000	0000000000000000	0	1
8			0000000000000000	0000000000000000		

number of pairs with an α difference before the core function is 2^{62} that produce approximately 2^{124} quartets of which $2^{124} \cdot 2^{-64} \cdot 2^{-58} = 2^2 = 4$ are expected to be *right*. The overall attack works as follows:

1. Generate 2^{39} structures of 2^{24} plaintexts and encrypt each structure of plaintexts with K^1, K^2, K^3 and K^4 to obtain the corresponding pool of ciphertexts C^j where $1 \leq j \leq 4$.
   - This step requires data complexity of 2^{63} chosen plaintexts and time complexity of 2^{65} PRESENT-128-17 encryptions.
2. Generate $2^{24+8+24} = 2^{56}$ counters each of which corresponds to a different key guess in E_b and E_f respectively.
   - Time complexity of this step is 2^{56} memory accesses.
3. Insert 2^{65} ciphertexts (C^1, C^3) and (C^2, C^4) into hash tables (T^1_{13}, T^3_{13}) and (T^2_{24}, T^4_{24}) respectively indexed by 40 (expected inactive) bits. If a collision occurs in the same bins of (T^1_{13}, T^3_{13}) and (T^2_{24}, T^4_{24}), check whether the differences of the collided ciphertexts are one of the $2^{21.22}$ expected ciphertext differences.
   - This step has time complexity of 2^{65} memory accesses from inserting all the ciphertext into hash tables. In the hash tables there exist 2^{40} bins and in each bin we expect to have 2^{23} ciphertexts. Therefore, we can form $(2^{23})^2 = 2^{46}$ pairs where one of the components from $T^1_{13}(T^2_{24})$ and the other is from $T^3_{13}(T^4_{24})$. That makes 2^{86} pairs in total for each pair of tables (T^1_{13}, T^3_{13}) and (T^2_{24}, T^4_{24}). In order to check whether colliding ciphertexts' differences are one of the expected ciphertext differences we have to make 2^{87} memory accesses in total. The number of remaining pairs is $2^{86-2.78} = 2^{83.22}$ for each of (T^1_{13}, T^3_{13}) and (T^2_{24}, T^4_{24}) since the probability of having expected difference in the ciphertexts is $2^{21.22-24} = 2^{-2.78}$.
4. For each surviving pair (C_1, C_3) (and (C_2, C_4)) from the previous step, find the corresponding plaintext pairs (P_1, P_3) (and (P_2, P_4)) from the structures. For each such pair, check whether $P_1 \oplus P_2$ satisfies the required difference in E_b. If this check succeeds, examine the ciphertexts C_3 and C_4 that collided with C_1 and C_2 respectively. If the difference between the corresponding plaintexts P_3 and P_4 also satisfies the required difference in E_b, continue to analyze the quartet $((P_1, P_2), (P_3, P_4))$.

- The probability that $P_1 \oplus P_2$ satisfies the required difference is $2^{15.5-24} = 2^{-8.5}$, if they are in the same structure. So, the probability that the required difference is satisfied is $2^{-8.5-39} = 2^{-47.5}$ under the assumption of uniform distribution of plaintexts and structures. This reduces the number of pairs satisfying the condition in (T^1_{13}, T^3_{13}) to $2^{83.22-47.5} = 2^{35.72}$. Similarly, there exist $2^{35.72}$ pairs in (T^2_{24}, T^4_{24}). Thus, we can form $(2^{35.72})^2 = 2^{71.44}$ quartets satisfying the conditions in E_b and E_f. In order to do this filtering we have to make $2^{84.22}$ memory accesses.

5. For each remaining quartet $((P_1, P_2), (P_3, P_4)), ((C_1, C_2), (C_3, C_4))$ and every possible subkey value (k_b and k_f independently) of E_b and E_f test whether

$$E_{b_{k_b}}(P_1) \oplus E_{b_{k_b'}}(P_2) = E_{b_{k_b''}}(P_3) \oplus E_{b_{k_b'''}}(P_4) = \alpha \quad \text{where } k_b' = k_b \oplus \Delta K^{12} \text{ and } k_b'' = k_b''' \oplus \Delta K^{12},$$

$$E^{-1}_{f_{k_f}}(C_1) \oplus E^{-1}_{f_{k_f''}}(C_3) = E^{-1}_{f_{k_f'}}(C_2) \oplus E^{-1}_{f_{k_f'''}}(P_4) = \delta \quad \text{where } k_f'' = k_f \oplus \Delta K^{13} \text{ and } k_f' = k_f''' \oplus \Delta K^{13} \text{ hold.}$$

   If this is the case, increment the counters that correspond to k_b, k_f.
   - In this step, every surviving quartet is partially encrypted (in E_b) and decrypted (in E_f) independently. As the number of subkeys guessed are more in E_f than in E_b, the overall complexity of this step is $2^{71.44+32} = 2^{103.44}$ memory accesses and half round decryptions (as eight S boxes are affected in total); the latter is equivalent to 2^{99} PRESENT-128-17 evaluations.
6. Output the subkeys whose counters are maximal.
   - This step requires 2^{56} memory accesses.

Since α difference after E_b can be obtained from $2^{15.5}$ input differences in step 5, the probability that the intermediate difference is α before the core function is $2^{-15.5}$ on average. Thus, each subkey is suggested by a quartet with probability 2^{-31}. Similarly, the probability that the difference after the core function has an δ difference is $2^{-21.22}$ on average leading to $2^{-42.44}$ of the subkeys by the quartets. Each of the $2^{71.44}$ quartets that enter step 5 suggests $2^{56-2\times15.5-2\times21.22} = 2^{-17.44}$ subkeys, so the total number of suggested subkeys is about 2^{54}. As there are 2^{56} subkeys, the expected number of times a wrong subkey is suggested is about 2^{-2}. This means that we can find the right subkey or at least discard almost all the wrong subkeys.

Thus, the overall attack has memory complexity of 2^{53} bytes, time complexity of 2^{104} memory accesses and data complexity of 2^{63} chosen plaintexts. The expected number of right quartets is taken to be four.

4 Impossible Differential Attacks on HIGHT

In this section, we introduce improved impossible differential attack on 26-round and related-key impossible differential attack on 31-round HIGHT which utilize

Table 5. 26-Round impossible differential

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_0	?	?	?	?	?	$e_{0,\sim}$	0	0	SK_3	SK_2	SK_1	SK_0
ΔX_1	?	?	?	?	$e_{0,\sim}$	0	0	0	SK_7	SK_6	SK_5	SK_4
ΔX_2	?	?	?	$e_{0,\sim}$	0	0	0	0	SK_{11}	SK_{10}	SK_9	SK_8
ΔX_3	?	?	$e_{0,\sim}$	0	0	0	0	0	SK_{15}	SK_{14}	SK_{13}	SK_{12}
ΔX_4	?	$e_{0,\sim}$	0	0	0	0	0	0	SK_{19}	SK_{18}	SK_{17}	SK_{16}
ΔX_5	$e_{0,\sim}$	0	0	0	0	0	0	0	SK_{23}	SK_{22}	SK_{21}	SK_{20}
ΔX_6	0	0	0	0	0	0	0	$e_{0,\sim}$	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	?	$e_{0,\sim}$	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	?	?	$e_{0,\sim}$	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	?	?	?	$e_{0,\sim}$	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	?	?	?	$e_{0,\sim}$	0	0	0	?	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	?	?	$e_{0,\sim}$	0	0	?	?	?	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	?	$e_{0,\sim}$	0	?	?	?	?	?	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	$e_{0,\sim}$	?	?	?	?	?	?	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{13}	$e_{\bar{0},\sim}$	80_x	?	?	?	?	?	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	80_x	0	?	?	?	?	?	$e_{0,\sim}$	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	0	0	?	?	?	?	$e_{0,\sim}$	80_x	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	0	0	?	?	?	$e_{0,\sim}$	80_x	0	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	0	0	?	?	$e_{0,\sim}$	80_x	0	0	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	0	0	?	$e_{2,\sim}$	80_x	0	0	0	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	0	0	$e_{2,\sim}$	80_x	0	0	0	0	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	0	0	80_x	0	0	0	0	0	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	0	80_x	0	0	0	0	0	0	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	80_x	0	0	0	0	0	0	$e_{0,\sim}$	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	0	0	0	0	0	?	$e_{0,\sim}$	80_x	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	0	0	0	?	?	$e_{0,\sim}$	80_x	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	?	?	?	$e_{0,\sim}$	80_x	0	0	SK_{103}	SK_{102}	SK_{101}	SK_{100}
ΔX_{26}	?	?	?	$e_{\bar{0},\sim}$	80_x	0	0	?	WK_7	WK_6	WK_5	WK_4
FT	?	?	?	?	$e_{\bar{0},\sim}$	80_x	0	0				

16-round impossible differential and 22-round related-key impossible differential characteristics respectively. For impossible differential attack on 26-round HIGHT, we use a similar characteristic as in [19] which enables us to attack 26-round of HIGHT with a lower complexity. However, we use better characteristic for related-key impossible differential attack.

The process of both attacks is similar. First, a data collection part is processed for the generation of necessary plaintext-ciphertext pairs. Then, to guarantee the impossible differential characteristic, pairs are filtered by checking the conditions at each intermediate rounds. At the end, the guessed key is eliminated if any one of the remaining pairs satisfies the impossible differential characteristic.

4.1 Impossible Differential Attack on HIGHT-26

We use the following 16-round impossible differential which is also given in Table 5 in detail:

$$(e_{0,\sim}, 0, 0, 0, 0, 0, 0, 0) \nrightarrow (0, 80_x, 0, 0, 0, 0, 0, 0)$$

In Table 5, the contradictory differences and the guessed subkey bytes in the attack are labeled with gray background. The differences used here are considered with respect to XOR operation and shown as hexadecimal. The propagation of the differences can easily be checked by the properties of addition and linear subround functions F_i. Contradiction is shown by miss in the middle manner at values $X_{13,7}$. This attack covers the rounds 0 - 25 and excludes the input whitening as done in [19]. Overall attack on 26-round HIGHT works as follows:

1.) Data Collection

(i) Choose 2^{13} structures of 2^{48} plaintexts each where the bytes $(1, 0)$ have fixed values, bytes $(7, 6, 5, 4, 3)$ and most significant 7 bits of the byte (2) take all possible values.
  – Such a structure of plaintexts propose 2^{94} plaintext pairs and so we get 2^{107} pairs in total.

(ii) Obtain all the ciphertexts C^i of the plaintexts P^i. Choose only the ciphertext pairs satisfying the difference $(?, ?, ?, ?, e_{\bar{0},\sim}, 80_x, 0, 0)$.
  – This step can be done by inserting all the ciphertexts into a hash table indexed by expected inactive bits and choosing the colliding pairs which satisfy the required difference. There is a 25-bit filtering condition over the ciphertext pairs. Therefore, 2^{82} pairs remain.

2.) Filtering and Key Elimination

We have 28 similar steps given in Table 6 to reach impossible differential characteristic and eliminate the wrong key values.

Let us look at the first step as an example. Guess MK_3 and partially encrypt every plaintext pairs by using SK_3 to obtain $(7, 0)$-th bytes of X_1 (The relation between the MK values and SK values are given in the Table 9). The expected difference for the $(7, 0)$-th bytes is $(?, 0)$ which comes up with an eight-bit condition. Therefore, the number of total pairs is decreased to $2^{82-8} = 2^{74}$ after this step. In this step, we partially encrypt 2^{82} pairs with the guessed eight-bit of the secret key. Each partial encryption is equivalent to 1/4th of a round of HIGHT and the overall attack is done on 26 rounds. Thus, the complexity of this step is $2 \cdot 2^{82} \cdot 2^8 \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{84.30}$ 26-round HIGHT encryptions. Remaining steps given in Table 6 follow similarly. Moreover, if the secret key byte MK is already guessed and known for the required subkey SK, it is directly used and since most of the previous conditions are preserved for the next rounds there does not exist too much conditions on the evaluation process of intermediate rounds.

In step 28, if a pair satisfies the impossible differential characteristic, we eliminate that guessed key. Since there is an eight-bit condition, every pair eliminates 2^{-8} of the keys. Therefore after the first pair, there remain $2^{112} - 2^{104} = 2^{112} \cdot (1 - 2^{-8})$ keys. After the second pair, it is expected to have $2^{112} \cdot (1 - 2^{-8}) - 2^{112} \cdot (1 - 2^{-8}) \cdot 2^{-8} = 2^{112} \cdot (1 - 2^{-8})^2$ remaining keys. Following that manner, after the last pair, we have $2^{112}(1 - 2^{-8})^{2^{11}} \approx 2^{100.46}$ remaining keys. Complexity of this step is $2 \cdot 2^{112} \left\{1 + (1 - 2^{-8}) + \ldots + (1 - 2^{-8})^{2^{11}-1}\right\} \cdot \frac{1}{4} \cdot \frac{1}{26} \approx 2^{114.30}$ HIGHT encryptions.

3.) Final Step

For every recorded 112 bit key at the end of Step 28, we obtain the remaining 16 bits and the original key itself with exhaustive search by checking over two plaintext-ciphertext pairs. The probability that a wrong key is suggested is approximately $2^{-64\times2} = 2^{-128}$. So, the expected number of wrong keys is about $2^{-128} \cdot 2^{116.46} = 2^{-11.54}$. Thus, it is very likely that we can find the correct key.

Table 6. 26-Round impossible differential attack

	Guess Key Byte	Use	Obtain	Check Difference	Condition (In terms of bits)	Remaining Pairs	Time Complexity(HE)
1	MK_3	SK_3	$(7,0)$ of X_1	$(?,0)$	8	2^{74}	$\approx 2^{84.30}$
2	MK_1	WK_7, SK_{103}	$(7,6)$ of X_{25}	$(0,?)$	8	2^{66}	$\approx 2^{84.30}$
3	MK_2	SK_2	$(6,5)$ of X_1	-	-	2^{66}	$\approx 2^{84.30}$
4	MK_7	SK_7	$(7,0)$ of X_2	$(?,0)$	8	2^{58}	$\approx 2^{92.30}$
5	MK_0	WK_6, SK_{102}	$(5,4)$ of X_{25}	-	-	2^{58}	$\approx 2^{92.30}$
6	MK_4	SK_{98}	$(5,4)$ of X_{24}	$(0,?)$	8	2^{50}	$\approx 2^{100.30}$
7	-	WK_5, SK_{101}	$(3,2)$ of X_{25}	-	-	2^{50}	$\approx 2^{92.30}$
8	-	SK_{97}	$(3,2)$ of X_{24}	-	-	2^{50}	$\approx 2^{92.30}$
9	MK_8	SK_{93}	$(3,2)$ of X_{23}	$(0,?)$	8	2^{42}	$\approx 2^{100.30}$
10	-	SK_1	$(4,3)$ of X_1	-	-	2^{42}	$\approx 2^{92.30}$
11	MK_6	SK_6	$(6,5)$ of X_2	-	-	2^{42}	$\approx 2^{100.30}$
12	MK_{11}	SK_{11}	$(7,0)$ of X_3	$(?,0)$	8	2^{34}	$\approx 2^{108.30}$
13	-	WK_4, SK_{100}	$(1,0)$ of X_{25}	-	-	2^{34}	$\approx 2^{100.30}$
14	-	SK_{96}	$(1,0)$ of X_{24}	-	-	2^{34}	$\approx 2^{100.30}$
15	MK_{15}	SK_{92}	$(1,0)$ of X_{23}	-	-	2^{34}	$\approx 2^{108.30}$
16	-	SK_{88}	$(1,0)$ of X_{22}	$(0, e_{0,\sim})$	8	2^{26}	$\approx 2^{108.30}$
17	-	SK_0	$(2,1)$ of X_1	-	-	2^{26}	$\approx 2^{100.30}$
18	MK_5	SK_5	$(4,3)$ of X_2	-	-	2^{26}	$\approx 2^{108.30}$
19	MK_{10}	SK_{10}	$(6,5)$ of X_3	-	-	2^{26}	$\approx 2^{116.30}$
20	-	SK_{15}	$(7,0)$ of X_4	$(?,0)$	8	2^{18}	$\approx 2^{116.30}$
21	-	SK_{99}	$(7,6)$ of X_{24}	-	-	2^{18}	$\approx 2^{108.30}$
22	-	SK_{95}	$(7,6)$ of X_{23}	-	-	2^{18}	$\approx 2^{108.30}$
23	MK_{14}	SK_{91}	$(7,6)$ of X_{22}	-	-	2^{18}	$\approx 2^{116.30}$
24	-	SK_{87}	$(7,6)$ of X_{21}	$(0, 80_x)$	7	2^{11}	$\approx 2^{116.30}$
25	-	SK_4	$(2,1)$ of X_2	-	-	2^{11}	$\approx 2^{109.30}$
26	MK_9	SK_9	$(4,3)$ of X_3	-	-	2^{11}	$\approx 2^{117.30}$
27	-	SK_{14}	$(6,5)$ of X_4	-	-	2^{11}	$\approx 2^{117.30}$
28	-	SK_{19}	$(7,0)$ of X_5	$(e_{0,\sim}, 0)$	8	-	$\approx 2^{114.30}$

The total complexity of the steps given in Table 6 is $2^{119.35}$. Since we have approximately $2^{100.46}$ remaining keys before the final step, the complexity of the final step is $2^{100.46+16} = 2^{116.46}$. Therefore, the overall complexity of the attack is $2^{119.35} + 2^{116.46} = 2^{119.53}$ 26-round HIGHT evaluations, 2^{61} chosen plaintexts of data and 2^{109} bytes of memory.

4.2 Related-Key Impossible Differential Attack on HIGHT-31

In this section, we introduce our related-key impossible differential attack on 31-round HIGHT which utilizes a new 22-round related-key impossible differential. The differences of this attack from [19] are the used related-key impossible differential and the overall complexity which makes use of a related-key impossible differential of three more rounds. We use the following 22-round impossible differential which is given in Table 7 in detail:

$$(0,0,0,0,0,0,80_x,0) \nrightarrow (0,0,0,80_x,0,0,0,0)$$

The related-key impossible differential occurs by using the key difference $(\Delta MK_{15}, \Delta MK_{14}, \ldots, \Delta MK_0) = (80_x, 0, \ldots, 0)$. The contradiction occurs at values $X_{17,3}$ and can be shown similarly by miss in the middle manner which is given in Table 7 where the contradictory differences and the subkey bytes having nonzero differences are shown with gray background. The related-key impossible differential was found by imposing the difference 80_x to all 16 key bytes and observing the impossibility at the differentials. It can be concluded that the best

Table 7. 31-Round related-key impossible differential

	$\Delta X_{i,7}$	$\Delta X_{i,6}$	$\Delta X_{i,5}$	$\Delta X_{i,4}$	$\Delta X_{i,3}$	$\Delta X_{i,2}$	$\Delta X_{i,1}$	$\Delta X_{i,0}$	Subkeys			
ΔX_0	?	?	?	$e_{0,\sim}$	80_x	0	?	?	SK_3	SK_2	SK_1	SK_0
ΔX_1	?	?	$e_{0,\sim}$	80_x	0	0	?	?	SK_7	SK_6	SK_5	SK_4
ΔX_2	?	$e_{0,\sim}$	80_x	0	0	0	?	?	SK_{11}	SK_{10}	SK_9	SK_8
ΔX_3	$e_{0,\sim}$	80_x	0	0	0	0	?	?	SK_{15}	SK_{14}	SK_{13}	SK_{12}
ΔX_4	80_x	0	0	0	0	0	?	$e_{2,\sim}$	SK_{19}	SK_{18}	SK_{17}	SK_{16}
ΔX_5	0	0	0	0	0	0	$e_{2,\sim}$	80_x	SK_{23}	SK_{22}	SK_{21}	SK_{20}
ΔX_6	0	0	0	0	0	0	80_x	0	SK_{27}	SK_{26}	SK_{25}	SK_{24}
ΔX_7	0	0	0	0	0	0	0	0	SK_{31}	SK_{30}	SK_{29}	SK_{28}
ΔX_8	0	0	0	0	0	0	0	0	SK_{35}	SK_{34}	SK_{33}	SK_{32}
ΔX_9	0	0	0	0	0	0	0	0	SK_{39}	SK_{38}	SK_{37}	SK_{36}
ΔX_{10}	0	0	0	0	0	0	0	0	SK_{43}	SK_{42}	SK_{41}	SK_{40}
ΔX_{11}	0	0	0	80_x	0	0	0	0	SK_{47}	SK_{46}	SK_{45}	SK_{44}
ΔX_{12}	0	$e_{2,\sim}$	80_x	0	0	0	0	0	SK_{51}	SK_{50}	SK_{49}	SK_{48}
ΔX_{13}	$e_{2,\sim}$	80_x	0	0	0	0	0	?	SK_{55}	SK_{54}	SK_{53}	SK_{52}
ΔX_{14}	80_x	0	0	0	0	?	?	$e_{0,\sim}$	SK_{59}	SK_{58}	SK_{57}	SK_{56}
ΔX_{15}	0	80_x	0	?	?	?	$e_{0,\sim}$	80_x	SK_{63}	SK_{62}	SK_{61}	SK_{60}
ΔX_{16}	80_x	?	?	?	?	$e_{0,\sim}$	80_x	$e_{0,\sim}$	SK_{67}	SK_{66}	SK_{65}	SK_{64}
ΔX_{17}	?	?	?	?	$e_{0,\sim}$	?	$e_{0,\sim}$	?	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{17}	?	?	?	$e_{0,\sim}$	80_x	0	?	?	SK_{71}	SK_{70}	SK_{69}	SK_{68}
ΔX_{18}	?	?	$e_{0,\sim}$	80_x	0	0	?	?	SK_{75}	SK_{74}	SK_{73}	SK_{72}
ΔX_{19}	?	$e_{0,\sim}$	80_x	0	0	0	?	?	SK_{79}	SK_{78}	SK_{77}	SK_{76}
ΔX_{20}	$e_{0,\sim}$	80_x	0	0	0	0	?	?	SK_{83}	SK_{82}	SK_{81}	SK_{80}
ΔX_{21}	80_x	0	0	0	0	0	?	$e_{2,\sim}$	SK_{87}	SK_{86}	SK_{85}	SK_{84}
ΔX_{22}	0	0	0	0	0	0	$e_{2,\sim}$	80_x	SK_{91}	SK_{90}	SK_{89}	SK_{88}
ΔX_{23}	0	0	0	0	0	0	80_x	0	SK_{95}	SK_{94}	SK_{93}	SK_{92}
ΔX_{24}	0	0	0	0	0	0	0	0	SK_{99}	SK_{98}	SK_{97}	SK_{96}
ΔX_{25}	0	0	0	0	0	0	0	0	SK_{103}	SK_{102}	SK_{101}	SK_{100}
ΔX_{26}	0	0	0	0	0	0	0	0	SK_{107}	SK_{106}	SK_{105}	SK_{104}
ΔX_{27}	0	0	0	0	0	0	0	0	SK_{111}	SK_{110}	SK_{109}	SK_{108}
ΔX_{28}	0	0	0	80_x	0	0	0	0	SK_{115}	SK_{114}	SK_{113}	SK_{112}
ΔX_{29}	0	$e_{2,\sim}$	80_x	0	0	0	0	0	SK_{119}	SK_{118}	SK_{117}	SK_{116}
ΔX_{30}	$e_{2,\sim}$	80_x	0	0	0	0	0	?	SK_{123}	SK_{122}	SK_{121}	SK_{120}
ΔX_{31}	80_x	0	0	0	0	?	?	$e_{0,\sim}$	WK_7	WK_6	WK_5	WK_4
FT	$e_{0,\sim}$	80_x	0	0	0	0	?	?				

related-key impossible differential is 22 rounds and it can not be extended by the same technique[5].

Using this related-key impossible differential, we can attack up to 31 rounds of HIGHT. This attack covers the rounds 0 - 30 and excludes the input whitening as done in 26-round impossible differential attack [19]. We use the related-key impossible differential characteristic to attack 31-round of HIGHT which is detailed in Table 7. The attack can be described as follows.

1.) Data Collection

(i) Choose 2^{15} structures of 2^{48} plaintexts each where the byte (2) and the least significant seven bits of the byte (3) are fixed to certain values. The bytes $(7, 6, 5, 1, 0)$ and the most significant seven bits of the byte (7) contain every possible values.
   – There exist 2^{110} plaintext pairs in total which are encrypted by the prescribed difference in the key.

(ii) Obtain all the ciphertexts C^i of the plaintexts P^i encrypted with K^1 and ciphertext $C^{i'}$ of the plaintexts P^i encrypted with K^2 where $K^1 \oplus K^2 = (80_x, 0, \ldots, 0)$. Choose only the ciphertext pairs $(C^i, C^{j'})$ satisfying the difference $(e_{0,\sim}, 80_x, 0, 0, 0, 0, ?, ?)$.

[5] A similar attack with the same complexity can be mounted by imposing the difference 80_x to MK_9 instead of MK_{15}.

Table 8. 31-Round related key impossible differential attack

	Guess Key Byte	Use	Obtain	Check Difference	Condition (In terms of bits)	Remaining Pairs	Time Complexity(HE)
1	MK_0	SK_0	$(2,1)$ of X_1	$(0,?)$	8 bits	2^{61}	$\approx 2^{71.05}$
2	MK_3	SK_3	$(7,0)$ of X_1	-	-	2^{61}	$\approx 2^{71.05}$
3	MK_4	SK_4	$(2,1)$ of X_2	$(0,?)$	8 bits	2^{53}	$\approx 2^{79.05}$
4	MK_9	WK_4, SK_{120}	$(1,0)$ of X_{30}	$(0,?)$	8 bits	2^{45}	$\approx 2^{79.05}$
5	MK_{12}	WK_7, SK_{123}	$(7,6)$ of X_{30}	$(e_{2,\sim}, 80_x)$	2 bits	2^{43}	$\approx 2^{79.05}$
6	-	SK_{119}	$(7,6)$ of X_{29}	$(0, e_{2,\sim})$	8 bits	2^{35}	$\approx 2^{77.05}$
7	MK_2	SK_2	$(6,5)$ of X_1	-	-	2^{35}	$\approx 2^{77.05}$
8	MK_7	SK_7	$(7,0)$ of X_2	-	-	2^{35}	$\approx 2^{85.05}$
9	MK_8	SK_8	$(2,1)$ of X_3	$(0,?)$	8 bits	2^{27}	$\approx 2^{93.05}$
10	MK_{11}	WK_6, SK_{122}	$(5,4)$ of X_{30}	-	-	2^{27}	$\approx 2^{93.05}$
11	-	SK_{118}	$(5,4)$ of X_{29}	-	-	2^{27}	$\approx 2^{93.05}$
12	-	SK_{114}	$(5,4)$ of X_{28}	$(0, 80_x)$	5 bits	2^{22}	$\approx 2^{93.05}$
13	MK_1	SK_1	$(4,3)$ of X_1	-	-	2^{22}	$\approx 2^{96.05}$
14	MK_6	SK_6	$(6,5)$ of X_2	-	-	2^{22}	$\approx 2^{104.05}$
15	-	SK_{11}	$(7,0)$ of X_3	-	-	2^{22}	$\approx 2^{104.05}$
16	-	SK_{12}	$(2,1)$ of X_4	$(0,?)$	8 bits	2^{14}	$\approx 2^{104.05}$
17	MK_5	SK_5	$(4,3)$ of X_2	-	-	2^{14}	$\approx 2^{104.05}$
18	MK_{10}	SK_{10}	$(6,5)$ of X_3	-	-	2^{14}	$\approx 2^{112.05}$
19	MK_{15}	SK_{15}	$(7,0)$ of X_4	$(80_x, e_{2,\sim})$	2 bits	2^{12}	$\approx 2^{120.05}$
20	-	SK_{16}	$(2,1)$ of X_5	$(0, e_{2,\sim})$	8 bits	2^4	$\approx 2^{118.05}$
21	-	SK_9	$(4,3)$ of X_3	-	-	2^4	$\approx 2^{110.05}$
22	MK_{14}	SK_{14}	$(6,5)$ of X_4	-	-	2^4	$\approx 2^{118.05}$
23	-	SK_{19}	$(7,0)$ of X_5	-	-	2^4	$\approx 2^{118.05}$
24	-	SK_{20}	$(2,1)$ of X_6	$(0, 80_x)$	5 bits	-	$\approx 2^{117.89}$

– This step can be done by inserting all the ciphertexts into a hash table indexed by expected inactive bits and choosing the colliding pairs which satisfy the required difference. There is 41-bit filtering condition over the ciphertext pairs. Therefore 2^{69} pairs remain.

2.) Filtering and Key Elimination

We follow the steps as in 26-round attack which is given in Table 8 to reach impossible differential characteristic.

In step 24, if a pair satisfies the impossible differential characteristic, we eliminate the corresponding guessed key. Since there is five-bit condition, each pair eliminates 2^{-5} of the keys and after the last pair there remain $2^{120}(1-2^{-5})^{2^4} \approx 2^{119.27}$ keys.

3.) Final Step:

For every recorded 120 bit key $(MK_0, \ldots, MK_{12}, MK_{14}, MK_{15})$, obtain the remaining eight bits of the key by exhaustive search by checking over three plaintext-ciphertext pairs. The probability that a wrong key is suggested is approximately $2^{-64\times 3} = 2^{-192}$. So, the expected number of wrong keys is about $2^{-192} \cdot 2^{127.27} = 2^{-64.73}$. It is very likely that we can find the correct key.

The total complexity of the steps given in Table 8 is approximately $2^{121.03}$. Since there exists $2^{119.27}$ remaining keys, the complexity of the final step is approximately $2^{127.27}$. Therefore, the complexity of this whole attack is $2^{127.28}$ 31-round HIGHT computations. Even if this attack is not significant compared to the exhaustive search, it is still an important result against HIGHT which reduces the safety margin of HIGHT to one round.

5 Conclusion

In this paper, we present the first related-key cryptanalysis of PRESENT and improve the recent impossible differential attacks on HIGHT. Although our attacks are totally theoretical and have no practical implications, they show new results for both ciphers. The related-key attacks on PRESENT can be seen as a new approach to see the security of the cipher. HIGHT, on the other hand, was shown to be secure up to 19 rounds in the original proposal and recently attacked up to 28 rounds. However, our results show that the security of HIGHT can be further reduced up to one round.

Acknowledgements

The authors would like to thank anonymous reviewers of ACISP 2009 for pointing out recent results on PRESENT and Jonsung Kim for the additional information about HIGHT. We also like to thank Meltem Sönmez Turan, Shahram Khazaei and Martijn Stam for reviewing the previous versions of the paper. The simulations were run on the HPC Cluster of the Department of Computer Engineering, Middle East Technical University.

References

1. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer, New York (2002)
2. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B., Lee, C., Chang, D., Lee, J., Jeong, K., Kim, H., Kim, J., Chee, S.: HIGHT: A New Block Cipher Suitable for Low-Resource Device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
3. South Korea Telecommunications Technology Associations (TTA). 64-bit Block Cipher HIGHT. Standardization Number TTAS.KO-12.0040, December 27 (2006)
4. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
5. Lim, C.H., Korkishko, T.: mCrypton - A Lightweight Block Cipher for Security of Low-Cost RFID Tags and Sensors. In: Song, J.-S., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
6. Standaert, F.-X., Piret, G., Gershenfeld, N., Quisquater, J.-J.: SEA: A Scalable Encryption Algorithm for Small Embedded Applications. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 222–236. Springer, Heidelberg (2006)
7. Robshaw, M.J.B.: Searching for Compact Algorithms: CGEN. In: Nguyên, P.Q. (ed.) VIETCRYPT 2006. LNCS, vol. 4341, pp. 37–49. Springer, Heidelberg (2006)
8. Leander, G., Paar, C., Poschmann, A., Schramm, K.: New Lightweight DES Variants. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 196–210. Springer, Heidelberg (2007)
9. Wheeler, D.J., Needham, R.M.: TEA, a Tiny Encryption Algorithm. In: Preneel, B. (ed.) FSE 1994. LNCS, vol. 1008, pp. 363–366. Springer, Heidelberg (1995)

10. Wheeler, D.J., Needham, R.M.: TEA Extensions (October 1997)
11. The eSTREAM Portfolio. The eSTREAM Project (September 2008), http://www.ecrypt.eu.org/stream/
12. Wang, M.: Differential Cryptanalysis of Reduced-Round PRESENT. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 40–49. Springer, Heidelberg (2008)
13. Albrecht, M., Cid, C.: Algebraic Techniques in Differential Cryptanalysis. To appear in proceedings of FSE (2009)
14. Z'aba, M.R., Raddum, H., Henricksen, M., Dawson, E.: Bit-Pattern Based Integral Attack. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 363–381. Springer, Heidelberg (2008)
15. Collard, B., Standaert, F.-X.: A Statistical Saturation Attack against the Block Cipher PRESENT. To appear in proceedings of CT-RSA (2009)
16. Biryukov, A., Wagner, D.: Slide Attacks. In: Knudsen [26], pp. 245–259
17. Biham, E.: New Types of Cryptanalytic Attacks Using Related Keys. Journal of Cryptology 7(4), 229–246 (1994)
18. Lu, J.: Cryptanalysis of Block Ciphers. PhD thesis, Royal Holloway, University of London, England (July 2008)
19. Lu, J.: Cryptanalysis of Reduced Versions of the HIGHT Block Cipher from CHES 2006. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 11–26. Springer, Heidelberg (2007)
20. Biham, E., Dunkelman, O., Keller, N.: Related-Key Boomerang and Rectangle Attacks. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 507–525. Springer, Heidelberg (2005)
21. Biham, E., Dunkelman, O., Keller, N.: New Combined Attacks on Block Ciphers. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 126–144. Springer, Heidelberg (2005)
22. Biham, E., Dunkelman, O., Keller, N.: The Rectangle Attack - Rectangling the Serpent. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 340–357. Springer, Heidelberg (2001)
23. Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. Journal of Cryptology 18(4), 291–311 (2005)
24. Biham, E., Biryukov, A., Shamir, A.: Miss in the Middle Attacks on IDEA and Khufu. In: Knudsen [26], pp. 124–138
25. Dunkelman, O.: Techniques for Cryptanalysis of Block Ciphers. PhD thesis, Technion, Israel (February 2006)
26. Knudsen, L.R. (ed.): FSE 1999. LNCS, vol. 1636. Springer, Heidelberg (1999)

A A Brief Description of Related-Key Rectangle Attack

Let E denote the encryption function of the attacked cipher which is treated as a cascade of four subciphers as $E = E_f \circ E_1 \circ E_0 \circ E_b$ where the core $E' = E_1 \circ E_0$ is covered by additional rounds called E_b and E_f which are the subciphers before and after the core function respectively to be used in key recovery. The related-key differential is the quadruple $(\Delta X, \Delta Y, \Delta K, p)$ satisfying following under the nonlinear K-keyed function F_K,

$$Pr[F_K(P) \oplus F_{K \oplus \Delta K}(P \oplus \Delta X) = \Delta Y] = p.$$

Here, ΔX and ΔY are the corresponding input and output differences respectively, ΔK the key difference and p is the corresponding probability. We say the related-key differential characteristics $\Delta X \overset{\Delta K}{\rightarrow} \Delta Y$ holds with probability $Pr = p$.

Let $\alpha \xrightarrow{\Delta K^{12}} \beta$ with probability p be the first related-key differential used for E_0 and $\gamma \xrightarrow{\Delta K^{13}} \delta$ with probability q be the second differential used for E_1. Here, (α, β) and (γ, δ) stand for the input-output differences for E_0 and E_1 respectively. We define $\hat{p}$ and $\hat{q}$ as the probabilities related to α and δ respectively as follows: $\hat{p} = \sqrt{\sum_\beta P^2_{\Delta K^{12}}(\alpha \rightarrow \beta)}$ and $\hat{q} = \sqrt{\sum_\gamma P^2_{\Delta K^{13}}(\gamma \rightarrow \delta)}$. Here, β and γ are the possible differences at the end of E_0 and at the beginning of E_1. The key differences $\Delta K^{12} = K^1 \oplus K^2 = K^3 \oplus K^4$ and $\Delta K^{13} = K^1 \oplus K^3 = K^2 \oplus K^4$ are the respective key differences for the subciphers E_0 and E_1. The subciphers before and after the core function are formed according to the α and δ differences. The basic related-key rectangle attack for the core function works as follows:

- Take a randomly chosen plaintext P_1 and form $P_2 = P_1 \oplus \alpha$.
- Obtain the corresponding ciphertexts $C_1 = E'_{K^1}(P_1)$ and $C_2 = E'_{K^2}(P_2)$, where $K^2 = K^1 \oplus \Delta K^{12}$.
- Pick another randomly chosen plaintext P_3 and form $P_4 = P_3 \oplus \alpha$.
- Obtain the corresponding ciphertexts $C_3 = E'_{K^3}(P_3)$ and $C_4 = E'_{K^4}(P_4)$, where $K^3 = K^1 \oplus \Delta K^{13}$ and $K^4 = K^3 \oplus \Delta K^{12}$.
- Check $C_1 \oplus C_3 = C_2 \oplus C_4 = \delta$.

The probability of the rectangle distinguisher is given by $Pr = 2^{-n}\hat{p}^2\hat{q}^2$ where n is the block size. If $\hat{p} \cdot \hat{q}$ is sufficiently high, the rectangle distinguisher works. As shown in [25], if the expected number of right quartets is taken to be four, then there is at least one right quartet in the data set with probability 0.982 since it is a Poisson distribution with expectation of four. Therefore, for this success rate, the number of plaintext pairs needed is $N = 2^{n/2+1}/\hat{p}\hat{q}$ that consist of $2^{n+2}/\hat{p}^2\hat{q}^2$ quartets expecting four right quartets at a time. Further details about rectangle and boomerang attacks can be found in [20,21,22,25].

B Key Schedule Properties of HIGHT

Table 9. Relations of the original key with whitening keys and subkeys

Original Key	Whitening Keys	Subkeys							
MK_{15}	WK_3	SK_{15}	SK_{24}	SK_{41}	SK_{58}	SK_{75}	SK_{92}	SK_{109}	SK_{126}
MK_{14}	WK_2	SK_{14}	SK_{31}	SK_{40}	SK_{57}	SK_{74}	SK_{91}	SK_{108}	SK_{125}
MK_{13}	WK_1	SK_{13}	SK_{30}	SK_{47}	SK_{56}	SK_{73}	SK_{90}	SK_{107}	SK_{124}
MK_{12}	WK_0	SK_{12}	SK_{29}	SK_{46}	SK_{63}	SK_{72}	SK_{89}	SK_{106}	SK_{123}
MK_{11}	WK_{15}	SK_{11}	SK_{28}	SK_{45}	SK_{62}	SK_{79}	SK_{88}	SK_{105}	SK_{122}
MK_{10}	WK_{14}	SK_{10}	SK_{27}	SK_{44}	SK_{61}	SK_{78}	SK_{95}	SK_{104}	SK_{121}
MK_9	WK_{13}	SK_9	SK_{26}	SK_{43}	SK_{60}	SK_{77}	SK_{94}	SK_{111}	SK_{120}
MK_8	WK_{12}	SK_8	SK_{25}	SK_{42}	SK_{59}	SK_{76}	SK_{93}	SK_{110}	SK_{127}
MK_7	WK_{11}	SK_7	SK_{16}	SK_{33}	SK_{50}	SK_{67}	SK_{84}	SK_{101}	SK_{118}
MK_6	WK_{10}	SK_6	SK_{23}	SK_{32}	SK_{49}	SK_{66}	SK_{83}	SK_{100}	SK_{117}
MK_5	WK_9	SK_5	SK_{22}	SK_{39}	SK_{48}	SK_{65}	SK_{82}	SK_{99}	SK_{116}
MK_4	WK_8	SK_4	SK_{21}	SK_{38}	SK_{55}	SK_{64}	SK_{81}	SK_{98}	SK_{115}
MK_3	WK_7	SK_3	SK_{20}	SK_{37}	SK_{54}	SK_{71}	SK_{80}	SK_{97}	SK_{114}
MK_2	WK_6	SK_2	SK_{19}	SK_{36}	SK_{53}	SK_{70}	SK_{87}	SK_{96}	SK_{113}
MK_1	WK_5	SK_1	SK_{18}	SK_{35}	SK_{52}	SK_{69}	SK_{86}	SK_{103}	SK_{112}
MK_0	WK_4	SK_0	SK_{17}	SK_{34}	SK_{51}	SK_{68}	SK_{85}	SK_{102}	SK_{119}

Improved Cryptanalysis of the Common Scrambling Algorithm Stream Cipher

Leonie Simpson[1], Matt Henricksen[2], and Wun-She Yap[2]

[1] Information Security Institute,
Queensland University of Technology,
GPO Box 2434, Brisbane Qld 4001, Australia
lr.simpson@qut.edu.au
[2] Institute for Infocomm Research,
A*STAR, Singapore
{mhenricksen,wsyap}@i2r.a-star.edu.sg

Abstract. This paper provides a fresh analysis of the widely-used Common Scrambling Algorithm stream cipher (CSA-SC). Firstly, a new representation of CSA-SC with a state size of only 89 bits is given, a significant reduction from the 103 bit state of a previous CSA-SC representation. Analysis of this 89-bit representation demonstrates that the basis of a previous guess-and-determine attack is flawed. Correcting this flaw increases the complexity of that attack so that it is worse than exhaustive key search. Although that attack is not feasible, the reduced state size of our representation makes it obvious that CSA-SC is vulnerable to several generic attacks, for which feasible parameters are given.

Keywords: Digital Video Broadcasting, Common Scrambling Algorithm, Stream Cipher, Cryptanalysis.

1 Introduction

The Digital Video Broadcasting Common Scrambling Algorithm (CSA) has been used to encrypt European cable digital television signals since 1994. It was specified by the European Telecommunication Standards Institute (ETSI), and the proprietary algorithm was distributed to cable TV subscribers in the form of a hardware chip. Although some high-level details appear in patents [3], the algorithm has never officially been revealed. In 2002, a software program that implemented the algorithm was released in binary form. This was reverse engineered by hackers who released the details of the CSA algorithm.

The CSA algorithm can be considered as the application of two cipher layers: a stream cipher layer and a block cipher layer. For encryption, the block cipher is applied first, followed by the stream cipher layer. For decryption, the stream cipher layer is applied first, followed by the block cipher layer. Both the block and stream ciphers are initialized using the same 64-bit key. We do not consider

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 108–121, 2009.

the block cipher component within this paper. The stream cipher component is a binary additive stream cipher. We refer to the keystream generator for the stream cipher component of the CSA algorithm as CSA-SC.

The CSA-SC structure comprises two nonlinear Feedback Shift Registers (FSRs), a combiner with memory and an output function. In the patent application [3], the total internal state size of CSA-SC is described as 107 bits. In previous analysis of CSA-SC, Weinmann and Wirt [8] showed that it can be modelled using 103 bits, and note that the period of the keystream produced by CSA-SC is upper bounded by 2^{103}. In this paper, we provide a new representation of CSA-SC that uses only 89 state bits. Consequently the maximum CSA-SC keystream period must be much less than the 2^{103} bits asserted by Weinmann and Wirt [8], with an upper bound of 2^{89} bits. This significant reduction in state size also has implications for the security of CSA-SC.

Weinmann and Wirt [8] presented an analysis of CSA-SC and proposed a guess-and-determine attack with complexity less than 2^{45}, based on their 103 bit CSA-SC representation and predicated on the state cycle structure of one of the FSRs during keystream generation. They claimed that the state cycle structure for this FSR consists of many leading paths and short cycles, with experimental simulation to support this conjecture. However, in examining the cycle structure of the FSR when developing our 89-bit model, we identified that there are no leading paths to cycles, and short cycles were not readily located, implying that the Weinmann-Wirt attack can not work as claimed.

We present our model of the CSA-SC in Section 2. In Section 3, we provide theoretical observations about the state update functions used in CSA-SC. These observations contradict the results presented by Weinmann and Wirt [8], but show that there are security vulnerabilities that can be exploited in cryptanalytic attacks. Section 4 describes an exploration of the FSR state cycle structure. This is motivated by the discrepancy between our observations and the results presented in [8]. Section 5 discusses several possible attacks on CSA-SC. In Section 5.1, the analysis of CSA-SC in [8] is summarized, and the problem with their attack is discussed. In Section 5.2, the vulnerability of CSA-SC to time-memory tradeoff attacks is demonstrated. Section 6 presents some closing remarks on the security of CSA-SC.

2 Specification of the CSA-SC

CSA-SC comprises two FSRs and a combiner with memory. These are denoted FSR-A, FSR-B and FSM-C, respectively. For our representation of CSA-SC, FSR-A and FSR-B each have ten stages, with each stage containing one four-bit word (a nibble). The combiner FSM-C consists of two stages, each containing a nibble, and a single-bit carry. The total state size is 89 bits. Figure 1 shows a high-level view of the relationships between components of the CSA-SC during keystream generation.

During keystream generation, FSR-A is autonomous. The state update function for FSR-A is nonlinear, with the output of the s-box S_A used in calculating

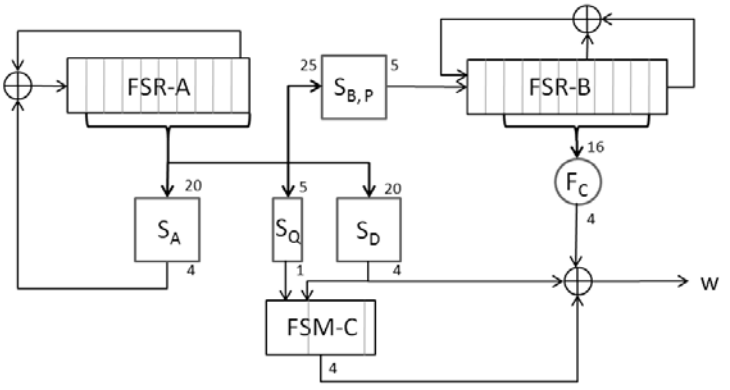


Fig. 1. High-level view of the relationship between components during CSA-SC keystream generation

the next state value. Nonlinear outputs from FSR-A are also used as input to the update functions of both FSR-B and FSM-C, by providing all of the input to s-boxes S_B, S_D, S_P, and S_Q. The outputs of S_B and S_P are used to modify FSR-B, and the outputs of S_D and S_Q are used to modify FSM-C. The specific state update functions for each component and the keystream output function are described in greater detail in Section 2.1.

The notation used in this paper is as follows. Let A^t represent the contents of FSR-A at time t. Then $A^t_{i,j}$ represents bit j of the ith stage of FSR-A at time t, where $i \in \{0, 1, \ldots, 9\}$ and $j \in \{0, 1, 2, 3\}$. Similarly, $B^t_{i,j}$ represents bit j of the ith stage of FSR-B at time t. At time t, the contents of the two four-bit stages of FSM-C are denoted $D^t_{i,j}$, where $i \in \{0, 1\}$ and $j \in \{0, 1, 2, 3\}$, and the contents of the one-bit carry are denoted c^t. According to ETSI conventions, the most significant bit of a stage is denoted by index 0. Binary addition and modular addition in $\mathbb{Z}_{2^4}$ are represented by $\oplus$ and $\boxplus$ operators, respectively. ROL_x represents word rotation to the left by x bits. $c||D$ represents the concatenation of bit c and word D. For example, $0||1001 = 01001$.

2.1 Generating Keystream

When the keystream generator is clocked at time t, FSR-A, FSR-B and FSM-C are simultaneously clocked. FSR-A is autonomous, and contributes to the state update functions of FSR-B and FSM-C. Nonlinear combinations of values stored in FSR-A stages, obtained through the use of various s-boxes, are used in the state update functions of all components. S-boxes S_A, S_B and S_D each take twenty-bit inputs from FSR-A and provide 4-bit outputs. S-boxes S_P and S_Q take 5 bits of input from FSR-A, and each produce a 1-bit output. Specific details regarding the s-boxes are contained in Appendix A. The state update functions for the CSA-SC components are as follows:

$$\begin{aligned} A_i^t &= A_{i-1}^{t-1} && 1 \le i \le 9 \\ A_0^t &= A_9^{t-1} \oplus S_A(A^{t-1}) \end{aligned}$$

$$\begin{aligned} B_i^t &= B_{i-1}^{t-1} && 1 \le i \le 9 \\ B_0^t &= ROL_{S_P(A^{t-1})}(B_6^{t-1} \oplus B_9^{t-1} \oplus S_B(A^{t-1})) \end{aligned}$$

$$\begin{aligned} D_1^t &= D_0^{t-1} \\ c^t || D_0^t &= \begin{cases} c^{t-1} || D_1^{t-1} & \text{if } S_Q(A^{t-1}) = 0 \\ S_D(A^{t-1}) \boxplus D_1^{t-1} \boxplus c^{t-1} & \text{if } S_Q(A^{t-1}) = 1 \end{cases} \end{aligned}$$

The keystream is produced as a series of 2-bit words. The contents of FSR-A, FSR-B and FSM-C all contribute to the keystream output word z^t. To facilitate effective cryptanalysis, we consider the formation of z^t based on an intermediate word, denoted w^t. The 4-bit intermediate word w^t and the 2-bit keystream output z^t are obtained as follows:

$$\begin{aligned} w^t &= S_D(A^{t-1}) \oplus F_C(B^{t-1}) \oplus D_1^{t-1} \\ z^t &= f(w^t \gg 2) || f(w^t \bmod 2^2) \end{aligned}$$

where

$$f(x) = \begin{cases} 0 \text{ if } x = 0 \text{ or } x = 3 \\ 1 \text{ if } x = 1 \text{ or } x = 2 \end{cases}$$

The function F_C produces four bits of output, with each output bit formed from a linear combination of four bits from FSR-B. Specifically, $F_C(B^t) = (B_{2,0}^t \oplus B_{5,1}^t \oplus B_{6,2}^t \oplus B_{8,3}^t) || (B_{5,0}^t \oplus B_{7,1}^t \oplus B_{2,3}^t \oplus B_{3,2}^t) || (B_{4,3}^t \oplus B_{7,2}^t \oplus B_{3,0}^t \oplus B_{4,1}^t) || (B_{8,2}^t \oplus B_{5,3}^t \oplus B_{2,1}^t \oplus B_{7,0}^t)$.

2.2 A Note on Previous Representations

In the DVB patent [3], the CSA-SC algorithm is defined as having a state of 107 bits. This representation facilitates an efficient hardware implementation. The CSA-SC representation of Weinmann and Wirt [8] reduced the state size from 107 to 103 bits.

Our representation obtains a further reduction in the state size to 89 bits, by removing the 4-bit memories X, Y, and Z and one-bit memories p and q from the representation used by Weinmann and Wirt. These memories hold the outputs of s-boxes at time t, and are used in the state update function at time $t+1$ to form feedback. In this representation, none of the s-boxes use the final stage of FSR-A, A_9. When the state update function is applied to FSR-A, the contents of $A_{0..8}^t$ are shifted to become $A_{1..9}^{t+1}$. All of the values required to calculate the feedback at time t remain in the FSR at time $t+1$, so the memories, while useful in constructing efficient hardware, are not required in an equivalent representation of the shift register. In the equivalent representation, the indices of the stages used as inputs to the s-boxes must be incremented by one.

Note that in the original representation, the initial value of all memories is zero. For our representation, this necessitates a special case for the first clock of the key initialization process where the output of the s-boxes must be treated as zero irrespective of their inputs. Although this may not be the most efficient hardware implementation, it permits a cryptographically equivalent representation of CSA-SC using only 89 bits.

Neither our work nor that of Weinmann and Wirt considers the key initialization during cryptanalysis, so we omit the details of the initialisation process in this paper. There are several errors in the specification given in the work of Weinmann and Wirt [8], which we correct in Appendix A of this paper.

3 Some Observations on CSA-SC

In this section, we make some observations regarding CSA-SC. Firstly, in Section 3.1, we show that during keystream generation the state update functions of both FSR-A and FSR-B are invertible. That is, given the state of the two FSRs at time $t+1$, the corresponding FSR states at time t can be uniquely determined. This contradicts the claims of leading paths to short cycles made by Weinmann and Wirt [8], motivating our exploration of the FSR-A state cycles presented in Section 4. Secondly, in Section 3.2 we make observations regarding the ratio of the CSA-SC state size to the key size, which indicates a vulnerability to a generic style of attack.

3.1 State Update Functions during Keystream Generation

The state update functions for FSR-A and FSR-B are invertible. As FSR-A is autonomous during keystream generation, but FSR-B is dependant on FSR-A, it is necessary to establish that the state update function for FSR-A is invertible before examining the state update function for FSR-B. Following this, the conditions under which the FSM-C may be inverted are also presented.

The state update function for FSR-A makes use of S_A, a 20×4 s-box. Although S_A itself is not bijective, the FSR-A state update function is nevertheless invertible because (rearranging the state update function in Section 2.1):

$$\begin{aligned} A_i^{t-1} &= A_{i+1}^t \qquad\qquad 0 \le i \le 8 \\ A_9^{t-1} &= A_0^t \oplus S_A(A^{t-1}) \end{aligned}$$

That is, for the inversion, the register contents are shifted back one stage, rather than forward, and the contents of the last stage, A_9, are computed from the contents of A_0^t and the output of S_A at time $t-1$. Now given that $S_A(A^{t-1})$ takes as input 20 bits from A^{t-1}, including the two bits $A_{9,0}^{t-1}$ and $A_{9,1}^{t-1}$, this initially appears to cause a circular dependancy. However, if S_A is considered as the concatenation of four 5-input Boolean functions, then the ouput of each of these functions can be computed individually and the dependency avoided. Table 3 in Appendix A shows the stages of FSR-A which provide the inputs to each of the four Boolean functions. $A_{9,0}^{t-1}$ and $A_{9,1}^{t-1}$ depend upon the input

sets s_3 and s_2, respectively. Only bits in stages 1-6 and 8 of A^{t-1} (equivalent to stages 2-7 and 9 of A^t), are required, and these are already known. $A_{9,2}^{t-1}$ and $A_{9,3}^{t-1}$ depend upon the input sets s_1 and s_0, respectively. These input sets include $A_{9,0}^{t-1}$ and $A_{9,1}^{t-1}$. Therefore, if $A_{9,0}^{t-1}$ and $A_{9,1}^{t-1}$ have been calculated, then $A_{9,2}^{t-1}$ and $A_{9,3}^{t-1}$ can also be calculated, obtaining the state A^{t-1} in its entirety. As the state update function for FSR-A is invertible, if the state of FSR-A is known at time t, then all future and previous states of FSR-A during keystream generation can be easily calculated.

It follows from the invertibility of FSR-A and the use of only a linear combination of the stages of FSR-B in the state update function for FSR-B that the state update function of FSR-B can also be inverted, once A^{t-1} is obtained, as follows:

$$\begin{aligned} B_i^{t-1} &= B_{i+1}^t & 0 \le i \le 8 \\ B_9^{t-1} &= ROL_{3-S_P(A^{t-1})}(B_0^t) \oplus B_6^{t-1} \oplus S_B(A^{t-1}) \\ &= ROL_{3-S_P(A^{t-1})}(B_0^t) \oplus B_7^t \oplus S_B(A^{t-1}) \end{aligned}$$

That is, given the internal state of FSR-A and FSR-B at time t during keystream generation, all previous and future state values for these two feedback shift registers can be calculated.

The final component to consider is FSM-C. From the FSM-C state update function, $D_0^{t-1} = D_1^t$. However, the value of $c^{t-1}||D_1^{t-1}$ is dependent on the values of c^t, D_0^t and A^{t-1}. When $S_Q(A^{t-1}) = 0$, then given D^t and c^t, it is obvious that $c^{t-1}||D_1^{t-1} = c^t||D_0^t$. However, when $S_Q(A^{t-1}) = 1$, the calculation of $c^{t-1}||D_1^{t-1}$ is more complex, due to the use of integer addition rather than XOR.

Consider the case where $S_Q(A^{t-1}) = 1$. As $c^t||D_0^t = S_D(A^{t-1}) \boxplus D_1^{t-1} \boxplus c^{t-1}$, clearly $D_1^{t-1} \boxplus c^{t-1} = c^t||D_0^t - S_D(A^{t-1})$, where the addition and subtraction are integer rather than bitwise operations. There are two possible values for c^{t-1}, so this provides two possible values for $c^{t-1}||D_1^{t-1}$.

The intermediate keystream word w^t is given by $w^t = S_D(A^{t-1}) \oplus F_C(B^{t-1}) \oplus D_1^{t-1}$, and w^t, $S_D(A^{t-1})$ and $F_C(B^{t-1})$ are all known, but D_1^{t-1} is unknown. However, the sum $D_1^{t-1} \boxplus c^{t-1}$ is known. The contribution of c^{t-1} to $D_1^{t-1} \boxplus c^{t-1}$ is in the least significant bit, but as the addition is integer addition, this raises the possibility of carry to the next bit position, and so on. That is, if $c^{t-1} = 1$ and the least significant bit of $D_1^{t-1} = 1$, then the least significant bit of the integer sum will be 0, and the influence of c^{t-1} is carried to the next position. However, where the least significant bit of the integer sum is 1 (that is, the sum is an odd value), clearly the value of c^{t-1} and the value of the least significant bit of D_1^{t-1} are not the same, so there is no possibility that the influence of c^{t-1} extends beyond that least significant bit position. The 2-bit keystream output z^t is useful in discovering whether the two least significant bits in D_1^{t-1} are the same (00 or 11) or different (01 or 10). Combining this knowledge with a known odd value of $D_1^{t-1} \boxplus c^{t-1}$ enables a unique value of $c^{t-1}||D_1^{t-1}$ to be determined.

Note that FSM-C is the only section of the CSA-SC internal state where the state cycle mapping may involve branching. If $S_Q(A^{t-1}) = 0$ there is a unique

previous state. However, if $S_Q(A^{t-1}) = 1$ then a unique previous state exists only if $D_1^{t-1} \boxplus c^{t-1} = c^t || D_0^t - S_D(A^{t-1})$ is odd, and known keystream bits can be used to determine this unique state. Otherwise, there are two possible prior states for the combiner.

3.2 The Ratio of State Size to Key Size

The CSA-SC key initialization uses the combination of a 64-bit key and 64-bit IV to populate the 89-bit state in preparation for keystream generation. That is, the initialisation function takes 128 bits of input and produces an 89-bit output: the CSA-SC initial state at the start of keystream generation. Although this paper does not consider the specific details of the initialisation function, clearly there exist multiple key-IV pairs that produce the same internal state at the start of keystream generation, and hence the same keystream. That is, the use of different keys does not guarantee the production of different keystreams. Even for a single 64-bit key, clearly there are multiple IVs for which the initial 40-bit state of FSR-A will be the same. As the nonlinearity of the functions used in CSA-SC is largely determined by the contents of FSR-A, CSA-SC may be vulnerable to divide and conquer attacks which target FSR-A.

The small CSA-SC state size also indicates a potential vulnerability to time-memory-data (TMD) tradeoff attacks. These known-plaintext attacks can be used to identify either the internal state of CSA-SC, or the key. These attacks are discussed in greater detail in Section 5.

4 Exploring the State Cycles of FSR-A

In Section 3.1, the state update function of FSR-A is shown to be invertible. Therefore every state of FSR-A has exactly one previous state and exactly one successor state. Consequently, there is either one cycle of length 2^{40} in the FSR-A state space or there are multiple disjoint cycles. It is possible that some of these may be short cycles. There can be no overlapping cycles, and there are no cycles with leading paths.

Floyd's algorithm [6] can be used to detect cycles. This simple algorithm, when applied to FSR-A of CSA-SC, detects a single cycle using the following steps:

Algorithm FA

1. Initialize two instances A_0 and A_1 of FSR-A with the same initial state s
2. Set counter l to 0.
3. Do
   (a) Clock A_0 once. Increment counter l.
   (b) Clock A_1 twice.
   while state(A_0) is not equal to state(A_1).
4. Output l as the length of the cycle.

Applying Floyd's algorithm to comprehensively map all of the cycles produced by FSR-A using the following algorithm raises a problem in detemining which values of s to select.

Algorithm A

1. Set t to 2^{40}.
2. Do
   (a) Choose unique value of s.
   (b) Invoke Floyd's algorithm (Algorithm FA) for s, which outputs cycle length l for state s.
   (c) Decrement t by the value of l.
   while $t > 0$.

A naive memoryless approach is to begin with $s = 0$ and increment s until $s = 2^{40} - 1$. This which ensures that all cycles are mapped, but the running time of the process, at $O(2^{80})$, makes this infeasible.

A modification to the algorithm, using a time-memory tradeoff, tracks the state values visited (using a one-bit flag per state). During each invocation of Floyd's algorithm, the value s is chosen from the complement of the set of visited states. The running time of this algorithm is $2^{40} \cdot c$, but the storage requirement is $2^{40} \times \log_2 2^{40}$ bits = 2.3 terabytes. In practice, this version of the algorithm must be implemented using hard disks, which have high latency, so that the constant c becomes quite large. The storage requirements can be improved by tracking and storing only "distinguished points". For example, states with an 8-bit prefix consisting of 0 bits may be considered "distinguished". This reduces disk usage by a factor of 2^8. However, the algorithm will not detect any cycles that traverse only non-distinguished points. Since Weinmann and Wirt [8] claim the existence of small cycles, we do not want to take this approach.

A possible compromise is to use the first approach, in which at iteration i, $s_i = i$, but with a slight modification to include early stopping criteria. If Floyd's algorithm traverses over state $j < i$ at iteration i, then the cycle has been visited previously and the algorithm aborts early. Similarly, if the cycle length l is larger than the state space not so far searched, then the algorithm has rediscovered a cycle and aborts.

Given that we know there are no leading paths, the algorithm can be optimized by using a single instance of FSR-A, with a stopping condition that a cycle has been found when the starting point is traversed for the second time.

Algorithm FA-M

1. Initialize an instance A of FSR-A with the initial state s.
2. Set counter l_c to 0, l_{max} to t.
3. Do
   (a) Clock A once.
   (b) Increment counter l_c.
   (c) If $l_c > l_{max}$ then abort.
   while state(A) is not equal to initial state s.

Table 1. Cycles identified in FSR-A State Space

Cycle number	Cycle length	Cycle length (log 2)
1	307,080,986	28.19
2	783,472,466	29.52
3	10,152,192,874	33.24
4	14,199,085,442	33.72
5	36,257,653,742	35.08
6	78,922,935,703	36.20
7	225,691,600,544	37.72
8	308,063,543,688	38.16
9	424,911,536,585	38.63
Total	1,099,289,102,030	39.9997

4. Output l_c as the length of the cycle

The running time of the FA-M algorithm varies depending on the length of the cycles that it finds. The results of our search are shown in Table 1. They were generated using several Intel Core Duo machines. Each core is capable of iterating through 2^{38} states per day.

The identification of nine large cycles in conjunction with the observation that the update function is invertible provides strongly contradictory evidence to the claims of Weinmann and Wirt [8] that 98% of the state space of FSR-A can be partitioned into very short cycles.

5 Cryptanalysis of CSA-SC

Observations made in Section 3.2 indicate CSA-SC may be vulnerable to two common styles of attack. The dependence of other components on the autonomous 40-bit FSR-A for nonlinearity indicates the potential for divide and conquer style attacks which target FSR-A. The ratio of the state size to the key size indicates a vulnerability to a generic style of attack known as Time-Memory Tradeoff (TMTO) attack. The attack in by Weinmann and Wirt [8] targets FSR-A, and is reviewed in Section 5.1. The application of TMTO attacks to CSA-SC is discussed in Section 5.2.

5.1 The Weinmann and Wirt Attack

A guess-and-determine attack on CSA-SC which targets FSR-A is presented by Weinmann and Wirt [8]. The aim of the attack is to recover the internal state of the cipher during keystream generation, when FSR-A is autonomous. The attack complexity is claimed to be less than 2^{45}. We explain the flaw in this attack and show that, when the flaw is corrected, the attack performance is actually worse than exhaustive key search.

The attack is performed in three phases. In the first phase, the attacker guesses 53 bits of state comprising FSR-A, FSM-C and the 4-bit memory X used by

Weinmann and Wirt [8] for their 103-bit CSA-SC representation. Because each output bit of F_C is a linear combination of bits within FSR-B, and these output bits are linearly combined to form the keystream, a system of equations can be formed relating the keystream bits to the unknown contents of FSR-B. The second phase of the attack solves this system of equations using Gaussian elimination. The third phase of the attack comprises consistency checking to establish the veracity of guesses made in the first phase. If the consistency checking fails, the guess in the first phase is considered incorrect and a new guess is made. Otherwise, the attack terminates and the combination of the 53-bit guess and the solution to the equation system is used to recover the initial internal state.

The cost of the first phase is 2^{53} operations. In the second phase, a system of equations containing 60 equations in 40 unknowns is developed, which can be solved using Strassen's algorithin in $2^{17.7}$ operations. The cost of the third phase is negligible. The total complexity of the state recovery attack is therefore around $2^{70.7}$ operations, which is about one hundred times worse than a brute-force key search on the 64-bit key.

Weinmann and Wirt [8] claimed to have identified numerous short cycles produced by the FSR-A feedback function. They performed 10,000 random initializations of the cipher, and found that for 98.4% of cases, those key-IV pairs led to FSR-A state cycles with lengths of between 108 and 121,992. This lead to the assumption that for any key-IV pair, the effective state space for FSR-A is equal to the sum of the lengths of those short cycles, with 98.4% probability. An attacker does not know the key, so must guess FSR-A states from all of the points on all of the short cycles. Ignoring leading paths, this gives a total of 313,169 possibilities. Therefore, Weinmann and Wirt[8] claim that the cost of the first phase is reduced to $2^{19} \times 2^9$, where the second term is the cost of guessing the memories and registers in FSM-C.

The optimisation of the guessing phase of the divide and conquer attack is necessary in order for the attack to be faster than exhaustive search. However, both the theoretical observation in Section 3.1 that no leading paths exist because the FSR-A feedback function is invertible, and the empirical results in Section 4 that demonstrate the FSR-A state cycles form a small number of disjoint large cycles show that the basis for the optimisation is unfounded. Thus the performance of Weinmann and Wirt's attack is worse than exhaustive key search, unless further optimizations are identified.

5.2 Time-Memory Tradeoff Attacks

As the TMTO approach is well known, it is not described here. Instead, we refer the reader to the work of Hong and Sarkar [7] for a description of the phases of TMTO attacks. Our analysis aims to determine the feasibility of applying TMTO attacks to CSA-SC given the constraints on the amount of keystream available to an attacker. The specification for Digital Video Broadcasting indicates that keystream is propagated at a maximum rate of 64 Mb/s, and that rekeying occurs at least every 120 seconds. Therefore, for a single key-IV pair, assume

that an attacker has access to about $D = 2^{33}$ bits of data. We consider possible tradeoffs for two styles of TMTO.

The first style of TMTO attack is one in which the attacker attempts to invert the CSA-SC keystream to recover the internal state. For this style of attack, the attacker must satisfy the time-memory-data tradeoff curve $N^2 = TM^2D^2$ for $T \geq D^2$ and $T < 2^K$, where $N = 2^{89}$ is the total number of states, T is the time taken to execute the attack, and M is the amount of memory required to store precomputed tables. The value $2^K = 2^{64}$ represents the computational effort required to launch a brute force attack. The attacker is unable to make use of all available data since $T \geq D^2$ implies that $T > 2^K$. Reasonable parameters are therefore $D = 2^{25}, T = 2^{50}$ and $M = 2^{39}$, for which the attacker requires around 6 terabytes of disk space. This is feasible by today's standards.

If the key initialization function was invertible, then the recovered internal state could be used to derive the key. However, this does not seem to be the case. Therefore this attack is of limited use, since frequent key-IV rekeying is mandatory within the DVB specification, and this attack must be performed on the keystream segment obtained after each rekeying.

The second style of TMTO attack attempts to invert the CSA-SC keystream to recover the key. For this style of attack, the state size is immaterial. The same time-memory-data tradeoff curve holds, but N now refers to the number of possible key+IV values, rather than the number of possible internal states. Here, $N = 2^{64+64}$. Taking the Dunklemann-Keller approach [4], the attacker prepares different tables for many IVs in the precomputation phase. This removes the restriction that $T \geq D^2$, at the expense of reducing the success rate of the attack if the right IVs are not used. Due to the larger size of N, the computational complexity of this attack is inferior to the first, with one possible parameter set being $D = 2^{48.5}, T = 2^{53}, M = 2^{53}$. However, as this is key recovery rather than state recovery, the stipulation for frequent rekeying does not present a limitation. Therefore, any proposed key recovery attack on CSA-SC with time complexity greater than T=2^{53} should be regarded as unnecessary unless the data and memory requirements are much less than those given for this TMD attack.

6 Discussion and Conclusion

In this paper we provide a new representation of CSA-SC that uses only 89 state bits, a significant reduction over the 107 bits and 103 bits used for previous representations. Theoretical observations about the state update functions of components of CSA-SC contradict the empirical results presented in previous research [8], motivating an exploration of the state cycles for FSR-A.

Our FSR-A state-cycle findings raise doubts about the validity of the optimisation required in order for the divide and conquer attack presented by Weinmann and Wirt [8] to be successful. It appears that the complexity of that state recovery attack is not around 2^{45}, as claimed, but in fact worse than exhaustive key search. Even applying their approach to our representation of CSA-SC,

where the state size is reduced because the memory X is redundant, results in an overall attack complexity of about $2^{66.7}$ operations. This is about thirteen times worse than brute force attack and several orders of magnitude worse than TMTO attacks, although with the memory requirement is less than for the TMTO attacks.

The reduction in the state space obtained in our model indicates that CSA-SC is vulnerable to TMTO attacks. Given a keystream segment produced from a single key-IV pair, a state recovery attack is possible with data, time and memory parameters of $D = 2^{25}, T = 2^{50}$ and $M = 2^{39}$, respectively. Additionally, for an increased data value obtained by taking keystream segments formed from a single key, but possibly multiple known IVs, a key recovery attack is possible, with data, time and memory requirements of $D = 2^{48.5}, T = 2^{53}, M = 2^{53}$, respectively. This application of a generic attack style to CSA-SC shows that CSA-SC is vulnerable to cryptanalytic attack.

The attacks discussed in this paper made no use of the CSA-SC initialisation process. Exploring the initialisation process may reveal weaknesses that will lead to improved attacks on this cipher. Additionally, it may be possible to improve the performance of divide and conquer attacks targeting FSR-A by reducing the complexity of determining whether guessed FSR-A and FSM-C states are correct. This could be accomplished, by using a distinguisher, and only proceeding to solving the system of equations to recover the contents of FSR-B for a correct guess.

We examined the cipher with respect to differential and linear attacks, and to guess-and-determine attacks. Even though the s-boxes are far from optimal with respect to differential and linear attacks, the fact that in each clock cycle, half of the bits in FSR-A are passed through s-boxes makes it difficult to utilize the s-box biases; ie. the diffusion in the register is good. Likewise, this means that a large number of bits must be guessed in order to determine a single nibble in the register, and a straightforward guess-and-determine approach is ineffective in reducing the complexity of an attack below 2^{53} operations, even considering the effective reduced state size.

Although there are generic attacks that apply to CSA-SC, it appears that the attack strategy of Weinmann and Wirt [8] does not succeed in key recovery with better complexity than brute force.

Acknowledgements. Thanks to anonymous referees for their valuable comments. Thanks also to Yian Chee Hoo for his spare computer cycles.

References

1. Anonymous. CSA - known facts and speculations (2003), http://csa.irde.to
2. Bernstein, D.J.: Costs of cryptanalytic hardware, Posting to ECRYPT eSTREAM forum, August 21 (2005), http://www.ecrypt.eu.org/stream/phorum/read.php?1,95,95#msg-95
3. Bewick, S.: Descrambling DVB data according to ETSI common scrambling specificiation. UK Patent Application GB2322995A (1998)

4. Dunkelman, O., Keller, N.: Treatment of the Initial Value in TMTO Attacks. In: SASC 2008: The State of the Art of Stream Ciphers, Lausanne, Switzerland, pp. 249–258 (2008)
5. FFDeCSA 1.0.0 implementation, http://www.dvbsupport.net/download/index.php?act=view&id=129
6. Floyd, R.: Non-deterministic Algorithms. Journal of the Association for Computing 4(14), 636–644 (1967)
7. Hong, J., Sarkar, P.: New Applications of Time Memory Data Tradeoffs. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 353–372. Springer, Heidelberg (2005)
8. Weinmann, R.-P., Wirt, K.: Analysis of the DVB Common Scrambling Algorithm. Communications and Multimedia Security. In: Proceedings of the 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004). Springer, Heidelberg (2004)

A CSA-SC S-Boxes

CSA-SC has not been published by ETSI. Information has been revealed in patents [3] and by reverse engineering of implementations; see, for example, [5]. The specification by Weinmann and Wirt [8] is the best academic description of CSA-SC in the public literature to date but contains multiple errors; including a misprint in the table that specifies inputs from FSR-A into the s-boxes, and incorrect ANFs for the component boolean functions of the s-boxes. We give the (hopefully) correct versions below.

Table 2 describes the s-boxes referred to in Section 2. Each s-box is built from 5-input boolean functions. S-boxes S_A, S_B and S_D are each constructed from four boolean functions, while s-boxes S_P and S_Q are each built from a single boolean function. These boolean functions are denoted $F_i(s_j)$ in the following tables, where i denotes the function index and s_j denotes the jth set of FSR-A positions which provide the five inputs to F_i. The FSR-A positions for the 5 inputs $x_i, 0 \leq i \leq 4$ are given in Table 3. The boolean functions are given in Algebraic Normal Form in Table 4. For example, the most significant output bit of s-box S_A is $F_6(s_3)$, where F_6 is $1+x_0+x_1+x_5+x_0 \cdot x_4 ... + x_0x_1x_2x_3x_4x_5$, and from the definition of s_3 in Table 3, $x_0 = A_{3,3}, x_1 = A_{1,1}, x_2 = A_{2,3}, x_3 = A_{4,2}$ and $x_4 = A_{8,0}$.

Table 2. S-boxes used in our model of CSA-SC

S-box	Input Size (bits)	Output Size (bits)	Output
S_A	20	4	$F_6(s_3)\|\|F_4(s_2)\|\|F_3(s_1)\|\|F_1(s_0)$
S_B	20	4	$F_{10}(s_5)\|\|F_8(s_4)\|\|F_7(s_3)\|\|F_5(s_2)$
S_D	20	4	$F_2(s_1)\|\|F_0(s_0)\|\|F_{11}(s_5)\|\|F_9(s_4)$
S_P	5	1	$F_{13}(s_6)$
S_Q	5	1	$F_{12}(s_6)$

Table 3. Inputs from FSR-A into the S-box boolean functions

Function input	x_0	x_1	x_2	x_3	x_4
s_0	$A_{4,0}$	$A_{1,2}$	$A_{6,1}$	$A_{7,3}$	$A_{9,0}$
s_1	$A_{2,1}$	$A_{3,2}$	$A_{6,3}$	$A_{7,0}$	$A_{9,1}$
s_2	$A_{1,3}$	$A_{2,0}$	$A_{5,1}$	$A_{5,3}$	$A_{6,2}$
s_3	$A_{3,3}$	$A_{1,1}$	$A_{2,3}$	$A_{4,2}$	$A_{8,0}$
s_4	$A_{5,2}$	$A_{4,3}$	$A_{6,0}$	$A_{8,1}$	$A_{9,2}$
s_5	$A_{3,1}$	$A_{4,1}$	$A_{5,0}$	$A_{7,2}$	$A_{9,3}$
s_6	$A_{2,2}$	$A_{3,0}$	$A_{7,1}$	$A_{8,2}$	$A_{8,3}$

Table 4. Algebraic Normal Forms of 5-input Boolean functions $F_i, 0 \leq i < 14$

i	Algebraic Normal Form of F_i
0	$1 + x_4 + x_3 + x_3x_4 + x_2x_4 + x_2x_3 + x_1x_4 + x_1x_3 + x_1x_2 + x_1x_2x_4 + x_1x_2x_3 + x_0 + x_0x_3x_4 + x_0x_2 + x_0x_2x_3 + x_0x_1 + x_0x_1x_3 + x_0x_1x_3x_4 + x_0x_1x_2 + x_0x_1x_2x_3$
1	$x_3 + x_2x_4 + x_1 + x_1x_4 + x_1x_3x_4 + x_0x_4 + x_0x_1 + x_0x_1x_3 + x_0x_1x_2 + x_0x_1x_2x_4$
2	$1+x_4+x_3+x_2x_4+x_2x_3+x_2x_3x_4+x_1+x_0x_2x_3+x_0x_1x_4+x_0x_1x_3+x_0x_1x_3x_4+x_0x_1x_2$
3	$1 + x_3 + x_2 + x_2x_4 + x_1x_3x_4 + x_1x_2x_4 + x_0x_3x_4 + x_0x_2 + x_0x_1 + x_0x_1x_3x_4 + x_0x_1x_2x_4$
4	$1 + x_4 + x_3 + x_2x_4 + x_2x_3 + x_2x_3x_4 + x_1 + x_1x_4 + x_1x_3 + x_1x_3x_4 + x_1x_2 + x_1x_2x_3 + x_0 + x_0x_3 + x_0x_3x_4 + x_0x_2 + x_0x_2x_4 + x_0x_2x_3 + x_0x_2x_3x_4 + x_0x_1x_4 + x_0x_1x_2 + x_0x_1x_2x_3$
5	$x_3 + x_3x_4 + x_2x_4 + x_1 + x_0$
6	$1 + x_4 + x_3x_4 + x_2 + x_2x_3x_4 + x_1 + x_1x_2x_3 + x_0 + x_0x_4 + x_0x_3 + x_0x_2x_3x_4 + x_0x_1 + x_0x_1x_4 + x_0x_1x_3x_4 + x_0x_1x_2 + x_0x_1x_2x_3$
7	$1 + x_3 + x_3x_4 + x_2 + x_1x_4 + x_1x_3x_4 + x_1x_2 + x_0x_4 + x_0x_3 + x_0x_2x_3x_4 + x_0x_1 + x_0x_1x_4 + x_0x_1x_3x_4 + x_0x_1x_2 + x_0x_1x_2x_3$
8	$1+x_4+x_3+x_3x_4+x_2x_4+x_2x_3+x_2x_3x_4+x_1+x_1x_4+x_1x_3x_4+x_1x_2x_4+x_1x_2x_3+x_0x_4 + x_0x_3 + x_0x_2 + x_0x_2x_3 + x_0x_2x_3x_4 + x_0x_1x_4 + x_0x_1x_3 + x_0x_1x_2x_4 + x_0x_1x_2x_3$
9	$x_3x_4 + x_2 + x_2x_4 + x_2x_3x_4 + x_1x_4 + x_1x_3 + x_1x_2x_4 + x_0x_4 + x_0x_2 + x_0x_2x_4 + x_0x_2x_3 + x_0x_2x_3x_4 + x_0x_1 + x_0x_1x_4 + x_0x_1x_3 + x_0x_1x_3x_4$
10	$x_3 + x_2x_4 + x_1x_3x_4 + x_1x_2 + x_1x_2x_4 + x_0 + x_0x_3x_4 + x_0x_1x_4$
11	$x_4+x_2+x_2x_3+x_2x_3x_4+x_1x_3+x_1x_2+x_1x_2x_3+x_0x_3x_4+x_0x_2x_3+x_0x_2x_3x_4+x_0x_1x_3x_4 + x_0x_1x_2x_3$
12	$x_4+x_3+x_3x_4+x_2+x_1+x_1x_3x_4+x_0x_4+x_0x_3x_4+x_0x_2+x_0x_2x_3+x_0x_2x_3x_4+x_0x_1x_3x_4 + x_0x_1x_2x_3$
13	$x_4 + x_3x_4 + x_2 + x_2x_3 + x_2x_3x_4 + x_1 + x_1x_2 + x_0 + x_0x_1x_3 + x_0x_1x_3x_4$

Testing Stream Ciphers by Finding the Longest Substring of a Given Density*

Serdar Boztaş[1], Simon J. Puglisi[2], and Andrew Turpin[2]

[1] School of Mathematical & Geospatial Sciences
[2] School of Computer Science & Information Technology,
RMIT University, Melbourne VIC, Australia
serdar.boztas@ems.rmit.edu.au,
{simon.puglisi,andrew.turpin}@rmit.edu.au

Abstract. Given a string $x[1..n]$ drawn from the alphabet $\{0, 1\}$, and a rational density parameter $0 \leq \theta \leq 1$, this paper considers algorithms for finding the longest substring of x with density θ. That is, if the length of the substring is m, the number of one-bits in the substring is exactly $\theta \times m$. It is surprisingly difficult to devise an algorithm that has worst case time less than the obvious brute-force algorithm's $O(n^2)$. We present three new approaches to reducing the running time, and an algorithm that solves the problem in $O(n \log n)$ expected time.

We then apply the new algorithm, as well as an empirical estimate of the lim-sup and the lim-inf of a centred statistic which is expected to obey a law of the iterated logarithm, to the randomness testing of (a) the output of the BSD function Random, and (b) the output of the stream cipher Dragon. The results for these outputs warrant further study.

1 Introduction

"A method of producing random numbers should not be chosen at random"−Donald Knuth, in [8].

The security of modern information systems depends on the quality of pseudo-random number generators and the "randomness" of numbers they produce. As such, algorithms that test these generators are of interest – they provide some level of assurance. Randomness testing has been investigated by various authors in the past: Knuth [8] has a good overview of such tests, L'Ecuyer [4] has also considered the same problem, from a different point of view. Other references in this area include *The Handbook of Applied Cryptography* [10] (see Chapter 5), Marsaglia's work [9] including his DIEHARD battery of randomness tests, and the recent volume [12] by Neuenschwander. A suite of randomness tests are implemented by the US Government's NIST [13], and another suite of randomness tests called CRYPT-XS are implemented by the Queensland University of Technology(QUT)'s Information Security Institute [15].

* This work is supported by the Australian Research Council.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 122–133, 2009.

In this paper we consider algorithms for an easy to state problem that is useful for randomness testing, but which is deceptively difficult to solve in worst case asymptotic time less than an obvious brute-force solution. We discuss the problem of randomness testing and its application to security further in Section 3 and focus on algorithmics below.

Throughout we consider a string $x[1..n] = x[1]x[2]\ldots x[n]$ of n symbols drawn from a binary alphabet $\Sigma = \{0, 1\}$. We call $x[i..j] = x[i]x[i+1]\ldots x[j], 1 \leq i \leq j \leq n$ a *substring* of x. A substring $x[i..j]$ is a *suffix* of x if $j = n$ and a *prefix* of x if $i = 1$.

The *density* of a string x is the ratio of the number of one-bits in the string to the length of the string. More formally given a string x and $\theta = c/d$, c and d positive integers $c \leq d$, we say a substring $x[i..j]$ is θ-*dense* (or has density θ) if and only if $j - i + 1 = kd$ and $x[i..j]$ contains kc one-bits, for some positive integer k.

Problem 1 (Longest θ-dense substring). *Given a string $x[1..n]$ on $\{0, 1\}$ and $\theta = c/d$, c and d positive integers, with $c \leq d$ and $gcd(c, d) = 1$, find the longest substring of x having density θ. In other words, find the largest k such that there is a substring of x of length kd that has kc one-bits.*

Note that, a string as defined in Problem 1, need not exist, and if it does, the problem is finding it efficiently. Throughout this paper, we use log to denote logarithms to the base 2.

The remainder of this paper is as follows. Section 2 considers Problem 1 and gives a basic solution to which we add several heuristics. We show that one of the heuristics leads to a $O(n \log n)$ expected time solution to the problem. We discuss randomness testing and display experimental results using the new algorithm in in Section 3. Finally, in Section 4 we offer some reflections and future directions.

2 Locating the Longest Substring with Given Density

An $O(n^2)$ worst case solution to Problem 1, finding the longest θ-dense substring, is relatively straightforward.

Firstly we preprocess x so that we can answer rank queries on x in constant time. The answer to a rank query $rank_x(i)$ is the number of one-bits in $x[1..i]$. A simple data structure to support rank queries is an array of integers containing the a count of the number of one-bits in x up to that position in the array. Such an array can be calculated in a single pass over x, requiring $\theta(n)$ time, and the largest element in such an array would be n, requiring at least $\log n$ bits of storage, so total space for the auxiliary data structure would be $O(n \log n)$. A more sophisticated data structure supporting constant time rank queries can be built in $O(n)$ time and requires $n + o(n)$ bits of extra space [6,11,14].

Once the auxiliary data structure is constructed so rank queries can be answered in constant time, the number of bits in substring $x[i..j]$ can be calculated as $rank_x(j) - rank_x(i-1)$, also in $O(1)$ time. Then for each position in the string $i \in [1..n-d+1]$, compute the number of bits in each substring $x[i..i+kd], k \in [1..(n-i+1)/d]$ keeping track of the longest substring containing

```
BruteForce (x[1..n],θ = c/d)
  Set up a rank data structure that computes rank_x(i) in O(1) time.
  Set max ← (0, 0).
  for i ← 1 to n − d + 1 do
      Set kc ← c.
      for j ← i + d − 1 to n in steps of d do
          if (rank_x(j) − rank_x(i − 1)) = kc and (j − i > max.j − max.i) then
              Set max ← (i, j).
          Set kc ← kc + c.
  if max ≠ (0, 0) then output x[max.i..max.j] else no substring found.
```

Fig. 1. Algorithm BruteForce to find the longest substring of $x[1..n]$ that is θ-dense

kc one-bits. For each position we ask at most $O(n/d)$ rank queries, there are at most n positions, so overall the time spent is $O(n^2/d) = O(n^2)$. Figure 1 shows some pseudo code for this approach.

Given that the rank data structure allows a constant time decision on whether a given substring of length kd contains the necessary number of one-bits, the difficulty with Problem 1 seems to be the number of substrings an algorithm must consider, so any reduction in running time must be gained through a reduction in the number of substrings. This observation is consistent with literature describing $O(n)$ algorithms for finding θ-dense substrings with bounded lengths [7]. In the next two subsections we examine two approaches to reducing the number of substrings considered.

2.1 Exploiting θ-Primitives

It would seem intuitive that long θ-dense strings would contain substrings that were themselves θ-dense. If this were true, then an inductive style algorithm that constructs long θ-dense strings by combining smaller θ-dense substrings should reduce the running time of an algorithm to solve Problem 1. In this regard, we observe the following lemma.

Lemma 1. *A θ-dense string x of length kd containing exactly kc one-bits must have at least one substring of length d containing c one-bits.*

Proof. Call a substring of x of length d *under* or *over* if it contains less than or more than c one-bits respectively. We proceed with a proof by contradiction. Assume the Lemma is not true, then every substring of x of length d must be either under or over. If all length d substrings were under, then there would be less than kc one bits in x in total; similarly, if all of the length d substrings were over, then there would be more than kc one bits in x in total. Hence, there must be at least one under and one over substring in x as it is θ-dense. Moreover, there must be a point in the string where an over and under substring overlap by $d - 1$ characters, otherwise all substrings must be under, or all over.

Without loss of generality say that position in x where the overlap between an under and an over substring occurs is $i{+}1$; that is, $x[i..i{+}d]$ is under and $x[i{+}1..d{+}1]$

is over. The substring $x[i..i+d]$ can have at most $c-1$ one-bits, as it is under, hence the substring $x[i..i+d]x[i+d+1] = x[i..i+d+1]$ can have at most c one-bits. But if $x[i..i+d+1]$ can have at most c one-bits, it is not possible for its suffix $x[i+1..i+d+1]$ to have more than c bits hence $x[i+1..d+1]$ cannot be over. Therefore x cannot consist of only substrings of length d that are a mix of under and over. Therefore x contains a substring of length d that contains c one-bits. □

We call a θ-dense string of length d a θ-primitive string.

It would be nice if the θ-primitive substrings of a string could be used as a base for constructing longer θ-dense substrings. That is, beginning with θ-primitives of length d, expand them to θ-dense strings of length $2d$, and so on. Unfortunately if x is of length kd and contains kc one-bits, there is no guarantee that it must contain a substring of length $(k-1)d$ containing $(k-1)c$ bits. For example, if $x = 111000111$ and $d = 3$, $c = 2$, then the entire string ($k = 3$) has density $2/3$, but there is no substring of length 6 containing 4 one-bits ($k = 2$). So while any θ-dense string must contain a θ-primitive of length d containing c one-bits, there is no guarantee that it must contain a substring of length $2d$ containing $2c$ one-bits. This limits the power of Lemma 1 to reduce the number of substrings that must be considered by any algorithm designed to find the longest θ-dense substring of a string below $O(n^2)$.

Interestingly, Lemma 1 can be used to solve the following (weaker) problem in $O(n)$ time.

Problem 2. *Given a string $x[1..n]$ on $\{0,1\}$ and $\theta = c/d$, c and d relatively prime positive integers, $c < d$, determine if x has any substrings with density of one-bits equal to θ.*

Lemma 1 allows us to only check the substrings of length d — if none are present then there can be no longer ones. The substrings of length d can be checked for in $O(n)$ time by sliding a window of size d over x.

Although there is no ready inductive argument that might allow an algorithm to identify all θ-primitive regions, and then use them to construct longer substrings in sub-$O(n^2)$ time, in practice Lemma 1 may provide some useful heuristics that will reduce running time on random strings.

Lemma 2. *Let i_1 and i_2 be the positions of consecutive θ-primitive substrings in x. No θ-dense substring can begin after position i_1 and end before position $i_2 + d - 1$.*

Proof. From Lemma 1 we know that any θ-dense substring must contain a θ-primitive substring. Any substring beginning after i_1 and ending before i_2 will not contain a θ-primitive substring, so cannot be θ-dense. □

If θ-primitives are few and far between, as we might expect in random strings for large d, then this Lemma allows us to discard quite a few substrings from consideration. An algorithm similar to BruteForce, where i and j, the left and right boundaries of possible substrings respectively, move right along x can exploit this. The right boundary, j must be to the right of the closest primitive

```
SkipMisMatch (x[1..n],θ = c/d)
  Set up a rank data structure that computes rank_x(i) in O(1) time.
  for k ← ⌊n/d⌋ downto 1 do
      Set i ← 1, and j ← kd.
      while i ≤ n − kd + 1 do
          Set δ ← |kc − rank_x(j) + rank_x(i − 1)|.
          if δ = 0 then
              Set max ← (i, j) and terminate.
          Set i ← i + δ, and j ← j + δ.
  Output no suitable substring found.
```

Fig. 2. Algorithm SkipMisMatch to find the longest substring of $x[1..n]$ that is θ-dense

to i, if such a primitive exists, allowing j to skip several positions from consideration. For each pair, however, all strings beginning at i and ending at a position greater than j must be checked, so the algorithm still requires $O(n^2)$ time. The worst case is realised when every position in the string is the end of a θ-primitive; for example, $x = 110110110..$ when $\theta = 2/3$.

2.2 Exploiting Mismatches

A reversal of the loops in Figure 1 yields an alternate brute force algorithm that for every possible substring length, every position is checked; rather than one that checks every possible length for every position. This simple reversal of loops does not lower the asymptotic cost of Algorithm BruteForce, but it does allow an optimisation, as captured in the following Lemma.

Lemma 3. *If the number of one-bits, say m, in a substring of length kd beginning at position i, where $1 \leq i \leq n - kd + 1$, $x[i..i + kd - 1]$, is not equal to kc, then there can be no θ-dense substring of length kd beginning in positions $x[i..i + |kc - m|]$.*

Proof. By definition, a θ-dense substring of length kd must contain kc one-bits. If a substring of x of length kd contains only $m < kc$ one-bits, then at least a further $kc - m$ bits must be added to the substring to bring the count of one-bits up to kc. That is, the end of the substring must be increased by $kc - m$ positions. In order to remain of length kd, however, the left boundary of the substring must also be increased by $kc - m$ positions, hence it cannot begin in positions $x[i..i + (kc - m)]$. Similarly, if $m > kc$, then $m - kc$ zero-bits must be added to the end of x to reduce its density to θ. □

This lemma allows us to exploit mismatches in windows of length kd in much the same way that pattern matching algorithms such as KMP or Boyer-Moore skip over mismatching areas of text [3]. Algorithm SkipMisMatch outlines the approach. Note that the order of loops is reversed from Algorithm BruteForce, and k decreases from the maximum possible until some θ-dense substring is found. Because

the algorithm searches from longest to shortest substrings, it can terminate as soon as it has found a θ-dense substring, as there can be no longer such substrings. For each substring considered, if there is an abundance or deficit of one-bits ($\delta \neq 0$), then i and j are stepped by the difference as per Lemma 3.

The worst case running time of Algorithm SkipMisMatch is still $O(n^2)$, but the expected running time can be less, depending on the number and size of the skips arising from Lemma 3.

Lemma 4. *Algorithm* SkipMisMatch *requires* $O(n \log n)$ *expected time.*

Proof. Let us assume that the string x has been generated by a source that emits a one-bit with probability p. For some substring of length kd, the expected number of one-bits is $p \times kd$, and the number of one-bits required for a θ-dense substring is $\theta \times kd$. The expected value of δ, therefore, is $\hat{\delta}_k = |\theta kd - pkd|$. So for each possible k value we expect to skip over $\hat{\delta}_k$ positions for every position examined, so at most $n/\hat{\delta}_k$ positions are examined. Given that examining each position requires $O(1)$ time, The expected time is bounded by

$$\sum_{k=1}^{\lfloor n/d \rfloor} \frac{n - kd + 1}{\hat{\delta}_k} \leq \frac{n}{|\theta - p|d} \sum_{k=1}^{n/d+1} 1/k = O(n \log n),$$

and the proof is complete. □

2.3 Experimental Results on BSD Function Random

In order to confirm that the expected running time of Algorithm SkipMisMatch is sub-$O(n^2)$ we ran some simple timing experiments on strings generated using

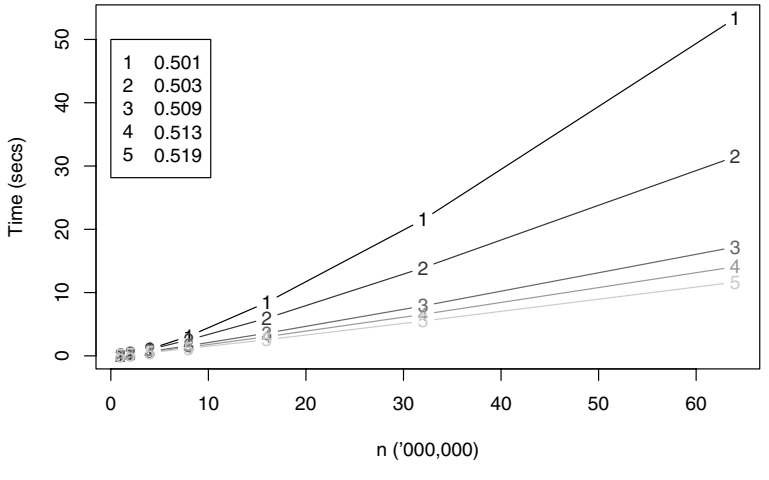


Fig. 3. Mean time taken for Algorithm SkipMisMatch on strings of varying length generated by the BSD function Random (x-axis) when $p = 0.5$. The values in the legend indicate the θ value used for each run.

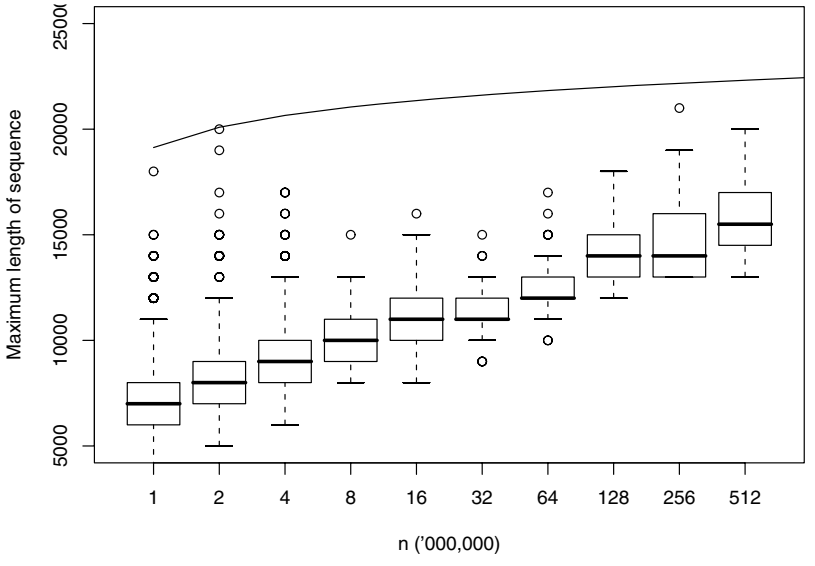


Fig. 4. Length of the longest θ-dense sequence, generated by Random, where $\theta = 0.519$, $p = 0.5$. Boxes show 0.25 and 0.75 quantiles, the black line is the median, and whiskers and dots show extreme values. The solid curve is the bound given by the Strong Law of Large Numbers: $\log(n)/H(0.519, 0.5)$.

the BSD function `random` on a Sun Fire V210 Server with 16GB of RAM running Solaris 10, utilising one of the two 1.34 GHz UltraSPARCIII-i processors. Times reported are the CPU time as reported by the Solaris command line `time` program.

Figure 3 shows the mean running time of Algorithm SkipMisMatch, where the mean is over 20 runs with a different random seed for the random function at each run, for various θ values. The standard deviations of each group of runs were very small, and so are not plotted as error bars. The digits labelling the curves represent data points, and the lines are drawn for clarity. In all cases, the time required increases sub-$O(n^2)$ as n increases. As $|\theta - p|$ increases (curves 1 down to 5) the running time of SkipMisMatch decreases accordingly, as the constant of proportionality in the $O(n \log n)$ expected running time bound includes $1/|\theta - p|$. To examine the effect of varying p, we fixed θ at 0.6, and ran 20 runs for various p values, again using new random seeds for each run. The data is consistent with an expected running time of $O(|\theta - p|^{-1} n \log n)$.

Figure 4 shows some of the lengths of the longest sequences we observed at the output of Random. The plot is consistent with the expectation that the length of the longest sequence is proportional to $\log(n)/H(\theta, p)$, as discussed in detail in the next section.

3 Application to Randomness Testing of Stream Ciphers

The eStream project www.ecrypt.eu.org/stream/ is a recently completed project described as a "multi-year effort running from 2004 to 2008 [which] has identified a portfolio of promising new stream ciphers". An interesting and highly efficient cipher submitted to that project is Dragon [16] designed by researchers at QUT. Dragon has been subjected to extensive randomness testing with no weaknesses identified.

While many randomness tests are currently known and used, it is still of interest to look for further tests, since in cryptography, a cipher is considered suspect as soon as its output fails a single well-designed test. Moreover, recent work has shown that the issue of statistical dependence between commonly used tests is quite complicated. In particular some tests output test results which have a much higher correlation than what would be expected if the tests were statistically independent. See Turan et. al. [17] for more details.

3.1 Testing Dragon Using the Erdös-Rényi Strong Law of Large Numbers

The relevance of Problem 1 to randomness testing comes from the fact that for a random sequence, the statistic computed by Problem 1 has an almost-sure limit as $n \to \infty$. Let $X_1, \ldots, X_n$ be a given sequence of $\{0,1\}-$valued random variables and define $L_n(\theta)$, the length of the "longest run of 1's with density θ" by

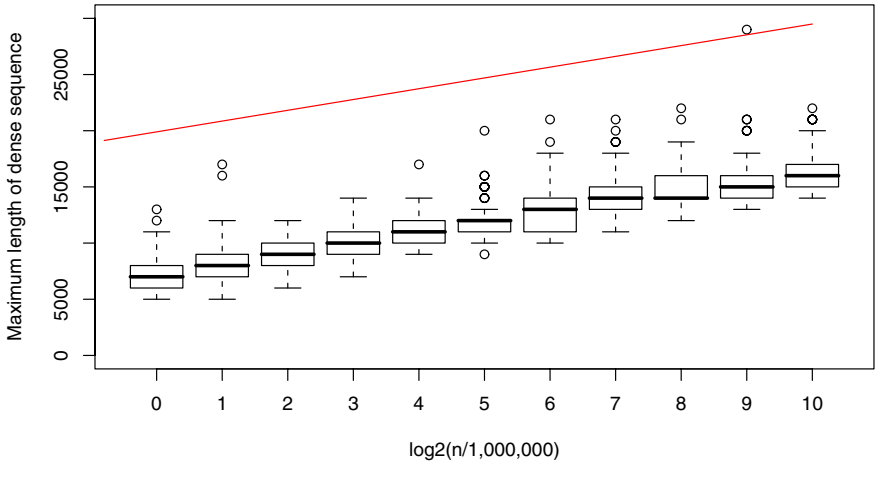


Fig. 5. Length of the longest θ-dense sequence, generated by Dragon, where $\theta = 0.519$, $p = 0.5$. Boxes show 0.25 and 0.75 quantiles, the black line is the median, and whiskers and dots show extreme values. The solid curve is the bound given by the SLLN: $\log(n)/H(0.519, 0.5)$.

$$L_n(\theta) = \max\left\{ t\colon \exists i \in \{0,\ldots,n-t\},\ \theta \le \frac{1}{t}\sum_{k=1}^{t} X_{i+k} \right\}.$$

Erdös and Rényi [5] have proved the following.

Theorem 1. *If $X_1,\ldots,X_n$ are i.i.d. with $Pr[X_i = 1] = p$, for all $\theta \in (p,1]$, we have $L_n(\theta) \to \frac{\log(n)}{H(\theta,p)}$, almost surely, where the binary relative entropy is given by $H(\theta,p) = \theta\log(\theta/p) + (1-\theta)\log((1-\theta)/(1-p))$.*

We now give an intuitive and non-rigorous explanation of what the phrase *converges almost surely* (implicit in the Theorem above) means. The rigorous definition of this measure-theoretic terminology can be found in many statistics textbooks.

If we say that a random quantity T_n *converges in probability* to a constant T, this means that, for the random sequence T_n, for all $\varepsilon > 0$, the quantity $Pr[|T_n - T| > \varepsilon]$ goes to zero as $n \to \infty$. But this probability is averaged over all possible realizations of the random sequence $(T_n)_{n\ge 1}$ so that there can be some realizations for which the convergence above does not hold, albeit for a vanishing fraction as of all possible realizations $n \to \infty$.

If we say that a random quantity T_n *almost surely converges* to a constant T, this means that, for the random sequence T_n, for all $\varepsilon > 0$, the quantity $Pr[|T_n - T| > \varepsilon]$ goes to zero as $n \to \infty$ for essentially all possible realizations of the random sequence (equivalently except for a set of probability zero among all possible realizations). It is, however, possible that the rate of convergence of $Pr[|T_n - T| > \varepsilon]$ to zero depends on the realization.

Note that $\lim_{\theta\to p} H(\theta,p) = 0$, and that's why we take $\theta \in (p,1]$. It is relatively simple to obtain an upper bound on this growth rate by using large deviations, see [2], but much harder to get the lower bound and hence the exact growth rate. Using this almost sure growth rate to test the randomness of a given sequence would essentially proceed as follows (for simplicity, we assume the sequence is unbiased, i.e., $p = 1/2$) which should hold for the output of any good random bit generator.

For $i = 1,2,\ldots,s$ compute the length of the longest substring in $X_1,\ldots,X_{n_i}$ with density $\theta_1,\ldots,\theta_m$. Use the theorem to visually observe whether the growth rate of the length of the longest substring is roughly:

(i) Proportional to $\log(n_i), i = 1,\ldots,s$; and
(ii) Inversely Proportional to $H(\theta_j, 1/2)$ for $j = 1,\ldots,m$.

As a preliminary experiment to exploit Theorem 1 for solving Problem 1, we could assume a value of p, and assume that string x is long enough for the strong law to be accurate, and only search the string for substrings of length $\log(n)/H(\theta,p)$. We have done this for the function Random and displayed the results in Figure 4.

We have also applied this approach as a preliminary test of the output of Dragon. In particular, for keystream output lengths of $2^{25} + 1000k$, with $k =$

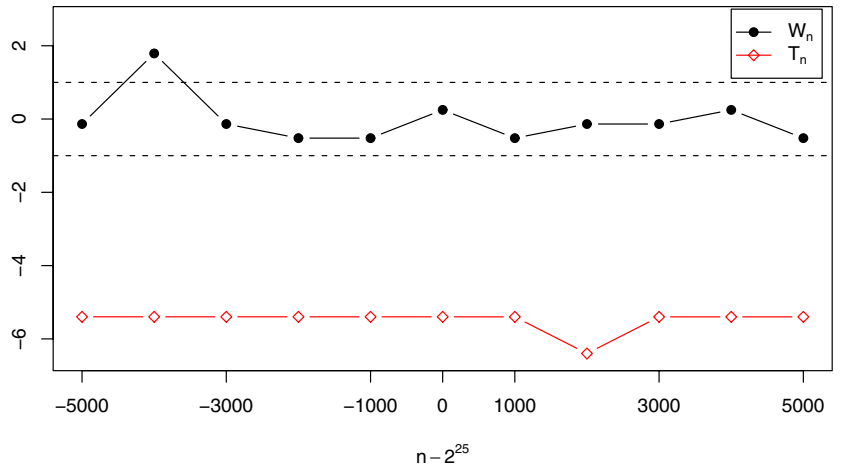


Fig. 6. The values of the quantities T_n defined in (1) (solid circles) and and W_n defined in (2) for the output of Dragon, where the output length ranges from $2^{25} - 5000$ to $2^{25} + 5000$ bits at equal intervals of 1000 bits

$-5, -4, \ldots, +4, +5$, we have generated 100 random keys from /dev/random for each n_k and used the 100 output keystreams thus generated as input to our algorithm computing the length of the longest substring with density $\theta = 0.519$ assuming $p = 1/2$. The results look as expected and are plotted in Figure 5. These results are, of course, *very preliminary, and extensive testing will follow*. Note that we have used a logarithmic plot of the $x-$axis in Figure 5, hence the straight line of the solid curve, when compared to the curved line of the solid curve in Figure 4. When an $x-$axis value of Figure 5 is converted to the corresponding $x-$axis value in Figure 4, the statistical behaviour (in terms of median and dispersion) of $L_n(\theta)$ is essentially the same for Dragon and for the BSD function Random.

3.2 Testing Dragon Using the Law of the Iterated Logarithm

A related statistical strong law, described in [1], has also been used to test Dragon, and is described below.

Let $X_1, X_2, \ldots$ be an i.i.d. sequence of binary random variables with $p = P[X_i = 1] = 1 - P[X_i = 0]$ for all $i \geq 1$, and let $S_{n:t}$ be defined as the maximum number of ones occuring in a window of length t *starting within the first* n *tosses*:

$$S_{n:t} = \max_{1 \leq i \leq n} (X_i + X_{i+1} + \cdots + X_{i+t-1}).$$

Assume that $a \in (p, 1)$ and use a Law of the Iterated Logarithm from [1] to plot the variability of the output of Dragon. Fix $\theta \in (p, 1)$. With $t \stackrel{\triangle}{=} t(n) = \lfloor log(n)/H(\theta, p) \rfloor$, and the "odds ratio" $r \stackrel{\triangle}{=} r(\theta, p) =$

$= p(1-\theta)/(\theta(1-p))$, as $n \to \infty$ define centering constants $b(n,t) \triangleq b(n,t;\theta,p)$ by

$$b(n,t) = \frac{\theta \log n}{H(\theta,p)} - \frac{1}{2}\log_{1/r}\log(n) - \frac{1}{2}\log_{1/r}\left(\frac{2\pi\theta(1-\theta)}{H(\theta,p)}\right) + \log_{1/r}(\theta - p).$$

We then have:

$$P\left[\limsup_n \frac{S_{n:t} - b(n,t)}{\log_{1/r}\log n} = 1\right] = 1. \tag{1}$$

This means that for $\epsilon > 0$ arbitrarily small and for n large enough if we plot the time series $T_n = (S_{n:t} - b(n,t))/\log_{1/r}\log n$, T_n will approach arbitrarily closely to 1 from below but never reach it, infinitely often. Similarly

$$P\left[\liminf_n \left(S_{n:t} - b(n,t) + \log_{1/r}\log\log n\right) = -1\right] = 1. \tag{2}$$

which means that the time series $W_n = S_{n:t} - b(n,t) + \log_{1/r}\log\log n$ will approach -1 arbitrarily closely from above but never reach it, infinitely often.

Interestingly, when we apply this test to Dragon, as seen in Figure 6, T_n (shown as circles) exceeds +1 somewhat but is usually under it, which is reasonable. However, W_n (shown as diamonds) varies around a mean of roughly -5, which may be an indicator of a problem with Dragon. Clearly, this preliminary observation deserves further study.

4 Conclusions and Discussion

In this paper we have presented a new algorithm for finding the longest substring of a given density in a string, and shown that it runs in $O(n \log n)$ expected time. Moreover, we have given two alternate approaches to solving the problem: exploiting the idea of θ-primitives as introduced in Lemma 1; and exploiting the bound given by the Strong Law of Large Numbers. While our implementations of these ideas did not decrease running times below those of Algorithm SkipMisMatch for parameterisations that would be typical in randomness testing, both seem promising avenues of exploration.

In particular, one ready optimisation for an algorithm that solves the problem could be to check for the existence of any θ-primitive during the construction of the data structure used for rank queries. If no such primitive exists, then no further work is required. This is particularly useful when $|\theta - p|$ is large, but is not so useful if the data structure is computed once for a string, but used for many different θ values.

For the output of a stream cipher, the assumption $p = 1/2$ is eminently reasonable, and is tested by the basic frequency test which is $O(n)$ in complexity. Once the cipher passes that test, one can apply the tests developed here. We have presented preliminary results on the output of Dragon which deserves further study.

Acknowledgements

The authors acknowledge the constructive comments by the anonymous referees whose suggestions have improved the presentation of the results in this paper.

References

1. Arratia, R., Gordon, L., Waterman, M.S.: The Erdös-Rényi Law in Distribution, for Coin Tossing and Pattern Matching. Annals of Statistics 18(2), 539–570 (1990)
2. Arratia, R., Waterman, M.S.: The Erdös-Rényi Strong Law for Pattern Matching with a Given Proportion of Mismatches. Annals of Probability 17(3), 1152–1169 (1989)
3. Boyer, R.S., Moore, J.S.: A Fast String Searching Algorithm. Comm. of the ACM 20(10), 762–772 (1977)
4. L'Ecuyer, P.: Testing Random Number Generators. In: Proceedings of the 1992 Winter Simulation Conference, pp. 305–313 (1992)
5. Erdös, P., Rényi, A.: On a New Law of Large Numbers. J. Analyse Math. 22, 103–111 (1970)
6. González, R., Grabowski, S., Mäkinen, V., Navarro, G.: Practical implementation of rank and select queries. In: Nikoletseas, S.E. (ed.) WEA 2005. LNCS, vol. 3503, pp. 27–38. Springer, Heidelberg (2005)
7. Greenberg, R.I.: Fast and Space-Efficient Location of Heavy or Dense Segments in Run-Length Encoded Sequences. In: Warnow, T.J., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 528–536. Springer, Heidelberg (2003)
8. Knuth, D.: The Art of Computer Programming: Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1981)
9. Marsaglia, G.: A Current View of Random Number Generators. Computer Science and Statistics: The Interface, pp. 3–10. Elsevier Science, Amsterdam (1985)
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
11. Munro, J.I.: Tables. In: Chandru, V., Vinay, V. (eds.) FSTTCS 1996. LNCS, vol. 1180, pp. 37–42. Springer, Heidelberg (1996)
12. Neuenschwander, D.: Probabilistic and Statistical Methods in Cryptology: An Introduction by Selected Topics. In: André, E., Dybkjær, L., Minker, W., Heisterkamp, P. (eds.) ADS 2004. LNCS, vol. 3068. Springer, Heidelberg (2004)
13. National Institute of Standards and Technology, Random Number Generation and Testing, Publication SP-800-22 (visited February 4, 2009), http://csrc.nist.gov/rng/
14. Okanohara, D., Sadakane, K.: Practical entropy-compressed rank/select dictionary. In: Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX 2007) (visited February 6, 2009), http://www.siam.org/proceedings/alenex/2007/
15. Queensland University of Technology, Information Security Institute, CRYPT-XS, http://www.isi.qut.edu.au/resources/cryptx/ (visited February 6, 2009)
16. Queensland University of Technology, Information Security Institute, Dragon Stream Cipher, http://www.isi.qut.edu.au/resources/dragon/ (visited February 6, 2009)
17. Turan, M.S., Doganaksoy, A., Boztaş, S.: On Independence and Sensitivity of Statistical Randomness Tests. In: Golomb, S.W., Parker, M.G., Pott, A., Winterhof, A. (eds.) SETA 2008. LNCS, vol. 5203, pp. 18–29. Springer, Heidelberg (2008)

New Correlations of RC4 PRGA Using Nonzero-Bit Differences

Atsuko Miyaji* and Masahiro Sukegawa

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

Abstract. RC4 is the stream cipher proposed by Rivest in 1987, which is widely used in a number of commercial products because of its simplicity and substantial security. RC4 exploits shuffle-exchange paradigm, which uses a permutation S. Many attacks have been reported so far. No study, however, has focused on correlations in the Pseudo-Random Generation (PRGA) between two permutations S and S' with some differences, nevertheless such correlations are related to an inherent weakness of shuffle-exchange-type PRGA. In this paper, we investigate the correlations between S and S' with some differences in the initial round. We show that correlations between S and S' remain before "i" is in the position where the nonzero-bit difference exists in the initial round, and that the correlations remain with non negligible probability even after "i" passed by the position. This means that the same correlations between S and S' will be observed after the 255-th round. This reveals an inherent weakness of shuffle-exchange-type PRGA.

1 Introduction

RC4 is the stream cipher proposed by Rivest in 1987, which is widely used in a number of commercial products because of its simplicity and substantial security. Though many cryptanalysis of RC4 have been proposed so far [1,4,13,7,2,12,8,11,9,3,5], it has remained secure under proper use. As a result, RC4 is widely used in many applications such as Secure Sockets Layer (SSL), Wired Equivalent Privacy (WEP), etc.

RC4 exploits shuffle-exchange paradigm, which uses a permutation $S = (S[0], \cdots, S[N-1])$ given in the initial, and outputs 8-bit data in each round by updating the permutation S, where typically each $S[i]$ $(i \in [0, N-1])$ is 8 bits and $N = 256$. In more detail, RC4 consists of two algorithms, the Key Scheduling Algorithm (KSA) and the Pseudo Random Generation Algorithm (PRGA). KSA is given a secret key with ℓ bytes (typically, $5 \leq \ell \leq 16$) and generates the initial permutation S_0, which is an input of PRGA. PRGA is given

* This study is partly supported by Grant-in-Aid for Scientific Research (B), 203000032.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 134–152, 2009.

the initial permutation S_0, uses two indices i and j, (where i is a public counter but j is one element of secret state information), updates S and j, and outputs $Z = S[S[i] + S[j]]$ as a key stream at the end of each round. There are mainly two approaches to the cryptanalysis of RC4, analysis of the weaknesses of the KSA, and analysis of the weaknesses of the PRGA. Many works, however, focus on the bias between a secret key and the initial permutation, which is an input of PRGA. Some analysis of the weaknesses of the PRGA also focus on the correlation between the first keystream output of PRGA and the secret key. We have not seen any research on correlations in PRGA between two permutations with some differences. However, such correlations should be investigated, since it is reported that sets of two keys which output either the same initial permutations or initial permutations with differences of just a few bits can be intentionally induced [6]. Furthermore, correlations between outputs of two consecutive rounds is an inherent weakness of shuffle-exchange-type PRGA.

In this paper, we focus on a shuffle-exchange structure of PRGA, where 1 swap is executed in each round. We investigate how the structure mixes the permutation S, by observing correlations between two permutations, S and S', with some differences in the initial round. The set of indices where differences exist in the initial round is represented by Diff_0. The correlations are measured over (a) the difference value of two permutations $\Delta S = S \oplus S'$, (b) the difference value of two outputs of PRGA, $\Delta Z = Z \oplus Z'$, and (c) the difference value of two indices $\Delta j = j \oplus j'$. We start with $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\}$. Our results, however, are easily applicable to other cases where there exist differences Diff_0 with $\#\mathsf{Diff}_0 > 2$ in the initial round.

We show theoretically that correlations between two permutations S and S', such as $\Delta Z = 0$, $\Delta j = 0$, and the hamming weight of ΔS, remain when $i < \mathsf{df}_0[1]$. Furthermore, we show that such correlations between two permutations S and S' remain with non negligible probability when $i \geq \mathsf{df}_0[1]$, thus, the same correlations between permutations will be observed when $i < \mathsf{df}_0[2]$. For example, the probability that such correlations remain when $i > \mathsf{df}_0[1]$ is greater than 30% in the cases of $\mathsf{df}_0[1] \geq 93$. We give the theoretical formulae of the probability of both outputs being equal when $i = \mathsf{df}_0[1]$. All theoretical results have been confirmed experimentally.

This paper is organized as follows. Section 2 summarizes the known facts on RC4 together with notation. Section 3 investigates correlations in each round between two permutations S and S' with some differences in the initial round. Section 4 investigates correlations in each round between outputs of two permutations S and S'. Section 5 shows the experimental results which confirm all theories in Sections 3 and 4. Section 6 investigates how to predict inner states.

2 Preliminary

This section presents the KSA and the PRGA of RC4, after explaining the notations used in this paper.

S, S' : permutations
S_0, S'_0 : the initial permutations of PRGA
Diff_0 : the set of indices where differences between S and S' exist in the initial round
r : number of rounds ($r = 0$ means the initial round)
$\mathsf{df}_0[1], \mathsf{df}_0[2], \cdots$: the positions where differences exist in the initial round
i_r, j_r (j'_r): i and j (j') of S (S') after r rounds
S_r (S'_r): the permutation S (S') after r rounds
$S_r[i]$ ($S'_r[i]$): the value of S_r (S'_r) in the position i after r rounds
ΔS_r : $S_r \oplus S'_r$
$\Delta S_r[i]$: $S_r[i] \oplus S'_r[i]$
$|\Delta S_r|$: the number of indices with $\Delta S_r[i] \neq 0$
Z_r (Z'_r): the output under S (S') at the r-th round
ΔZ_r : $Z_r \oplus Z'_r$
Δj_r : $j_r \oplus j'_r$
$\Delta\mathsf{State}[0], \Delta\mathsf{State}[1], \cdots$: the state differences between S and S' (j and j') in a round r.
(The state differences of i are omitted since the same i is used each other.)

RC4 has a secret internal state which is a permutation of all the $N = 2^n$ possible n-bit words and index j. RC4 generates a pseudo-random stream of bits (a keystream) which, for encryption, is combined with the plaintext using XOR; decryption is performed in the same way. To generate the keystream, the cipher makes use of a secret internal state which consists of two parts (shown in Figure 1): A key scheduling algorithm, KSA, which turns a random key (whose typical size is 40-256 bits) into an initial permutation S_0 of $\{0, \ldots, N-1\}$, and an output generation algorithm, PRGA, which uses the initial permutation to generate a pseudo-random output sequence.

The algorithm KSA consists of N loops. It initializes S to be the identity permutation, and both i and j to 0, and then repeats three simple operations: increment i, which acts as a counter, set j by using S and a secret key K with ℓ bytes where each word contains n bits, and swap two values of S in positions i and j. Finally, it outputs a random permutation $S = S_0$.

KSA(K)	PRGA(K)
Initialization	Initialization:
For $i = 0 \ldots N-1$	$i = 0$
$S[i] = i$	$j = 0$
$j = 0$	Generation loop:
Scrambling:	$i = i + 1$
For $i = 0 \ldots N-1$	$j = j + S[i]$
$j = j + S[i] + K[i \pmod{\ell}]$	Swap($S[i], S[j]$)
Swap($S[i], S[j]$)	Output $z = S[S[i] + S[j]]$

Fig. 1. The Key Scheduling Algorithm and the Pseudo-Random Generation Algorithm

The algorithm PRGA is similar to KSA. It repeats four simple operations: increment i, which act as a counter, set j by using S and the previous j, swap two values of S in positions i and j, and output the value of S in position $S[i]+S[j]$. Each value of S is swapped at least once (possibly with itself) within any N consecutive rounds. All additions used in both KSA and PRGA are in general additions modulo N unless specified otherwise.

3 State Analysis of Permutations with Some Differences

This section analyzes correlations between two permutations, S and S', with some differences in the initial round. The set of indices where differences exist in the initial round is represented by $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2], \cdots\}$. The indices with nonzero bit differences are arranged in order of positions that i will reach after the next round. Therefore, if nonzero bit differences exist in positions 0 and $N-1$ in the initial round, then $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\} = \{N-1, 0\}$ since i will be incremented to 1 in the first round.

3.1 Overview of Analysis

Assume that two permutations S and S' with $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\}$ in the initial round are given, where $(S_0[\mathsf{df}_0[1]], S_0[\mathsf{df}_0[2]]) = (a, b)$ and $(S'_0[\mathsf{df}_0[1]], S'_0[\mathsf{df}_0[2]]) = (b, a)$ (See Figure 2). Then, the initial state of differences between S_0 and S'_0 is:

$$\Delta\mathsf{State}[0] : (\Delta S[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_0) \wedge (\Delta j = 0).$$

Then, we analyze the conditions in each round in which the initial state changes from the current state to another, or remains the same.

The transitions of state are different according to the position of i, that is, $i < \mathsf{df}_0[1]$; $i = \mathsf{df}_0[1]$ and the nonzero bit difference still exists in the position $\mathsf{df}_0[1]$; $i = \mathsf{df}_0[1]$ but the nonzero bit difference does not exist in the position $\mathsf{df}_0[1]$, which are formalized as follows.

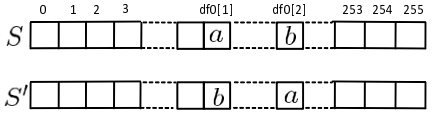


Fig. 2. $\Delta\mathsf{State}[0]$

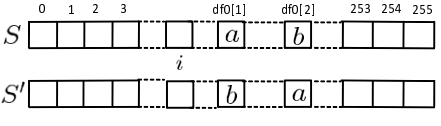


Fig. 3. $\mathsf{Event}[1]$

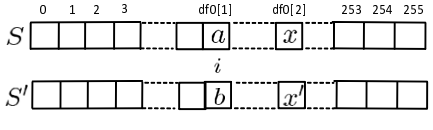


Fig. 4. $\mathsf{Event}[2]$

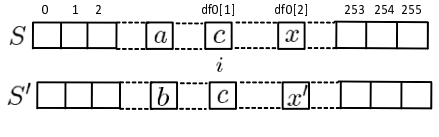


Fig. 5. $\mathsf{Event}[3]$

$$\mathsf{Event}[1] : i_r < \mathsf{df}_0[1] \quad \text{(Figure 3)},$$
$$\mathsf{Event}[2] : [i_r = \mathsf{df}_0[1]] \wedge [\Delta S_{r-1}[\mathsf{df}_0[1]] \neq 0] \quad \text{(Figure 4)},$$
$$\mathsf{Event}[3] : [i_r = \mathsf{df}_0[1]] \wedge [\Delta S_{r-1}[\mathsf{df}_0[1]] = 0] \quad \text{(Figure 5)}.$$

Figures 3, 4, and 5 show each event, where $(x, x') = (b, a)$ or $x = x'$. We will see the reason for this in the following subsections. The following subsections describe each transition and the probability of its occurrence in each event. We will see that the state of differences between two permutations S and S' has the Markov property, that is, given the state in a certain round (the present state), the state in a future round (future states) is independent of past rounds.

3.2 Transitions of ΔState[0] Before the Nonzero Bit Difference

This subsection shows Theorem 1, which describes the transitions from the initial state in Event[1] and their associated probabilities. The state diagram is given in Figure 6D

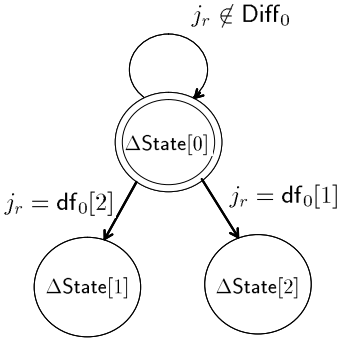


Fig. 6. State Diagram of PRGA in Event[1]

Theorem 1. *Assume that two initial permutations S and S' are in the state of differences ΔState[0] in the $(r-1)$-th round, and that* Event[1] *occurs in the r-th round.*
(1) The state changes to the state ΔState[0] (resp. ΔState[1], resp. ΔState[2]) if $j_r \notin \mathsf{Diff}_0$ (resp. $j_r = \mathsf{df}_0[2]$, resp. $j_r = \mathsf{df}_0[1]$), where

$$\Delta\mathsf{State}[0] : [\Delta S_r[x] \neq 0 \iff x \in \mathsf{Diff}_0] \wedge [\Delta j_r = 0],$$
$$\Delta\mathsf{State}[1] : [\Delta S_r[x] \neq 0 \iff x \in \mathsf{Diff}_1] \wedge [\Delta j_r = 0],$$
$$\Delta\mathsf{State}[2] : [\Delta S_r[x] \neq 0 \iff x \in \mathsf{Diff}_2] \wedge [\Delta j_r = 0].$$

and where $\mathsf{Diff}_1 = \{\mathsf{df}_1[1], \mathsf{df}_1[2]\} = \{\mathsf{df}_0[1], i_r\}$ *and* $\mathsf{Diff}_2 = \{\mathsf{df}_2[1], \mathsf{df}_2[2]\} = \{\mathsf{df}_0[2], i_r\}$.
(2) Each transition occurs with the following probabilities if j is distributed randomly:

$$\mathsf{Prob}\,[\Delta\mathsf{State}[0]] = \frac{N-2}{N},\ \mathsf{Prob}\,[\Delta\mathsf{State}[1]] = \frac{1}{N},\ \textit{and}\ \mathsf{Prob}\,[\Delta\mathsf{State}[2]] = \frac{1}{N},$$

where each probability is taken over choices of S and S' in state ΔState[0] in the initial round.

Proof: (1) It is clear that $j_r = j'_r$ holds in any case, since $j_r = j_{r-1} + S_{r-1}[i_r]$, $\Delta j_{r-1} = 0$, and $i_r \notin \mathsf{Diff}_0$. In the case of $j_r \notin \mathsf{Diff}_0$, $\Delta S_{r-1}[i_r] = \Delta S_{r-1}[j_r] = 0$ holds and, thus, positions of non-zero-bit differences remain the same as those in $(r-1)$-round. Therefore, $\Delta\mathsf{State}[0]$ occurs. In the case of $j_r = \mathsf{df}_0[2]$,

$$(S_r[i_r], S_r[j_r]) = (S_{r-1}[j_r], S_{r-1}[i_r]) = (b, S_{r-1}[i_r]);$$
$$(S'_r[i_r], S'_r[j'_r]) = (S'_{r-1}[j'_r], S'_{r-1}[i_r]) = (a, S'_{r-1}[i_r]);$$

and, thus, the non-zero-bit difference in the position $\mathsf{df}_0[2]$ moves to the current i_r. Therefore, $\Delta\mathsf{State}[1]$ occurs. In the case of $j_r = \mathsf{df}_0[1]$,

$$(S_r[i_r], S_r[j_r]) = (S_{r-1}[j_r], S_{r-1}[i_r]) = (a, S_{r-1}[i_r]);$$
$$(S'_r[i_r], S'_r[j'_r]) = (S'_{r-1}[j'_r], S'_{r-1}[i_r]) = (b, S'_{r-1}[i_r]);$$

and, thus, the non-zero-bit difference in the position $\mathsf{df}_0[1]$ moves to the current i_r. Therefore, $\Delta\mathsf{State}[2]$ occurs.

(2) The probability that each state will occur follows from the above discussion. □

Theorem 1 implies that

- $|\Delta S_r| = 2$ and $\Delta j_r = 0$ hold as long as i_r is not equal to the position that a nonzero bit difference exits in the initial round.
- If $j_r = \mathsf{df}_0[1]$ at least once in the r-th round during $i_r < \mathsf{df}_0[1]$, then the nonzero bit difference in the position $\mathsf{df}_0[1]$ moves to the current i_r. As a result, the nonzero-bit difference that was originally in the position $\mathsf{df}_0[1]$ affects neither $|\Delta S|$ nor Δj until the $(r+N-1)$-th round. This is the case in which $\mathsf{Event}[3]$ occurs.

The following corollary describes the detailed cases in which i is not equal to any position that a nonzero bit difference exits before the N-th round.

Corollary 1. *Assume that two initial permutations S and S' in the state of differences $\Delta\mathsf{State}[0]$ are given. Then, if either of the following events occurs, then i is not equal to any position that a nonzero bit difference exits; and both $|\Delta S_r| = 2$ and $\Delta j_r = 0$ hold until the N-th round.*

$$\mathsf{Event}[4] : [j_{r_1} = \mathsf{df}_0[1](1 \le \exists i_{r_1} < \mathsf{df}_0[1])] \quad \wedge [j_{r_2} = \mathsf{df}_0[2](i_{r_1} < \exists i_{r_2} < \mathsf{df}_0[2])]$$
$$\mathsf{Event}[5] : [j_{r_3} = \mathsf{df}_0[2](1 \le \exists i_{r_3} < \mathsf{df}_0[1] - 1)] \wedge [j_{r_4} = \mathsf{df}_0[1](i_{r_3} < \exists i_{r_4} < \mathsf{df}_0[1])]\,.$$

Note that i_{r_3} is less than $\mathsf{df}_0[1] - 1$ since $i_{r_3} < i_{r_4} < \mathsf{df}_0[1]$.

Proof: Assume that $\mathsf{Event}[4]$ has occurred in (j_{r_1}, j_{r_2}), that is, first $j_{r_1} = \mathsf{df}_0[1]$ for $1 \le i_{r_1} < \mathsf{df}_0[1]$ has occurred. This means that $\Delta\mathsf{State}[2]$ has occurred in the index of i_{r_1} and, thus, $\Delta S_{r_1}[x] \neq 0 \iff x \in \mathsf{Diff}_2$. Therefore, the nonzero-bit difference in the position $\mathsf{df}_0[1]$ moves to the position i_{r_1}. Next, it is assumed that $j_{r_2} = \mathsf{df}_0[2](i_{r_1} < i_{r_2} < \mathsf{df}_0[2])$ has occurred. Then, $\Delta S_{r_2}[x] \neq 0 \iff x \in \{i_{r_1}, i_{r_2}\}$ by applying Theorem 1 to Diff_2. Thus, i is not equal to any position that a nonzero bit difference exits until the N-th round.

Assume that $\mathsf{Event}[5]$ has occurred in (j_{r_3}, j_{r_4}), that is, first $j_{r_3} = \mathsf{df}_0[2]$ for $1 \leq i_{r_3} < \mathsf{df}_0[1] - 1$ has occurred. This means that $\Delta\mathsf{State}[1]$ has occurred in the index of i_{r_3} and, thus, $\Delta S_{r_3}[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_1$. Then, the index $\mathsf{df}_0[2]$ no longer indicates a nonzero bit difference. Next, it is assumed that $j_{r_4} = \mathsf{df}_0[1](i_{r_3} < i_{r_4} < \mathsf{df}_0[1])$ has occurred. Then, $\Delta S_{r_4}[x] \neq 0 \Longleftrightarrow x \in \{i_{r_3}, i_{r_4}\}$ by applying Theorem 1 to Diff_1. Thus, i is not equal to any position that a nonzero bit difference exits until the N-th round. □

The probability that $\mathsf{Event}[3]$ occurs, $\mathsf{Prob}\,[\mathsf{Event}[3]]$, is computed by the next theorems.

Theorem 2. *Assume that two initial permutations S and S' in the state of differences $\Delta\mathsf{State}[0]$ with $\mathsf{df}_0[1] \geq 5$ are given. Then, $\mathsf{Event}[3]$ will occur with the following probability if j is distributed randomly:*

$$\mathsf{Prob}\,[\mathsf{Event}[3]] = 1 - \left(\frac{N-1}{N}\right)^{\mathsf{df}_0[1]-1},$$

where the probability is taken over choices of S and S' with differences in Diff_0 in the initial round.

Proof: $\mathsf{Event}[2]$, the complement of $\mathsf{Event}[3]$, occurs if and only if $j \neq \mathsf{df}_0[1]$ during $i < \mathsf{df}_0[1]$. Therefore, $\mathsf{Prob}\,[\mathsf{Event}[3]] = 1 - \left(\frac{N-1}{N}\right)^{\mathsf{df}_0[1]-1}$ if j is distributed randomly. □

In the case of $\mathsf{df}_0[1] < 5$, we can describe $\mathsf{Prob}\,[\mathsf{Event}[3]]$ by the conditions of S_0 as follows:

Theorem 3. *Assume that two initial permutations S and S' in the state of differences $\Delta\mathsf{State}[0]$ with $\mathsf{df}_0[1] \leq 5$ are given. Then, $\mathsf{Event}[3]$ will occur in the following probability if $S_0[1]$, $S_0[2]$, and $S_0[3]$ are distributed randomly:*
(1) In the case of $\mathsf{df}_0[1] = 2$, $\mathsf{Prob}\,[\mathsf{Event}[3]] = \mathsf{Prob}\,[S_0[1] = j_1 = 2] = \frac{1}{N}$,
(2) In the case of $\mathsf{df}_0[1] = 3$,

$$\mathsf{Prob}\,[\mathsf{Event}[3]] = \mathsf{Prob}\,[S_0[1] = 3] + \mathsf{Prob}\,[S_0[1] \neq 2, 3 \wedge S_0[1] + S_0[2] = 3] = \frac{2N-3}{N(N-1)},$$

(3) In the case of $\mathsf{df}_0[1] = 4$,

$$\begin{aligned}\mathsf{Prob}\,[\mathsf{Event}[3]] &= \mathsf{Prob}\,[S_0[1] = 2] + \mathsf{Prob}\,[S_0[1] = 4] + \mathsf{Prob}\,[S_0[1] = 3 \wedge S_0[2] = N-2] \\ &\quad + \mathsf{Prob}\,[S_0[1] \neq 2, 3, 4 \wedge S_0[3] \neq 0, 1 \wedge S_0[1] + S_0[2] + S_0[3] = 4] \\ &= \frac{2(2N-3)}{N(N-1)},\end{aligned}$$

where the probability is taken over choices of S and S' with differences in Diff_0 in the initial round.

Proof: (1) $\mathsf{Event}[3]$ occurs if and only if $j_1 = \mathsf{df}_0[1] = 2$, where $j_1 = j_0 + S_0[1] = S_0[1]$. Therefore, $\mathsf{Prob}\,[\mathsf{Event}[3]] = \mathsf{Prob}\,[S_0[1] = 2] = \frac{1}{N}$.

(2) $\mathsf{Event}[3]$ occurs if and only if $j_1 = \mathsf{df}_0[1] = 3$ or $j_2 = \mathsf{df}_0[1] = 3$. If $S_0[1] = 3$, then we get $j_1 = j_0 + S_0[1] = S_0[1] = 3 = \mathsf{df}_0[1]$. If $S_0[1] \neq 2$, then $S_0[1] = j_1 \neq 2$, which means that $S_0[1]$ is not swapped with $S_0[2]$ in the first round. This implies that $S_1[2] = S_0[2]$. Thus, if $[S_0[1] \neq 2, 3] \wedge [S_0[1] + S_0[2] = 3]$, we get $j_2 = j_1 + S_1[2] = S_0[1] + S_0[2] = 3 = \mathsf{df}_0[1]$. Therefore, $\mathsf{Prob}[\mathsf{Event}[3]] = \frac{1}{N} + \frac{N-2}{N(N-1)} = \frac{2N-3}{N(N-1)}$.
(3) $\mathsf{Event}[3]$ occurs if and only if $j_1 = \mathsf{df}_0[1] = 4$, $j_2 = \mathsf{df}_0[1] = 4$, or $j_3 = \mathsf{df}_0[1] = 4$. If $S_0[1] = 4$, then we get $j_1 = j_0 + S_0[1] = S_0[1] = 4 = \mathsf{df}_0[1]$. If $S_0[1] = 2$, then $j_1 = j_0 + S_0[1] = 2$; $S_0[1]$ is swapped with $S_0[2]$; and, we get $j_2 = j_1 + S_1[2] = j_1 + S_0[1] = 4 = \mathsf{df}_0[1]$. Note that $S_0[1]$ is swapped with $S_0[2]$ if and only if $S_0[1] = 2$. If $S_0[1] \neq 2, 4$ and $S_0[1] + S_0[2] = 4$, then we get $j_2 = j_1 + S_1[2] = S_0[1] + S_0[2] = 4 = \mathsf{df}_0[1]$.

If $S_0[1] = 3$ and $S_0[2] = N-2$, then $j_1 = S_0[1] = 3$; and $S_0[1]$ is swapped with $S_0[3]$, which implies that $(S_1[1], S_1[3]) = (S_0[3], S_0[1])$. Then, in the 2nd round, $j_2 = j_1 + S_1[2] = 3 + S_0[2] = 1$; and $S_1[2]$ is swapped with $S_1[1]$, which implies that $S_2[3] = S_1[3] = 3$. Thus, in the 3rd round, we get $j_3 = j_2 + S_2[3] = 4$. Note that $S_0[1]$ is swapped with $S_0[3]$ if and only if $S_0[1] = 3$.

If $S_0[1] \neq 2, 3, 4$; $S_0[3] \neq 0, 1$; and $S_0[1] + S_0[2] + S_0[3] = 4$, then $S_1[2] = S_0[2]$; $S_1[3] = S_0[3]$; and $S_0[1] + S_0[2] \neq 3$. This implies that $S_1[3]$ is not swapped with $S_1[2]$ and that $S_2[3] = S_1[3]$. Thus, we get $j_3 = S_0[1] + S_0[2] + S_0[3] = 4$. To sum up all conditions, which are independent of each other, $\mathsf{Prob}[\mathsf{Event}[3]] = \frac{2}{N} + \frac{N-2}{N(N-1)} + \frac{1}{N(N-1)} + \frac{N-3}{N(N-1)} = \frac{2(2N-3)}{N(N-1)}$. □

3.3 Transitions of $\Delta\mathsf{State}[0]$ on the Nonzero Bit Difference

This subsection shows Theorem 4, which describes each transition of the initial state $\Delta\mathsf{State}[0]$ and the probability of its occurrence in $\mathsf{Event}[2]$. The state diagram is given in Figure 7.

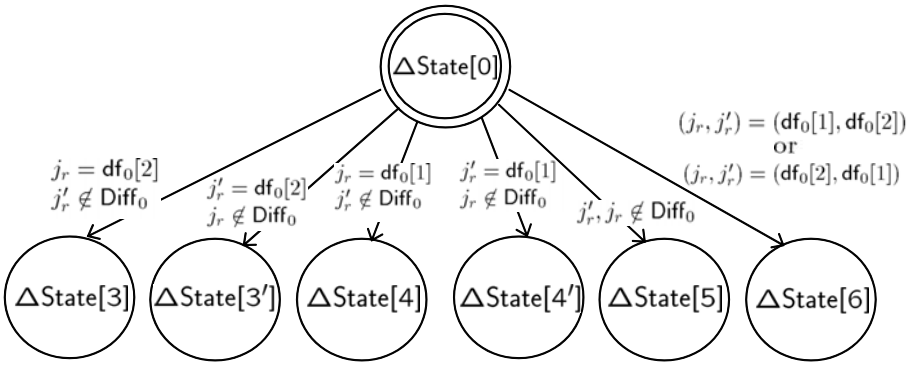


Fig. 7. State Diagram of PRGA in $\mathsf{Event}[2]$

Theorem 4. *Assume that two permutations S and S' are in the state of differences $\Delta\mathsf{State}[0]$ in the $(r-1)$-th round.*
(1) The state changes to $\Delta\mathsf{State}[3]$ (resp. $\Delta\mathsf{State}[3']$, resp. $\Delta\mathsf{State}[4]$, resp. $\Delta\mathsf{State}[4']$, resp. $\Delta\mathsf{State}[5]$, resp. $\Delta\mathsf{State}[6]$), if $[j_r = \mathsf{df}_0[2]] \wedge [j'_r \notin \mathsf{Diff}_0]$ (resp. $[j'_r = \mathsf{df}_0[2]] \wedge [j_r \notin \mathsf{Diff}_0]$, resp. $[j_r = \mathsf{df}_0[1]] \wedge [j'_r \notin \mathsf{Diff}_0]$, resp. $[j'_r = \mathsf{df}_0[1]] \wedge [j_r \notin \mathsf{Diff}_0]$, resp. $j'_r, j_r \notin \mathsf{Diff}_0$, resp. $j'_r, j_r \in \mathsf{Diff}_0$), where

$$\begin{aligned}
\Delta\mathsf{State}[3]&: [\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_3] \wedge [\Delta j_r \neq 0],\\
\Delta\mathsf{State}[3']&: [\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_{3'}] \wedge [\Delta j_r \neq 0],\\
\Delta\mathsf{State}[4]&: [\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_4] \wedge [\Delta j_r \neq 0],\\
\Delta\mathsf{State}[4']&: [\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_{4'}] \wedge [\Delta j_r \neq 0],\\
\Delta\mathsf{State}[5]&: [\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_5] \wedge [\Delta j_r \neq 0],\\
\Delta\mathsf{State}[6]&: [|\Delta S_r| = 0] \wedge [\Delta j_r \neq 0],
\end{aligned}$$

where

$$\begin{aligned}
\mathsf{Diff}_3 &= \{\mathsf{df}_3[1], \mathsf{df}_3[2]\} &&= \{\mathsf{df}_0[1], j'_r\} &&= \{i_r, j'_r\},\\
\mathsf{Diff}_{3'} &= \{\mathsf{df}_{3'}[1], \mathsf{df}_{3'}[2]\} &&= \{\mathsf{df}_0[1], j_r\} &&= \{i_r, j_r\},\\
\mathsf{Diff}_4 &= \{\mathsf{df}_4[1], \mathsf{df}_4[2], \mathsf{df}_4[3]\} &&= \{\mathsf{df}_0[1], \mathsf{df}_0[2], j'_r\} &&= \{i_r, \mathsf{df}_0[2], j'_r\},\\
\mathsf{Diff}_{4'} &= \{\mathsf{df}_{4'}[1], \mathsf{df}_{4'}[2], \mathsf{df}_{4'}[3]\} &&= \{\mathsf{df}_0[1], \mathsf{df}_0[2], j_r\} &&= \{i_r, \mathsf{df}_0[2], j_r\},\\
\mathsf{Diff}_5 &= \{\mathsf{df}_5[1], \mathsf{df}_5[2], \mathsf{df}_5[3], \mathsf{df}_5[4]\} &&= \{\mathsf{df}_0[1], \mathsf{df}_0[2], j_r, j'_r\} &&= \{i_r, \mathsf{df}_0[2], j_r, j'_r\}.
\end{aligned}$$

(2) Each transition occurs with the following probability, if j is distributed randomly:

$$\begin{aligned}
\mathsf{Prob}\,[\Delta\mathsf{State}[3] \vee \Delta\mathsf{State}[3']] &= \mathsf{Prob}\,[\mathsf{Event}[2]] \cdot \tfrac{2(N-2)}{N(N-1)},\\
\mathsf{Prob}\,[\Delta\mathsf{State}[4] \vee \Delta\mathsf{State}[4']] &= \mathsf{Prob}\,[\mathsf{Event}[2]] \cdot \tfrac{2(N-2)}{N(N-1)},\\
\mathsf{Prob}\,[\Delta\mathsf{State}[5]] &= \mathsf{Prob}\,[\mathsf{Event}[2]] \cdot \tfrac{(N-2)(N-3)}{N(N-1)},\\
\mathsf{Prob}\,[\Delta\mathsf{State}[6]] &= \mathsf{Prob}\,[\mathsf{Event}[2]] \cdot \tfrac{2}{N(N-1)}.
\end{aligned}$$

Proof: (1) It is clear that $j_r \neq j'_r$ in each case, since $\Delta j_r = \Delta j_{r-1} + \Delta S_{r-1}[i_r] = \Delta S_{r-1}[i_r] \neq 0$. In the case of $j_r = \mathsf{df}_0[2]$ and $j'_r \notin \mathsf{Diff}_0$, $S_{r-1}[i_r] = S_{r-1}[\mathsf{df}_0[1]] = a$ is swapped with $S_{r-1}[j_r] = b$; $S'_{r-1}[i_r] = S'_{r-1}[\mathsf{df}_0[1]] = b$ is swapped with $S'_{r-1}[j_r]$, which implies that $S'_{r-1}[\mathsf{df}_0[2]] = a$ remains the same. Thus, we get $\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_3$ after the r-th round. In the case of $j'_r = \mathsf{df}_0[2]$ and $j_r \notin \mathsf{Diff}_0$, the same also holds.

In the case of $j_r = \mathsf{df}_0[1]$ and $j'_r \notin \mathsf{Diff}_0$, $i_r = j_r = \mathsf{df}_0[1]$ occurs; $S_{r-1}[i_r] = S_{r-1}[j_r] = a$ remains the same; and $S'_{r-1}[i_r] = b$ is swapped with $S'_{r-1}[j_r]$, Thus, we get $\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_4$ after the r-th round. In the case of $j'_r = \mathsf{df}_0[1]$ and $j_r \notin \mathsf{Diff}_0$, the same also holds.

In the case of $j'_r, j_r \notin \mathsf{Diff}_0$, $S_{r-1}[i_r] = a$ (resp. $S'_{r-1}[i_r] = b$) is swapped with $S_{r-1}[j_r]$ (resp. $S'_{r-1}[j'_r]$), where nonzero-bit difference did not exist; and both $S_{r-1}[\mathsf{df}_0[2]] = b$ and $S'_{r-1}[\mathsf{df}_0[2]] = a$ still remain the same. Thus, we get $\Delta S_r[x] \neq 0 \Longleftrightarrow x \in \mathsf{Diff}_5$ after the r-th round.

In the case of $(j_r, j'_r) = (\mathsf{df}_0[1], \mathsf{df}_0[2])$, $S'_{r-1}[i_r] = S'_{r-1}[\mathsf{df}_0[1]] = b$ is swapped with $S'_{r-1}[j'_r] = S'_{r-1}[\mathsf{df}_0[2]] = a$ while both $S_{r-1}[i_r] = S_{r-1}[j_r] = a$

and $S_{r-1}[j_r] = b$ remain the same. Thus, all nonzero-bit differences disappear after swapping in the r-th round. The same also holds in the case of $(j_r, j'_r) = (\mathsf{df}_0[2], \mathsf{df}_0[1])$.
(2) The probability that each state will occur follows from the above discussion. □

4 Correlation between Outputs and State Transitions

This section analyzes the differences between outputs of two permutations S and S' in each transition described in Section 3, where two initial permutations S and S' are in the state of differences $\Delta\mathsf{State}[0]$.

4.1 Outputs before the Nonzero-Bit Difference

This subsection investigates the correlation between outputs of two permutations in each transition before the first nonzero-bit difference (i.e. $i < \mathsf{df}_0[1]$). The states of differences of two permutations in any round $r < \mathsf{df}_0[1]$ are $\Delta\mathsf{State}[0]$, $\Delta\mathsf{State}[1]$, or $\Delta\mathsf{State}[2]$ from Theorem 1. The probability that both outputs of permutations are equal, $\mathsf{Prob}\,[\Delta Z = 0]$, is given in the next theorem.

Proposition 1. *Assume that two initial permutations S and S' are in the state of differences $\Delta\mathsf{State}[0]$ in the $(r-1)$-th round, and that $\mathsf{Event}[1]$ occurs in the r-th round. Then, $\mathsf{Prob}\,[\Delta Z = 0]$ in each state is as follows:*

$$\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-2}{N}, \frac{2}{N(N-1)}, \text{ or } \frac{2}{N(N-1)}$$

if $\Delta\mathsf{State}[0]$, $\Delta\mathsf{State}[1]$, or $\Delta\mathsf{State}[2]$ occurs, respectively.

Proof: Theorem 1 has shown that

- $\Delta j_r = 0$ and $j_r, i_r \notin \mathsf{Diff}_0$ if $\Delta\mathsf{State}[0]$,
- $\Delta j_r = 0$, $i_r \in \mathsf{Diff}_1$ and $j_r \notin \mathsf{Diff}_1$ if $\Delta\mathsf{State}[1]$,
- $\Delta j_r = 0$, $i_r \in \mathsf{Diff}_1$ and $j_r \notin \mathsf{Diff}_2$ if $\Delta\mathsf{State}[2]$.

Then, the necessary and sufficient conditions for $\Delta Z = 0$ in each state are as follows.

In $\Delta\mathsf{State}[0]$: $\Delta Z = 0 \iff [\Delta(S_r[i_r] + S_r[j_r]) = 0] \wedge [S_r[i_r] + S_r[j_r] \notin \mathsf{Diff}_0]$
$\iff S_r[i_r] + S_r[j_r] \notin \mathsf{Diff}_0$
Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-2}{N}$.
In $\Delta\mathsf{State}[1]$: $\Delta Z = 0 \iff [\Delta(S_r[i_r] + S_r[j_r]) \neq 0] \wedge [S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j_r] \in \mathsf{Diff}_1]$
$\iff S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j_r] \in \mathsf{Diff}_1$
Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{2}{N(N-1)}$ since $\#\mathsf{Diff}_1 = 2$ and $S_r[i_r] + S_r[j_r] \neq S'_r[i_r] + S'_r[j_r]$.
In $\Delta\mathsf{State}[2]$: $\Delta Z = 0 \iff [\Delta(S_r[i_r] + S_r[j_r]) \neq 0] \wedge [S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j_r] \in \mathsf{Diff}_2]$
$\iff S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j_r] \in \mathsf{Diff}_2$
Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{2}{N(N-1)}$ since $\#\mathsf{Diff}_1 = 2$ and $S_r[i_r] + S_r[j_r] \neq S'_r[i_r] + S'_r[j_r]$.

From the above, Proposition 1 follows. □

From Theorem 1 and Proposition 1, the probability of $\mathsf{Prob}\,[\Delta Z = 0]$ if $r < \mathsf{df}_0[1]$ (i.e. $i < \mathsf{df}_0[1]$) can be computed as follows.

Corollary 2. *Assume that two initial permutations S and S' with $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\}$ are given. Then, $\mathsf{Prob}\,[\Delta Z = 0] = \left(\frac{N-2}{N}\right)^2 + \frac{4}{N^2(N-1)}$, if $r < \mathsf{df}_0[1]$.*

4.2 Outputs on the Nonzero-Bit Difference

This subsection investigates the correlation between outputs of two permutations in each transition when $r = \mathsf{df}_0[1]$ (i.e. $i = \mathsf{df}_0[1]$). The probability that both outputs are equal, $\mathsf{Prob}\,[\Delta Z = 0]$, is given in the next theorem.

Proposition 2. *Assume that two initial permutations S and S' are in the state of differences $\Delta\mathsf{State}[0]$ in the $(r-1)$-th round, and that $\mathsf{Event}[2]$ occurs in the r-th round. Then, $\mathsf{Prob}\,[\Delta Z = 0]$ in each state is as follows:*

$$\mathsf{Prob}\,[\Delta Z = 0] = \frac{2}{N(N-1)} \quad \text{if } \Delta\mathsf{State}[3] \vee \Delta\mathsf{State}[3']$$
$$\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-3}{N(N-2)} + \frac{3}{N(N-1)} \quad \text{if } \Delta\mathsf{State}[4] \vee \Delta\mathsf{State}[4']$$
$$\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-4}{N(N-3)} + \frac{4}{N(N-1)} \quad \text{if } \Delta\mathsf{State}[5]$$
$$\mathsf{Prob}\,[\Delta Z = 0] = 0 \quad \text{if } \Delta\mathsf{State}[6]$$

Proof: Let c and $c' \in [0, N-1]$ be values in positions of j_r and j'_r before swapping in the r-th round, that is, $(c, c') = (S_{r-1}[j_r], S'_{r-1}[j'_r])$. On the other hand, $(a, b) = (S_{r-1}[\mathsf{df}_0[1]], S_{r-1}[\mathsf{df}_0[2]]) = (S'_{r-1}[\mathsf{df}_0[2]], S'_{r-1}[\mathsf{df}_0[1]])$. (See Figure 2). Theorem 4 has shown that:

$\Delta\mathsf{State}[3] : (S_r[i_r], S_r[j_r]) = (b, a) \wedge (S'_r[i_r], S'_r[j'_r]) = (c', b)$ (i.e. $c = b$ and $c' \neq a, b$);
$\Delta\mathsf{State}[4] : (S_r[i_r], S_r[j_r]) = (a, a) \wedge (S'_r[i_r], S'_r[j'_r]) = (c', b)$ (i.e. $a = c$ and $c' \neq a, b$);
$\Delta\mathsf{State}[5] : (S_r[i_r], S_r[j_r]) = (c, a) \wedge (S'_r[i_r], S'_r[j'_r]) = (c', b)$ (i.e. $c' \neq c$ and $c', c \neq a, b$);
$\Delta\mathsf{State}[6] : \Delta S_r = 0,\ (S_r[i_r], S_r[j_r]) = (a, a) \wedge (S'_r[i_r], S'_r[j'_r]) = (a, b)$ (i.e. $i_r = j_r$);
or $\Delta S_r = 0,\ (S_r[i_r], S_r[j_r]) = (b, a) \wedge (S'_r[i_r], S'_r[j'_r]) = (b, b)$ (i.e. $i_r = j'_r$).

Therefore, the necessary and sufficient conditions of $\Delta Z = 0$ in each state are as follows.

In $\Delta\mathsf{State}[3] : \Delta Z = 0$
$\iff [S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j'_r] \in \mathsf{Diff}_3] \wedge [\Delta(S_r[i_r] + S_r[j_r]) \neq 0]$
$\iff [(a + b, c' + b) = (\mathsf{df}_0[1], j'_r), (j'_r, \mathsf{df}_0[1])]$.

Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{2}{N(N-1)}$ since $a + b \neq c' + b$ always holds.
The same reasoning holds in the case of $\Delta\mathsf{State}[3']$.

In $\Delta\mathsf{State}[4] : \Delta Z = 0$

$\iff [[\Delta(S_r[i_r] + S_r[j_r]) = 0] \wedge [S_r[i_r] + S_r[j_r] \notin \mathsf{Diff}_4]] \bigvee [[\Delta(S_r[i_r] + S_r[j_r]) \neq 0] \wedge$
$[S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j'_r] \in \mathsf{Diff}_4] \wedge S_r[S_r[i_r] + S_r[j_r]] = S'_r[S'_r[i_r] + S'_r[j'_r]]]$
$\iff [2a = c' + b \wedge 2a \notin \mathsf{Diff}_4] \bigvee [(2a, c' + b) = (i_r, \mathsf{df}_0[2]), (j'_r, i_r), (\mathsf{df}_0[2], i_r)]$.

Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-3}{N(N-2)} + \frac{3}{N(N-1)}$.
The same reasoning holds in the case of $\Delta\mathsf{State}[4']$.

In $\Delta\mathsf{State}[5] : \Delta Z = 0$

$\iff [[\Delta(S_r[i_r] + S_r[j_r]) = 0] \wedge [S_r[i_r] + S_r[j_r] \notin \mathsf{Diff}_5]] \bigvee [[\Delta(S_r[i_r] + S_r[j_r]) \neq 0] \wedge$
$[S_r[i_r] + S_r[j_r], S'_r[i_r] + S'_r[j'_r] \in \mathsf{Diff}_5] \wedge S_r[S_r[i_r] + S_r[j_r]] = S'_r[S'_r[i_r] + S'_r[j'_r]]]$
$\iff [c + a = c' + b \wedge c + a \notin \mathsf{Diff}_5] \bigvee$
$[(a + c, b + c') = (\mathsf{df}_0[1], j_r), (j_r, \mathsf{df}_0[2]), (\mathsf{df}_0[2], j'_r)], (j'_r, \mathsf{df}_0[1])]$,

Thus, $\mathsf{Prob}\,[\Delta Z = 0] = \frac{N-4}{N(N-3)} + \frac{4}{N(N-1)}$.

In $\Delta\mathsf{State}[6] : \mathsf{Prob}\,[\Delta Z = 0]$ since $\Delta(S_r[i_r] + S_r[j_r]) \neq 0$ and $\Delta S_r = 0$.

From the above, the proposition follows. □

The probability $\mathsf{Prob}\,[\Delta Z = 0]$ when $i = \mathsf{df}_0[1]$ follows immediately from Theorem 4 and Proposition 2.

Corollary 3. *Assume that two permutations S and S' in the $(r-1)$-th round are in $\Delta\mathsf{State}[0]$ and $\mathsf{Event}[2]$ occurs in the r-th round. Then, the probability that both outputs are equal in the r-th round, $\mathsf{Prob}\,[\Delta Z = 0]$, is given as follows:*

$$\mathsf{Prob}\,[\Delta Z = 0] = \mathsf{Prob}\,[\mathsf{Event}[2]] \cdot \left(\frac{N^2 - 4N + 2}{N^2(N-1)} + \frac{2(2N-1)(N-2)}{N^2(N-1)^2} \right)$$

From Corollaries 2 and 3, we get the following theorem.

Theorem 5. *Assume that two initial permutations S and S' with $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\}$ are given. Then, the probability $P_1 = \mathsf{Prob}\,[\Delta Z = 0]$ in the round $r = \mathsf{df}_0[1]$ is given as*

$$\begin{aligned} P_1 &= P_2 \cdot \left((\tfrac{N-2}{N})^2 + \tfrac{4}{N^2(N-1)} \right) + (1 - P_2) \cdot \left(\tfrac{N^2-4N+2}{N^2(N-1)} + \tfrac{2(2N-1)(N-2)}{N^2(N-1)^2} \right), \\ &= P_2 \cdot \left((\tfrac{N-2}{N})^2 - \tfrac{N^2-4N-2}{N^2(N-1)} - \tfrac{2(2N-1)(N-2)}{N^2(N-1)^2} \right) + \tfrac{N^2-4N+2}{N^2(N-1)} + \tfrac{2(2N-1)(N-2)}{N^2(N-1)^2}, \end{aligned}$$

where $P_2 = \mathsf{Prob}\,[\mathsf{Event}[3]]$.

Proof: The state of differences between two permutations has the Markov property. Therefore, the probability $\mathsf{Prob}\,[\Delta Z = 0]$ in $r = \mathsf{df}_0[1]$ is determined only by the state in the r-th round, where either $\mathsf{Event}[2]$ or $\mathsf{Event}[3]$ occurs. Theorem 5 follows from this fact. □

The second term of $(1 - P_2) \cdot \left(\frac{N^2-4N+2}{N^2(N-1)} + \frac{2(2N-1)(N-2)}{N^2(N-1)^2} \right)$ can be dealt with as an error term if $\mathsf{df}_0[1]$ is large, which will be discussed in Section 5.

5 Experimental Results and New Bias

This section shows experimental results of Theorems 2, 3, and 5, and Corollary 2 in Sections 3 and 4. All experiments were conducted under the following conditions: execute KSA of RC4 with $N = 256$ for 10^8 randomly chosen keys of 16 bytes, generate the initial permutation S_0, and set another initial permutation S_0' with Diff_0. Experiments are executed over the following sets of Diff_0: $\mathsf{df}_0[1] = 2, \cdots, 255$[1] for Theorems 2 and 3; and $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\} = \{2 - 254, 255\}$, $\{2, 3 - 255\}$, and $\{3, 4 - 255\}$ for Theorem 5 and Corollary 2. The percentage absolute error ϵ of experimental results compared with theoretical results is computed by $\epsilon = \frac{|\text{experimental value} - \text{theoretical value}|}{\text{experimental value}} \times 100(\%)$, which is also used in [10].

[1] $\mathsf{Event}[3]$ does not depend on $\mathsf{df}_0[2]$. See Theorems 2 and 3.

5.1 Experimental Results of Event[3]

Figure 8 shows experimental results of Prob [Event[3]] and its associated percentage absolute error, where the theoretical value is computed by Theorems 2 and 3. The horizontal axis represents $\mathsf{df}_0[1] = 2, \cdots, 255$. The left side of vertical axis represents Prob [Event[3]], and the right side represents the percentage absolute error. Table 1 shows the cases of $\mathsf{df}_0[1] \leq 6$ in detail.

Table 1. Experimental results with $\epsilon > 5$ of Event[3]

$\mathsf{df}_0[1]$	Theoretical value	Experimental value	$\epsilon(\%)$
2	0.003906	0.005350	26.991
3	0.007797	0.009069	14.027
4	0.015548	0.018221	14.667
5	0.015534	0.016751	7.265
6	0.019379	0.020501	5.472

Only the cases of $2 \leq \mathsf{df}_0[1] \leq 6$ give the percentage absolute error $\epsilon \geq 5$, and, thus, our theoretical formulae closely match the experimental results if $\mathsf{df}_0[1] > 6$. The initial permutation S_0, that is the output of KSA, has a great influence on Event[3] when $\mathsf{df}_0[1]$ is small. Our results indicate that the bias in S_0 is propagated to Prob [Event[3]] as the bias in S_0 has been reported in [8,2,10].

Figure 8 also indicates that the nonzero bit difference in the position $\mathsf{df}_0[1]$ moves to another position until $i = \mathsf{df}_0[1]$ with Prob [Event[3]] $> 30\%$ when $\mathsf{df}_0[1] \geq 93$ and, thus, the correlations between S and S' such as $\Delta j = 0$ and $|\Delta S| = 2$ remain the same until $i = \mathsf{df}_0[2]$.

5.2 Experimental Results of Outputs

Figure 9 shows experimental results of Prob $[\Delta Z = 0]$ in $r = \mathsf{df}_0[1] - 1$, $\mathsf{df}_0[1]$, and $\mathsf{df}_0[1] + 1$, and percentage absolute error in $r = \mathsf{df}_0[1]$ (i.e. $i = \mathsf{df}_0[1] - 1$), where the theoretical value is computed by Theorem 5. The horizontal axis represents $\mathsf{df}_0[1] = 2, \cdots, 254$. The left side of vertical axis represents Prob $[\Delta Z = 0]$, and the right side represents the percentage absolute error. By using the experimental results, we investigate each case of outputs before or on the nonzero-bit difference.

Outputs before the nonzero-bit difference
Let us discuss Prob $[\Delta Z = 0]$ in $r = \mathsf{df}_0[1] - 1$ (i.e. $i = \mathsf{df}_0[1] - 1$) for $\mathsf{df}_0[1] = 2, \cdots, 254$. The probability is theoretically estimated in Corollary 2. Our theoretical and experimental results indicate that both outputs of two permutations are coincident with a high probability Prob $[\Delta Z = 0] > 0.98$ during $i < \mathsf{df}_0[1]$[2].

[2] Similar experimental results to $i = \mathsf{df}_0[1] - 1$ hold during $i < \mathsf{df}_0[1] - 1$.

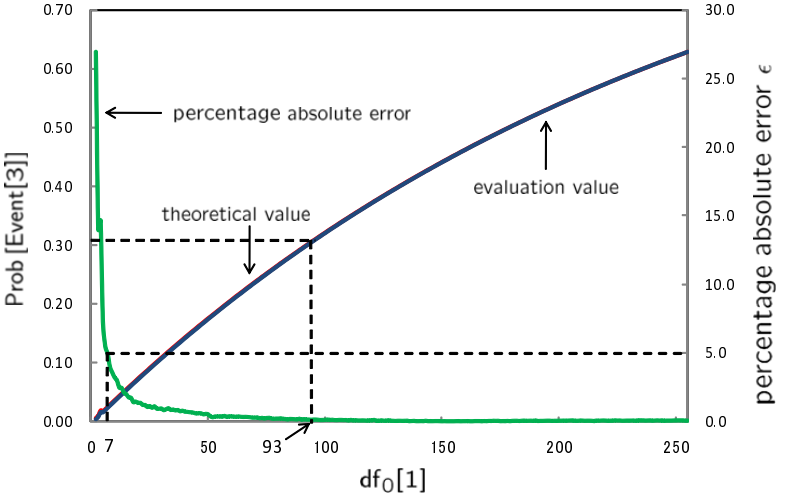


Fig. 8. Experimental results and ϵ of Event[3]

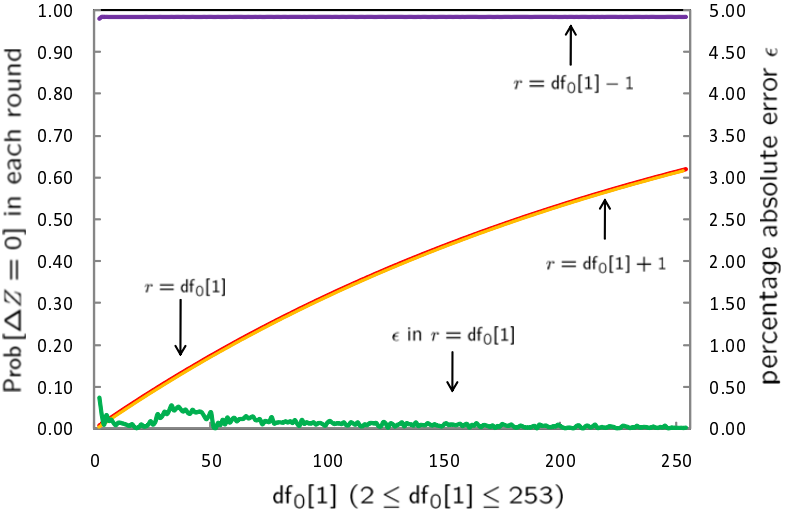


Fig. 9. Prob $[\Delta Z = 0]$ $(\mathsf{df}_0[2] = 255)$

Let us discuss[3] Prob $[\Delta Z = 0]$ in $r = \mathsf{df}_0[1] + 1$ for $\mathsf{df}_0[1] = 2, \cdots, 253$, where $\mathsf{df}_0[1]+1 = i < \mathsf{df}_0[2]$. Actually, it corresponds to the case in which i is before the nonzero bit difference $\mathsf{df}_0[2]$ since $\mathsf{df}_0[1]+1$ is an index of nonzero bit difference when $i = \mathsf{df}_0[1]+1$ from the fact of $\mathsf{df}_0[1]+1 < \mathsf{df}_0[2]$.

[3] The case of $\mathsf{df}_0[1] = 254$ is omitted since i indicates the second nonzero bit difference $\mathsf{df}_0[2] = 255$.

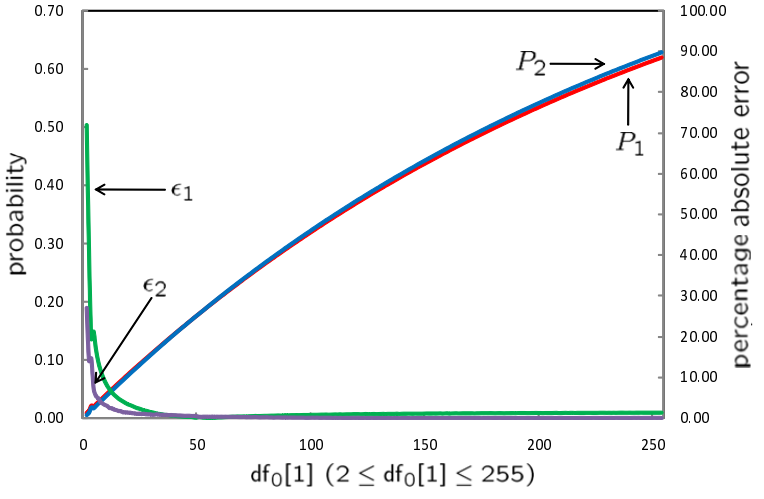


Fig. 10. Comparison of $\mathsf{Prob}\,[\mathsf{Event}[3]]$ and $\mathsf{Prob}\,[\Delta Z = 0]$

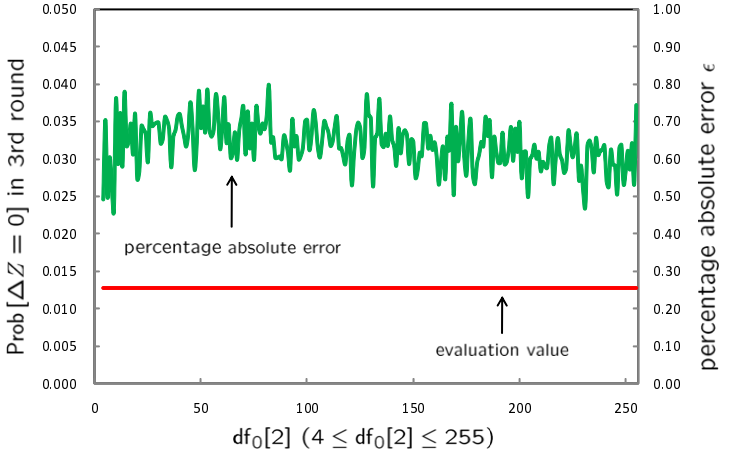


Fig. 11. $\mathsf{Prob}\,[\Delta Z = 0]$ ($\mathsf{df}_0[1] = 3$)

Our experimental results show that $\mathsf{Prob}\,[\Delta Z = 0]$ in the round $\mathsf{df}_0[1] + 1$ is almost the same as in the round $\mathsf{df}_0[1]$, which reflects the results in Theorem 1. To sum up, we see that it is highly probable that both outputs of permutations are coincident as long as i does not indicate the index of nonzero bit difference in the current round.

Outputs before the nonzero-bit difference:
Let us discuss $\mathsf{Prob}\,[\Delta Z = 0]$ in $r = \mathsf{df}_0[1]$, where there exists originally[4] a nonzero-bit difference. $\mathsf{Prob}\,[\Delta Z = 0]$ is estimated theoretically in Theorem

[4] If $\mathsf{Event}[3]$ has occurred in the round $r < \mathsf{df}_0[1]$, then $\mathsf{df}_0[1]$ is not an index of nonzero bit difference.

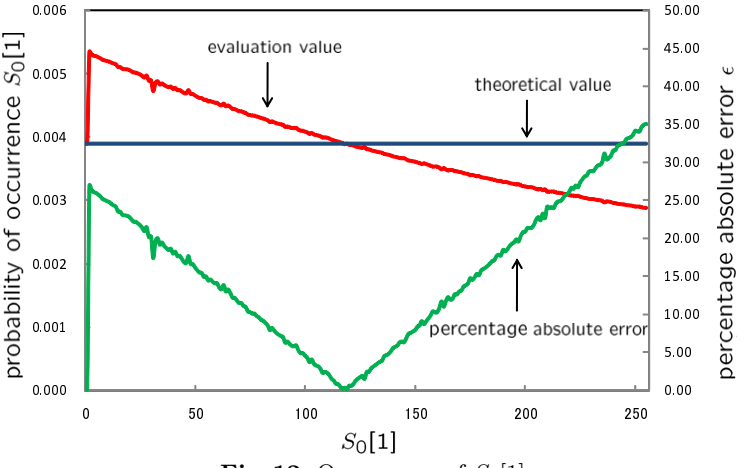


Fig. 12. Occurrence of $S_0[1]$

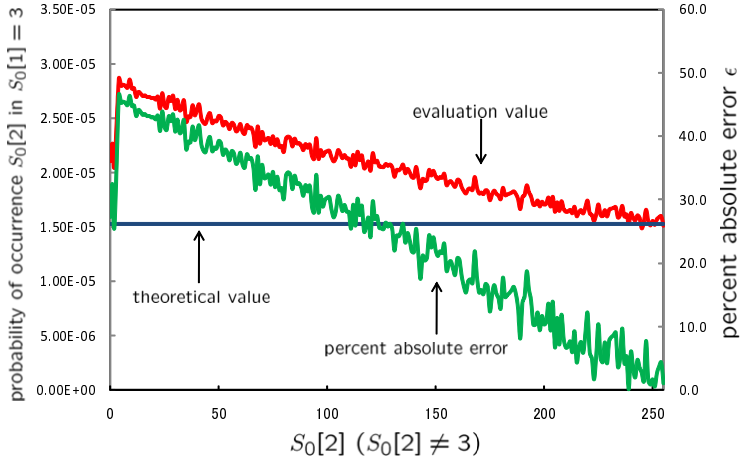


Fig. 13. Occurrence of $S_0[2]$ when $S_0[1] = 3$

5. From the fact that the percentage absolute error $\epsilon < 1$ holds in $2 \leq \forall \mathsf{df}_0[1] \leq 254$, we see that our theoretical formulae closely match the experimental results in any Diff_0.

Let us discuss the relation between two events of $\Delta Z = 0$ and $\mathsf{Event}[3]$ in $r = \mathsf{df}_0[1]$. Figures 8 and 9 show that $\mathsf{df}_0[1]$ satisfying $\mathsf{Prob}\,[\Delta Z = 0] > 30\%$ is almost the same as $\mathsf{df}_0[1]$ satisfying $\mathsf{Prob}\,[\mathsf{Event}[3]] > 30\%$. In fact, $P_1 = \mathsf{Prob}\,[\Delta Z = 0]$ in the round $\mathsf{df}_0[1]$ deeply affects $P_2 = \mathsf{Prob}\,[\mathsf{Event}[3]]$ as we have seen in Theorem 5. Figure 10 shows the comparison between P_1 and P_2 for $2 \leq \mathsf{df}_0[1] \leq 255$, where two percentage absolute errors are listed, $\epsilon_1 = \frac{|P_2 - P_1|}{P_2}$ and $\epsilon_2 = \frac{|P_2 - (theoretical)\mathsf{Prob}[\mathsf{Event}[3]]|}{P_2}$ for experimental values P_1 and P_2. The horizontal axis represents $\mathsf{df}_0[1] = 2, \cdots, 254$. The left side of vertical axis represents

$\mathsf{Prob}\,[\Delta Z = 0]$, and the right side represents the percentage absolute error. Experimental results show that $\epsilon_1 < 5$ (resp. 10) if $\mathsf{df}_0[1] > 15$ (resp. $\mathsf{df}_0[1] > 9$) and, thus, we see that the observable event $\Delta Z = 0$ can indicate that the internal event $\mathsf{Event}[3]$ occurs with extremely high probability.

Figure 11 shows experimental results of $\mathsf{Prob}\,[\Delta Z = 0]$ in the round $\mathsf{df}_0[1] = 3$ in each case of $4 \leq \mathsf{df}_0[2] \leq 255$ ($\mathsf{df}_0[1] = 3$), and percentage absolute error. The horizontal axis represents $\mathsf{df}_0[2]$. The left side of vertical axis represents $\mathsf{Prob}\,[\Delta Z = 0]$, and the right side represents the percentage absolute error. The percentage absolute error $\epsilon < 0.8$ holds in $4 \leq \forall \mathsf{df}_0[2] \leq 255$. We see that our theoretical formulae closely match the experimental results independent of another nonzero-bit difference $\mathsf{df}_0[2]$.

5.3 Experimental Results of Biases in $S_0[1]$ and $S_0[2]$

Let us discuss $\mathsf{Event}[3]$ when $\mathsf{df}_0[1] = 3$ in detail, where the error $\epsilon > 10$ (Table 1). Theorem 3 says that both $S_0[1]$ and $S_0[2]$ determine $\mathsf{Event}[3]$, that is, $\mathsf{Event}[3] \Longleftrightarrow [S_0[1] = 3] \bigvee [S_0[1] \neq 2, 3 \wedge S_0[1] + S_0[2] = 3]$. Here we investigate the bias in $S_0[1]$ and $S_0[2]$ from the point of view of $\mathsf{Event}[3]$.

Figure 12 shows experimental results concerning the occurrence of $S_0[1]$ with $0 \leq S_0[1] \leq 255$, and the percentage absolute error, where the theoretical value (a random association) of occurrence of each $S_0[1]$ is $\frac{1}{N} = 3.906 \times 10^{-3}$. Figure 13 shows experimental results concerning the occurrence of $S_0[2]$ when $S_0[1] = 3$, and the percentage absolute error, where the theoretical value (a random association) of occurrence of each $(S_0[1] = 3, S_0[2])$ is $\frac{1}{N(N-1)} = 1.532 \times 10^{-5}$. The horizontal axis represents $S_0[1]$ or $S_0[2]$. The left side of vertical axis represents each probability, and the right side represents each percentage absolute error.

Table 2. Probability of occurrence $S_0[1]$

$S_0[1]$	Probability of occurrence $S_0[1]$
0 - 9	0.0039 0.0039 0.0054 0.0053 0.0053 0.0053 0.0053 0.0052 0.0052 0.0052
10 - 19	0.0052 0.0052 0.0052 0.0052 0.0052 0.0051 0.0051 0.0051 0.0051 0.0051
20 - 29	0.0051 0.0050 0.0050 0.0050 0.0050 0.0050 0.0050 0.0049 0.0050 0.0049
30 - 39	0.0049 0.0047 0.0049 0.0049 0.0048 0.0049 0.0048 0.0048 0.0048 0.0048

Table 3. Probability of occurrence $S_0[2]$ in $S_0[1] = 3$

$S_0[2]$	Probability of occurrence $S_0[2]$ in $S_0[1] = 3$
0 - 6	0.0000211 0.0000227 0.0000207 - 0.0000286 0.0000280 0.0000281
7 - 13	0.0000280 0.0000278 0.0000286 0.0000277 0.0000278 0.0000270 0.0000274
14 - 20	0.0000273 0.0000270 0.0000271 0.0000270 0.0000270 0.0000269 0.0000269
108 - 114	0.0000216 0.0000213 0.0000213 0.0000206 0.0000216 0.0000207 0.0000219
115 - 121	0.0000212 0.0000216 0.0000204 0.0000207 0.0000210 0.0000202 0.0000218
122 - 128	0.0000210 0.0000211 0.0000206 0.0000206 0.0000205 0.0000208 0.0000206

These experimental results indicate a non-uniform distribution of $S_0[1]$ and $S_0[2]$ when $S_0[1] = 3$. Tables 2 and 3 show some cases that indicate a non-uniform distribution as follows:

$$\begin{aligned} \mathsf{Prob}\,[S_0[1] = 3] &= 5.303 \times 10^{-3} > 3.906 \times 10^{-3}, \\ \mathsf{Prob}\,[S_0[1] = 3 \wedge S_0[2] = x] &> 2.0 \times 10^{-5} > 1.532 \times 10^{-5} \text{ for } \forall x \leq 135, \\ \mathsf{Prob}\,[S_0[1] = 3 \wedge 0 \leq S_0[2] \leq 128] &= 3.05299 \times 10^{-3} > 1.9531 \times 10^{-3}. \end{aligned}$$

These non-uniform distribution will be used for a new cryptanalytic analysis in Section 6.

6 A New Cryptanalytic Analysis

Here we investigate how to analyze the internal state of S or j. Assume that two permutations S and S' with $\mathsf{Diff}_0 = \{\mathsf{df}_0[1], \mathsf{df}_0[2]\}$ in the initial round are given, and that both outputs of PRGA are observable.

Then, by observing both outputs Z and Z' of PRGA, we can recognize the index of the first nonzero-bit difference from the first round in which both outputs are not equal. This is investigated in Section 5.2. Therefore, if neither $\mathsf{df}_0[1]$ nor $\mathsf{df}_0[2]$ are known, the first nonzero-bit difference is predictable.

Consider the case of $\mathsf{df}_0[1] = 2$. By checking whether $\Delta Z = 0$ in the 2nd round, we can recognize whether $\mathsf{Event}[3]$ has occurred. If $\mathsf{Event}[3]$ has occurred, then $S_0[1] = 2$ holds from Theorem 3. The experimental result shows $\mathsf{Prob}\,[\mathsf{Event}[3] \mid \mathsf{df}_0[1] = 2] = 0.005350$ (see Table 1). However, if we try to predict $S_0[1]$ from a random association, then the probability is $1/256 = 0.003906$. Therefore, one can guess $S_0[1]$ with an additional advantage of $\frac{0.005350 - 0.003906}{0.003906} \times 100 = 36.9$ %.

Consider the case of $\mathsf{df}_0[1] = 3$. By checking whether $\Delta Z = 0$ in the 3rd round, we can recognize whether $\mathsf{Event}[3]$ has occurred. Let us discuss how to predict both $S_0[1]$ and $S_0[2]$. If $\mathsf{Event}[3]$ has occurred, then $[S_0[1] = 3] \vee [S_0[1] \neq 2, 3 \wedge S_0[1] + S_0[2] = 3]$ holds, from Theorem 3. In the case of $S_0[1] = 3$, the experimental results show that $\mathsf{Prob}\,[\mathsf{Event}[3] \mid \mathsf{df}_0[1] = 3] = 0.009069$ (see Table 1) and $\mathsf{Prob}\,[S_0[1] = 3] = 0.0053$ (see Table 2). On the other hand, we predict $S_0[2]$ with the probability $1/255$. Therefore, we can predict $(S_0[1], S_0[2])$ with the probability $0.0053 \times 1/255 = 2.078431 \times 10^{-5}$. In the case of $[S_0[1] \neq 2, 3 \wedge S_0[1] + S_0[2] = 3]$, if $S_0[1]$ is predicted, then $S_0[2]$ can be predicted promptly. We find that $\mathsf{Prob}\,[\mathsf{Event}[3] \wedge [S_0[1] \neq 2, 3] \wedge [S_0[1] + S_0[2] = 3]] = (0.009069 - 0.0053) \times 1/254 = 1.483858 \times 10^{-5}$. Therefore, we can predict $(S_0[1], S_0[2])$ with the probability 1.483858×10^{-5}. Taking both together, the probability to predict $(S_0[1], S_0[2])$ is $2.078431 \times 10^{-5} + 1.483858 \times 10^{-5} = 3.562289 \times 10^{-5}$. On the other hand, if we try to predict $(S_0[1], S_0[2])$ from a random association, then the probability is $1/256 \times 1/255 = 1.531863 \times 10^{-5}$. Therefore, one can guess $(S_0[1], S_0[2])$ with an additional advantage of $\frac{3.562289 - 1.531863}{1.531863} \times 100 = 132.54$ %.

7 Conclusion

In this paper, we have investigated, for the first time, correlations between two permutations, S and S', with some differences in the initial round. We have shown that correlations between two permutations S and S' remain before "i" is in the position where the nonzero-bit difference exists in the initial round, and that the correlations remain with non negligible probability even after "i" passed by the position. All theoretical results have been confirmed experimentally.

Our results imply that the same correlations between two permutations will be observed with non negligible probability after the 255-th round. This reveals a new inherent weakness of shuffle-exchange-type PRGA. We have also investigated how to predict inner states such as S and j and shown that we can guess inner states with an additional advantage.

References

1. Knudsen, L.R., Meier, W., Preneel, B., Rijmen, V., Verdoolaege, S.: Analysis methods for (alleged) RC4. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 327–341. Springer, Heidelberg (1998)
2. Mantin, I.: Analysis of the stream cipher RC4, Master's Thesis, The Weizmann Institute of Science, Israel (2001)
3. Paul, S., Preneel, B.: A New Weakness in the RC4 Keystream Generator and an Approach to Improve the Security of the Cipher. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 245–259. Springer, Heidelberg (2004)
4. Mantin, I., Shamir, A.: A practical attack on broadcast RC4. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 87–104. Springer, Heidelberg (2002)
5. Mister, S., Tavares, S.E.: Cryptanalysis of RC4-like Ciphers. In: Tavares, S., Meijer, H. (eds.) SAC 1998. LNCS, vol. 1556, pp. 131–143. Springer, Heidelberg (1999)
6. Matsui, M.: Key Collisions of the RC4 Stream Cipher. In: FSE 2009. LNCS. Springer, Heidelberg (to appear, 2009)
7. Golic, J.: Linear statistical weakness of alleged RC4 keystream generator. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 226–238. Springer, Heidelberg (1997)
8. Mironov, I. (Not So) Random Shuffles of RC4. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 304–319. Springer, Heidelberg (2002)
9. Paul, G., Rathi, S., Maitra, S.: On Non-negligible Bias of the First Output Byte of RC4 towards the First Three Bytes of the Secret Key. Designs, Codes and Cryptography 49, 123–134 (2008)
10. Paul, G., Maitra, S., Srivastava, R.: On Non-Randomness of the Permutation after RC4 Key Scheduling. In: Boztaş, S., Lu, H.-F(F.) (eds.) AAECC 2007. LNCS, vol. 4851. Springer, Heidelberg (2007), http://eprint.iacr.org/2007/305.pdf
11. Mantin, I.: Predicting and Distinguishing Attacks on RC4 Keystream Generator. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 491–506. Springer, Heidelberg (2005)
12. Tomasevic, V., Bojanic, S.: Reducing the State Space of RC4 Stream Cipher. In: Bubak, M., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2004. LNCS, vol. 3036, pp. 644–647. Springer, Heidelberg (2004)
13. Fluhrer, S.R., McGrew, D.A.: Statistical Analysis of the Alleged RC4 Keystream Generator. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 19–30. Springer, Heidelberg (2001)

Analysis of Property-Preservation Capabilities of the ROX and ESh Hash Domain Extenders*

Mohammad Reza Reyhanitabar, Willy Susilo, and Yi Mu

Centre for Computer and Information Security Research,
School of Computer Science and Software Engineering
University of Wollongong, Australia
{rezar,wsusilo,ymu}@uow.edu.au

Abstract. Two of the most recent and powerful multi-property preserving (MPP) hash domain extension transforms are the Ramdom-Oracle-XOR (ROX) transform and the Enveloped Shoup (ESh) transform. The former was proposed by Andreeva et al. at ASIACRYPT 2007 and the latter was proposed by Bellare and Ristenpart at ICALP 2007. In the existing literature, ten notions of security for hash functions have been considered in analysis of MPP capabilities of domain extension transforms, namely CR, Sec, aSec, eSec (TCR), Pre, aPre, ePre, MAC, PRF, PRO. Andreeva et al. showed that ROX is able to preserve seven properties; namely collision resistance (CR), three flavors of second preimage resistance (Sec, aSec, eSec) and three variants of preimage resistance (Pre, aPre, ePre). Bellare and Ristenpart showed that ESh is capable of preserving five important security notions; namely CR, message authentication code (MAC), pseudorandom function (PRF), pseudorandom oracle (PRO), and target collision resistance (TCR). Nonetheless, there is *no* further study on these two MPP hash domain extension transforms with regard to the other properties. The aim of this paper is to fill this gap. Firstly, we show that ROX *does not preserve* two other widely-used and important security notions, namely MAC and PRO. We also show a positive result about ROX, namely that it also preserves PRF. Secondly, we show that ESh *does not preserve* other four properties, namely Sec, aSec, Pre, and aPre. On the positive side we show that ESh can preserve ePre property. Our results in this paper provide a full picture of the MPP capabilities of both ROX and ESh transforms by completing the property-preservation analysis of these transforms in regard to all ten security notions of interest, namely CR, Sec, aSec, eSec (TCR), Pre, aPre, ePre, MAC, PRF, PRO.

Keywords: Hash Functions, Domain Extension, MPP, ROX, ESh.

1 Introduction

A cryptographic hash function is a function that can map variable length strings to fixed length strings. Hash functions have been used in a vast variety of

* The full version of this paper is available from [16].

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 153–170, 2009.

applications, e.g. digital signature, MAC, PRF, and must provide different security properties depending on the security requirements of the applications. The most well-known property for a hash function is collision resistance (CR). Nevertheless, hash functions are often asked to provide many other security properties ranging from merely being a one-way function (i.e. preimage resistance property) to acting as a truly random function (i.e. a random oracle).

In a formal study of cryptographic hash functions two different but related settings can be considered. The first setting is the traditional unkeyed hash function setting where a hash function refers to a single function $H : \mathcal{M} \rightarrow \{0,1\}^n$ (e.g. SHA-1) that maps variable length messages to a fixed length output hash value. In the second setting, a hash function is considered as a family of functions $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, also called a "dedicated-key hash function" [3], indexed by a key space $\mathcal{K}$. The exact role of the hash function key is application-dependent; it can be a public parameter, e.g. when the hash function is used in a digital signature, or a secret key like in MAC and PRF applications. In this paper, we consider hash functions and their security notions in the dedicated-key hash function setting.

Almost all cryptographic hash functions are designed based on the following two-step approach: first a compression function is designed which is only capable of hashing fixed-input-length (FIL) messages and then a domain extension transform is applied to get a full-fledged hash function which can hash variable-input-length (VIL) or arbitrary-input-length (AIL) messages, depending on the transform. Assume that we have a (dedicated-key) compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ that can only hash messages of fixed length $(n+b)$ bits. A domain extension transform can use this compression function (as a black-box) to construct a (dedicated-key) hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, where the message space $\mathcal{M}$ can be either $\{0,1\}^*$ (in which case H is an AIL hash function) or $\{0,1\}^{<2^\lambda}$, for some huge positive integer λ, e.g. $\lambda = 64$ (in which case H is a VIL hash function). For instance, the strengthened-MD domain extension transform [12, 8, 3] yields to a VIL hash function while the Prefix-free domain extension transform [7, 3] yields to an AIL hash function. In practice the difference between being VIL or AIL hash function will not be of a concern as for typical value of $\lambda = 64$ almost all messages will have length less than 2^{64} bits, i.e. will belong to $\{0,1\}^{<2^{64}}$.

From security viewpoint, the crux sought from a domain extension transform is its property preserving capability; that is, if the underlying compression function h possesses some security property P, then the obtained full-fledged hash function H should also *provably* possess the property P. The most well-known domain extension transform is the strengthened Merkle-Damgård (MD) construction which was shown by Merkle [12] and Damgård [8] to be a CR preserving transform. Bellare and Rogaway in [6] showed that strengthened MD, despite preserving CR property, is unable to preserve UOWHF property (put forth by Naor and Yung [15]) which is a weaker than CR property. They renamed UOWHF as target collision resistance (TCR) and provided four domain extension transforms for preserving the TCR property. Shoup in [18] provided a transform (improving XLH

transform of Bellare-Rogaway in [6]), which is shown to be UOWHF and CR preserving. Mironov [14] showed that Shoup's transform is optimal from key expansion viewpoint among masking based serial transforms for TCR preservation. Coron et al. [7] introduced the notion of random oracle preservation and provided prefix-free MD transform which is capable of preserving (pseudo-)random oracle, which means that if the compression function is modeled as a random oracle then the AIL hash function obtained by applying prefix-free MD transform will also be indifferentiable from a random oracle [7, 11]. A new line of research recently has been initiated by Bellare and Ristenpart in [5], and followed in several other works, e.g. [3, 1], with the aim of designing multi-property-preserving (MPP) domain extension transforms. An MPP transform is capable of preserving multiple security properties simultaneously while extending the domain of a compression function.

Two of the most recent and powerful MPP transforms are the Enveloped Shoup (ESh) transform designed by Bellare and Ristenpart in [3] and the Random-Oracle XOR (ROX) transform by Andreeva et al. in [1]. Both ESh and ROX are variants of Shoup (Sh) transform proposed in [18].

ESh is a standard model transform and was shown to be the best-performing, in terms of property preserving capability, among the nine transforms studies in [3]. It is shown in [3] that ESH preserves five security notions; namely CR, TCR, MAC, PRF, and PRO.

ROX was shown to be the only transform among the twelve transforms investigated in [1] which is able to preserve seven security notions; namely CR, Sec, aSec, eSec, Pre, aPre, and ePre as put forth by Rogaway and Shrimpton in [17]. But unlike to other transforms, ROX "..., quite controversially, uses a random oracle in the iteration." [1], although Andreeva et al. in [1] provide arguments justifying the merits of such a *limited application* of auxiliary FIL random oracles in their construction from practical viewpoint.

Our Contribution. We complete property-preservation analysis of the ROX and ESh transforms by providing new negative and positive results in regard to their MPP capabilities. Our results complete the property preservation analysis of ROX and ESh in regard to all ten security notions of interest, namely CR, Sec, aSec, eSec (TCR), Pre, aPre, ePre, MAC, PRF, PRO. For the ROX transform, we show that it *does not preserve* MAC and PRO. This settles the open question of [1] about MAC and PRO preservation capability of the ROX, in a negative way. On the positive side we notice that the ROX is also a PRF preserving transform. Regarding the ESh transform, we show that ESh *does not preserve* Sec, aSec, Pre, and aPre properties. As a positive result about ESh we show that it also preserves ePre property.

The overview of the results are shown in Table 1. A "Yes" means that the property is provably preserved by the transform. A "No" means that the property is not preserved and this is shown either by showing a counterexample compression function or by some attacks benefiting from the structural weakness of the transform in regard to the specific security property. Unreferenced entries are the results shown in this paper. We leave the question of a ten-property-preserving transform *without any random oracle* as an interesting open question.

Table 1. Overview of the MPP capabilities of the ESh and ROX hash domain extension transforms in regard to ten security notions. Unreferenced entries are the results shown in this paper.

	ESh	ROX
CR (Coll)	Yes [3]	Yes [1]
Sec	**No**	Yes [1]
aSec	**No**	Yes [1]
eSec (TCR or UOWHF)	Yes [3]	Yes [1]
Pre	**No**	Yes [1]
aPre	**No**	Yes [1]
ePre	**Yes**	Yes [1]
MAC	Yes [3]	**No**
PRF	Yes [3]	**Yes**
PRO	Yes [3]	**No**

2 Preliminaries

2.1 Notations

If A is a probabilistic algorithm with access to some oracle $f(.)$ then by $y \xleftarrow{\$} A^{f(.)}(x_1, \cdots, x_n)$ it is meant that y is the output random variable which is defined by running A, given inputs $x_1, \cdots, x_n$ and having oracle access to $f(.)$. To show that an algorithm A is run without any input, we use the notation $y \xleftarrow{\$} A()$. By time complexity of an algorithm we mean the running time, relative to some fixed model of computation plus the size of the description of the algorithm using some fixed encoding method. If X is a finite set, by $x \xleftarrow{\$} X$ it is meant that x is chosen from X uniformly at random. By $X \leftarrow Y$ it is meant that the value Y is simply assigned to the variable X. Let $x||y$ denote the string obtained from concatenating string y to string x. Let 1^m and 0^m, respectively, denote a string of m consecutive 1 and 0 bits, and 1^m0^n denote the concatenation of 0^n to 1^m. By (x, y) we mean an injective encoding of two strings x and y, from which one can efficiently recover x and y. For a binary string M, let $|M|$ denote its length in bits and $|M|_b \triangleq \lceil |M|/b \rceil$ denote its length in b-bit blocks. Let $M[i]$ denote the i-th bit of M, and $M_{i...j}$ denote the bits from i-th to j-th positions, i.e. $M_{i...j} = M[i] \cdots M[j]$. If S is a finite set we denote size of S by $|S|$. For a positive integer m, let $\langle m \rangle_\lambda$ denote its representation as a binary string of length exactly λ bits. The set of all binary strings of length n bits (for some positive integer n) is denoted as $\{0,1\}^n$, the set of all binary strings whose lengths are variable but upper-bounded by N is denoted by $\{0,1\}^{\leq N}$ and the set of all binary strings of arbitrary length is denoted by $\{0,1\}^*$. The set of all functions $f : Dom \rightarrow Rng$ (from a domain Dom to a range Rng) is denoted by $\mathrm{Func}(Dom, Rng)$.

2.2 Definition of Security Notions

In this section, we recall definition of ten security notions for hash functions; namely, the seven notions (Coll, Sec, aSec, eSec, Pre, aPre and ePre) formalized

in [17] as well as PRF, MAC, PRO. All definitions are for a dedicated-key hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$, where $\mathcal{C} = \{0,1\}^n$ for some positive integer n, the key space $\mathcal{K}$ is some nonempty set and the message space $\mathcal{M} \subseteq \{0,1\}^*$ such that $\{0,1\}^m \subseteq \mathcal{M}$ for at least a positive integer m. For any $M \in \mathcal{M}$ and $K \in \mathcal{K}$, we use the notations $H_K(M)$ and $H(K,M)$ interchangeably. The advantage measures for an adversary A attacking H are defined in Fig. 1 for the ten security notions.

We say that H is (t, l, ϵ)-xxx, for xxx $\in$ {Coll, Sec[δ], aSec[δ], eSec, Pre[δ], aPre[δ], ePre}, if the advantage of any adversary A with time complexity at most t and using messages of length at most l, is less than ϵ, in attacking H in xxx sense. Note that four of the notions (namely, Sec[δ], aSec[δ], Pre[δ] and aPre[δ]) are parameterized by δ where $\{0,1\}^\delta \subseteq \mathcal{M}$. If H is a compression function (i.e. an FIL hash function), then parameter δ and the resource parameter l for the adversary will be the same as the fixed input length of the compression function and hence omitted from the notations. It is shown in [17] that the strength of provisional implications between different notions depends on the relative size of δ and the hash size n. For more related details we refer to [17].

$$\mathrm{Adv}_H^{Coll}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K}; (M, M') \xleftarrow{\$} A(K) : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M')\right]$$

$$\mathrm{Adv}_H^{Sec[\delta]}(A) = \Pr\left[\begin{array}{ll} K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^\delta; & \\ M' \xleftarrow{\$} A(K, M) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right]$$

$$\mathrm{Adv}_H^{aSec[\delta]}(A) = \Pr\left[\begin{array}{ll} (K, State) \xleftarrow{\$} A(); & \\ M \xleftarrow{\$} \{0,1\}^\delta; & \\ M' \xleftarrow{\$} A(M, State) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right]$$

$$\mathrm{Adv}_H^{eSec}(A) = \Pr\left[\begin{array}{ll} (M, State) \xleftarrow{\$} A(); & \\ K \xleftarrow{\$} \mathcal{K}; & \\ M' \xleftarrow{\$} A(K, State) & : \quad M \neq M' \ \wedge \ H_K(M) = H_K(M') \end{array}\right]$$

$$\mathrm{Adv}_H^{Pre[\delta]}(A) = \Pr\left[\begin{array}{ll} K \xleftarrow{\$} \mathcal{K}; M \xleftarrow{\$} \{0,1\}^\delta; Y \leftarrow H_K(M); & \\ M' \xleftarrow{\$} A(K, Y) & : \quad H_K(M') = Y \end{array}\right]$$

$$\mathrm{Adv}_H^{aPre[\delta]}(A) = \Pr\left[\begin{array}{ll} (K, State) \xleftarrow{\$} A(); & \\ M \xleftarrow{\$} \{0,1\}^\delta; Y \leftarrow H_K(M); & \\ M' \xleftarrow{\$} A(Y, State) & : \quad H_K(M') = Y \end{array}\right]$$

$$\mathrm{Adv}_H^{ePre}(A) = \Pr\left[(Y, State) \xleftarrow{\$} A(); K \xleftarrow{\$} \mathcal{K}; M' \xleftarrow{\$} A(K, State) : \quad H_K(M') = Y\right]$$

$$\mathrm{Adv}_H^{MAC}(A) = \Pr\left[K \xleftarrow{\$} \mathcal{K}; (M, tag) \xleftarrow{\$} A^{H_K(.)}() : \quad H_K(M) = tag \ \wedge \ \text{M not queried}\right]$$

$$\mathrm{Adv}_H^{PRF}(A) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{H_K(.)}() \Rightarrow 1\right] - \Pr\left[\rho \xleftarrow{\$} \mathrm{Func}(\mathcal{M}, \mathcal{C}) : A^{\rho(.)}() \Rightarrow 1\right]\right|$$

$$\mathrm{Adv}_H^{PRO}(A) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{H_K^h(.),\, h_K(.)}(K) \Rightarrow 1\right] - \Pr\left[K \xleftarrow{\$} \mathcal{K} : A^{\mathcal{F}(.),\, \mathcal{S}^{\mathcal{F}}(K,.)}(K) \Rightarrow 1\right]\right|$$

Fig. 1. Definitions of ten security notions for a hash function family H [17, 3]

For xxx $\in \{MAC, PRF\}$, we say that H is (t, q, l, ϵ)-xxx if the advantage of any adversary A having time complexity at most t and making at most q queries with maximum query length of l bits is at most ϵ.

PRO Notion. The definition of pseudorandom oracle preservation for a hash function was first considered by Coron et al. in [7] using the indifferentiability framework of Maurer et al. in [11], and further studied in the following works, e.g. in [5, 3]. The definition for the dedicated-key setting that we consider in this paper, as shown in Fig. 1, is due to Bellare and Ristenpart [3].

PRO is defined formally as follows. Adversary A is given '*oracles access*' to either the VIL hash function $H^h_K(.)$ and FIL random oracle $h_K(.)$, or a VIL random oracle $\mathcal{F}(.)$ and a simulator $\mathcal{S}^{\mathcal{F}}(K, .)$. A must differentiate between these two worlds and the simulator's goal is to mimic the FIL random oracle $h_K(.)$ in a way that convinces adversary A that $H^h_K(.)$ is $\mathcal{F}(.)$ (i.e. the two worlds become indifferentiable from A's view). The PRO advantage of an adversary A against H is defined as the difference between the probability that A outputs a one when given *oracle* access to $H^h_K(.)$ and $h_K(.)$ and the probability that it outputs a one when given oracle access to $\mathcal{F}(.)$ and the simulator $\mathcal{S}^{\mathcal{F}}(K, .)$. We say that H is $(t_A, t_{\mathcal{S}}, q_1, q_2, l, \epsilon)-$PRO, if for any adversary A having time complexity at most t and making at most q_1 queries from its first (left) oracle and q_2 queries from its second (right) oracle with maximal query length of l bits, there exits a simulator $\mathcal{S}$ with time complexity $t_{\mathcal{S}}$ such that makes $\mathrm{Adv}^{PRO}_H(A) < \epsilon$.

The Special Case of ROX Construction. In the case of a hash function obtained using ROX construction, the VIL hash function H utilizes two FIL random oracles RO_1 and RO_2 in its construction as well as a compression function h. In this case the definitions should be straightforwardly adapted to consider the existence of these two auxiliary FIL random oracles, namely adversary A will be also given oracle access to RO_1 and RO_2 and the number of queries from these oracles should also be considered as additional resource parameters for the adversary A. One also must use the generalized PRO notion based on the indifferentiability framework of [11] to involve these additional random oracles. We provide the required generalized definition in section 3.1 of this paper, following [11, 5]. Briefly saying, the simulator will have to simulate three random oracles for the adversary, namely the compression function itself (which for PRO notion is modeled as an FIL random oracle), as well as the two auxiliary random oracles used by the ROX in addition to h. More details are given in section 3.1 of this paper.

2.3 Hash Domain Extension

Assume that we have a compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ that can only hash messages of fixed length $(n + b)$ bits. A domain extension transform can use this compression function (as a black-box) to construct a hash function $H : \mathcal{K} \times \mathcal{M} \rightarrow \{0,1\}^n$, where the message space $\mathcal{M}$ can be either $\{0,1\}^*$ or $\{0,1\}^{<2^\lambda}$, for some positive integer λ (e.g. $\lambda = 64$). The key space $\mathcal{K}$

is determined by the construction of a domain extender. Clearly $log_2(|\mathcal{K}|) \geq k$, as H involves at least one invocation of h.

A *domain extension transform* comprises two functions: an injective *'padding function'* and an *'iteration function'*. First, the padding function $Pad : \mathcal{M} \rightarrow D_I$ is applied to an input message $M \in \mathcal{M}$ to convert it to the padded message $Pad(M)$ in a domain D_I. Then, the iteration function $f : \mathcal{K} \times D_I \rightarrow \{0,1\}^n$ uses the compression function h as many times as required, and outputs the final hash value. The full-fledged hash function H is obtained by combining the two functions. In the case of ROX transform both the padding algorithm (rox-pad) and the iteration algorithm need small-input random oracles as well.

The padding functions used in the Sh, ESh and ROX domain extension transforms are 'Strengthening', 'Strengthened Chain Shift' and 'rox-pad' defined as follows, where 2^λ is the maximum message length in bits (typically $\lambda = 64$) :

- **Strengthening:** $\text{pad}_s : \{0,1\}^{<2^\lambda} \rightarrow \bigcup_{L\geq 1} \{0,1\}^{Lb}$, where $\text{pad}_s(M) = M||10^p||\langle|M|\rangle_\lambda$ and p is the minimum number of 0's required to make the length of $\text{pad}_s(M)$ a multiple of block length.
- **Strengthened Chain Shift:** $\text{padCS}_s : \{0,1\}^{<2^\lambda} \rightarrow \bigcup_{L\geq 1} \{0,1\}^{Lb+b-n}$, where $\text{padCS}_s(M) = M||10^r||\ \langle|M|\rangle_\lambda\ ||0^p$, and parameters p and r are defined in two ways depending on the block length b. If $b \geq n + \lambda$ then $p = 0$, otherwise $p = b - n$. Then r is the minimum number of 0's required to make the padded message a member of $\{0,1\}^{Lb+b-n}$, for some positive integer L.
- **rox-pad:** $\text{rox-pad}^{RO_2} : \{0,1\}^{<2^\lambda} \rightarrow \bigcup_{L\geq 1} \{0,1\}^{L.b}$, where $RO_2 : \{0,1\}^k \times \{0,1\}^\lambda \times \{0,1\}^{\lceil log\ b\rceil} \rightarrow \{0,1\}^{2n}$ is an auxiliary FIL random oracle, and

  $$\text{rox-pad}^{RO_2}(M) = M||RO_2(M_{1\ldots k}, \langle|M|\rangle_\lambda, \langle 1\rangle)||RO_2(M_{1\ldots k}, \langle|M|\rangle_\lambda, \langle 2\rangle)||\cdots$$

  where the last block of padded message must contain at least $2n$ bits generated by RO_2 which implies adding a new final block just for padding if necessary.

The *iteration functions* for Sh, ESh and ROX transforms are shown in Fig. 2, where IV, IV_1, and IV_2 are some known initial values and $IV_1 \neq IV_2$. $RO_1 : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{\lceil log\ \lambda\rceil} \rightarrow \{0,1\}^n$ is a random oracle used by the ROX iteration function to generate required key masks.

The variable-input-length (VIL) hash function $H : \mathcal{K} \times \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$, for $H \in \{Sh, ESh, ROX\}$, obtained by applying Sh, ESh, or ROX domain extension transforms on a fixed-input-length (FIL) hash function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ is defined, respectively, as follows:

$$Sh(\mathbf{K}, M) = f_{Sh}(\mathbf{K}, \text{pad}_s(M)),\ \text{where}\ \mathbf{K} = K||K_0||\cdots||K_{t-1} \in \{0,1\}^{k+tn}$$

$$ESh(\mathbf{K}, M) = f_{ESh}(\mathbf{K}, \text{padCS}_s(M)), \text{where}\ \mathbf{K} = K||K_0||\cdots||K_{t-1} \in \{0,1\}^{k+tn}$$

$$ROX^{RO_1,RO_2}(K, M) = f_{ROX}^{RO_1(.)}(K, \text{rox-pad}^{RO_2(.)}(M)),\ \text{where}\ K \in \{0,1\}^k$$

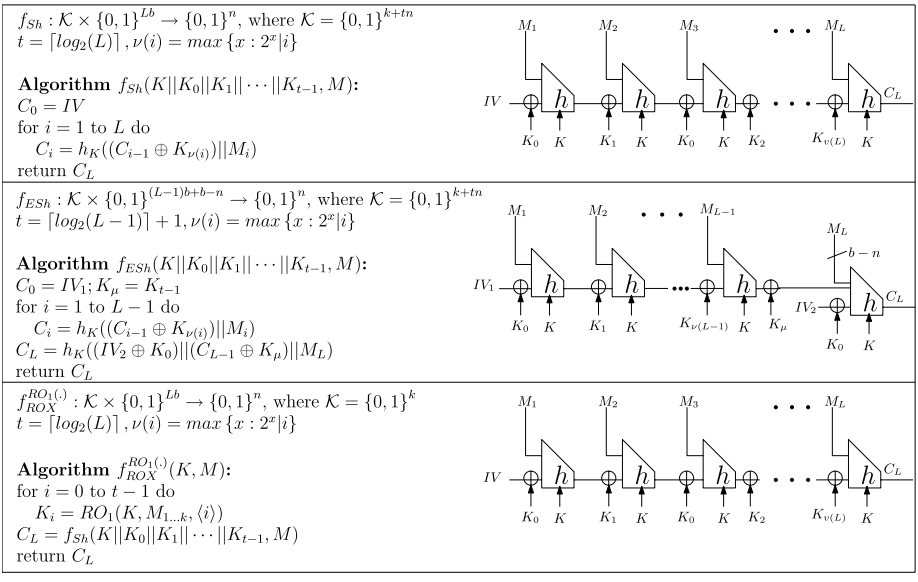


Fig. 2. Iteration functions of Shoup (Sh), Enveloped Shoup (ESh), and ROX transforms

3 Property Preservation Analysis of the Transforms

In this section we analyze property preserving capability of the ROX and ESh transforms in terms of the ten security notions defined in section 2.2, namely CR (Coll), Sec, aSec, eSec(TCR), Pre, aPre, ePre, MAC, PRF, PRO.

ROX was already shown in [1] to be able to preserve seven properties, namely: CR (Coll), Sec, aSec, eSec, Pre, aPre, and ePre. We complete a property-preservation analysis of ROX in regard to the other three important security notions (i.e. MAC, PRF, PRO) and gather both negative and positive results. On the negative side we show that ROX cannot preserve MAC and PRO notions. As a positive result we note that ROX can also preserve PRF. Next we investigate ESh transform. ESh was already shown in [3] to be able to preserve five properties, namely: CR (Coll), MAC, PRF, PRO, and TCR (eSec). We complete the property preservation analysis of ESh with respect to the remaining five properties among the ten notions by showing, as negative results, that ESh does not preserve four properties, namely Sec, aSec, Pre, aPre, and as a positive result we show that it can also preserve ePre.

3.1 Analysis of the ROX Transform

In this section we provide two negative results about PRO and MAC preservation and one positive result about PRF preserving capability of the ROX domain extension transform. Among these results, the negative result showing that ROX does not preserve PRO seems more interesting regarding the fact that ROX,

unlike all other hash domain extension transforms, uses random oracles in its construction.

Indifferentiability Analysis of the ROX Construction. Our aim is to show that ROX transform does not preserve PRO, i.e., the VIL hash function obtained using ROX transform is differentiable from a true VIL random oracle. We first provide the required generalization of PRO notion for the case of ROX construction based on the indifferentiability framework of [11]. Then we provide our negative result in Theorem 1.

For the purpose of PRO analysis (in the dedicated key hash setting [3]), the compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ is modeled as a family of FIL random oracles, i.e. $h_K : \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ is assumed to be an FIL random oracle for any value of the key K where $h_K(.) = h(K,.)$. We note that the ROX transform itself utilizes two additional FIL random oracles, namely RO_1 and RO_2 in generation of the key masks used in the iteration function and padding, respectively. Hence the VIL hash function $ROX^{h_K,RO_1,RO_2} : \{0,1\}^k \times \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$ will have access to three FIL random oracles, namely $h_K(.), RO_1(.)$ and $RO_2(.)$. According to the general indifferentiability framework of [11] which is used to define PRO in [7, 5, 3], adversary A will be given access to four oracles; namely, a VIL oracle $\mathcal{O}_1 : \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$, and three FIL oracles as $\mathcal{O}_2 : \{0,1\}^{n+b} \rightarrow \{0,1\}^n$, $\mathcal{O}_3 : \{0,1\}^k \times \{0,1\}^k \times \{0,1\}^{\lceil log\ \lambda \rceil} \rightarrow \{0,1\}^n$ and $\mathcal{O}_4 : \{0,1\}^k \times \{0,1\}^\lambda \times \{0,1\}^{\lceil log\ b \rceil} \rightarrow \{0,1\}^{2n}$, and must differentiate between the following two worlds:

- **World 1:** A random key $K \xleftarrow{\$} \{0,1\}^k$ is selected. The oracles are set as $\mathcal{O}_1(.) = ROX^{h_K,RO_1,RO_2}(K,.)$, $\mathcal{O}_2(.) = h_K(.)$, $\mathcal{O}_3(.) = RO_1(.)$, $\mathcal{O}_4(.) = RO_2(.)$. A is given K as input and has access to the four oracles.
- **World 2:** A random key $K \xleftarrow{\$} \{0,1\}^k$ is selected. $\mathcal{O}_1(.) = \mathcal{F}(.)$ where $\mathcal{F} : \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$ is a true VIL random oracle. A simulator $\mathcal{S}^{\mathcal{F}}(K) = (\mathcal{S}_1^{\mathcal{F}}(K), \mathcal{S}_2^{\mathcal{F}}(K), \mathcal{S}_3^{\mathcal{F}}(K))$, having access to the oracle $\mathcal{F}(.)$ and receiving K as input, simulates the role of the three FIL random oracles for the adversary. That is, in this world when adversary queries the first oracle (i.e. the VIL oracle) $\mathcal{O}_1$ the response comes from the true VIL random oracle $\mathcal{F}(.)$, but when adversary queries any of the three FIL oracles $\mathcal{O}_2(.), \mathcal{O}_3(.)$ and $\mathcal{O}_4(.)$, the queries are forwarded to the the simulator $\mathcal{S}$. Simulator will respond to these queries trying to mimic the oracles $h_K(.), RO_1(.)$ and $RO_2(.)$, respectively, by its sub-algorithms $\mathcal{S}_1, \mathcal{S}_2$ and $\mathcal{S}_3$ in a way that convinces A that $\mathcal{O}_1(.)$ is $ROX^{h_K,RO_1,RO_2}(K,.)$ although it is now actually $\mathcal{F}(.)$.

Let $H_K(.) = H(K,.) = ROX^{h_K,RO_1,RO_2}(K,.)$. The PRO advantage of the adversary in differentiating H from $\mathcal{F}$ is defined as follows:

$$\epsilon = \mathrm{Adv}_H^{PRO}(A) = \big|\Pr\left[A^{\mathcal{O}_1,\,\mathcal{O}_2,\,\mathcal{O}_3,\,\mathcal{O}_4}(K) \Rightarrow 1 \,|\text{World 1}\right] - \Pr\left[A^{\mathcal{O}_1,\,\mathcal{O}_2,\,\mathcal{O}_3,\,\mathcal{O}_4}(K) \Rightarrow 1 \,|\text{World 2}\right]\big|$$

We say that an adversary A is a $(t_A, t_{\mathcal{S}}, q_1, q_2, q_3, q_4, l, \epsilon)$-differentiating adversary against H if its PRO advantage is at least ϵ against any simulator $\mathcal{S}$ having time complexity at most $t_{\mathcal{S}}$, where the time complexity of the adversary is at most t_A, the number of queries from i-th oracle is at most q_i (for $1 \leq i \leq 4$) and the length of each query is at most l bits.

Now we are ready to state our negative result which shows the inability of ROX to preserve PRO.

Theorem 1 (Negative Result: PRO). *ROX domain extension transform does not preserve pseudorandom oracle.*

Proof. We show a $(c, t_{\mathcal{S}}, 2, 1, 1, 2, 3b-2n, 1-2^{-n})$-differentiating adversary against H, i.e. A has overwhelming PRO advantage of $1-2^{-n}$ in differentiating the VIL hash function $ROX^{h_K, RO_1, RO_2} : \{0,1\}^{<2^{\lambda}} \rightarrow \{0,1\}^n$ from a true VIL random oracle $\mathcal{F} : \{0,1\}^{<2^{\lambda}} \rightarrow \{0,1\}^n$, with respect to any simulator $\mathcal{S}$ with *arbitrary time complexity* $t_{\mathcal{S}}$. A has time complexity $t_A = c$, where c is a small constant, and it asks only: two queries from the first oracle $\mathcal{O}_1$, one query from the second oracle $\mathcal{O}_2$, one query from the third oracle $\mathcal{O}_3$ and two queries from the fourth oracle $\mathcal{O}_4$. The maximum query length is $3b - 2n$ bits.

Adversary A acts as follows:

1. $M \xleftarrow{\$} \{0,1\}^{2b-2n}$ and $Y \leftarrow \mathcal{O}_1(M)$;
2. $IP \leftarrow \mathcal{O}_4(M_{1...k}, \langle 2b-2n \rangle_{\lambda}, \langle 1 \rangle)$;
3. $M' \xleftarrow{\$} \{0,1\}^{b-2n}$ and $Z \leftarrow \mathcal{O}_1(M||IP||M')$;
4. $K_0 \leftarrow \mathcal{O}_3(K, M_{1...k}, \langle 0 \rangle)$;
5. $OP \leftarrow \mathcal{O}_4(M_{1...k}, \langle 3b-2n \rangle_{\lambda}, \langle 1 \rangle)$;
6. $Z' \leftarrow \mathcal{O}_2\big((Y \oplus K_0)||M'||OP\big)$;
7. If $Z = Z'$ then return 1 (i.e. guess that it is World 1) else return 0 (i.e. guess that it is World 2)

From the construction of the ROX hash function and the description of the World 1, it can be seen that $\Pr\left[A^{H_K(.),\, h_K(.),\, RO_1(.),\, RO_2(.)}(K) \Rightarrow 1\right] = 1$. We claim that $\Pr\left[A^{\mathcal{F}(.),\, \mathcal{S}_1^{\mathcal{F}}(K),\, \mathcal{S}_2^{\mathcal{F}}(K),\, \mathcal{S}_3^{\mathcal{F}}(K)}(K) \Rightarrow 1\right] = 2^{-min\{n, 2b-2n-k\}}$. This can be seen by noting that in World 2, the only queries that the simulator $\mathcal{S}^{\mathcal{F}}(K) = (\mathcal{S}_1^{\mathcal{F}}(K), \mathcal{S}_2^{\mathcal{F}}(K), \mathcal{S}_3^{\mathcal{F}}(K))$ can see and must respond to are the queries from $\mathcal{O}_2, \mathcal{O}_3$, and $\mathcal{O}_4$ oracles. It is worth reminding that, the simulator cannot see query-response sequence between the adversary A and the first oracle $\mathcal{O}_1$ due to the definition of indifferentiability [7, 5, 3], as these queries are answered directly by the true VIL random oracle $\mathcal{F}$ in World 2. Hence referring to the description of the adversary A, it can be seen that the simulator $\mathcal{S}$ just is given the first k bits of the first message M, i.e. $M_{1...k}$, and has *no information* about the remaining $2b - 2n - k$ bits of the message M. Hence to make A output a one (i.e. to fool A), simulator $\mathcal{S}$ must either guess these $2b - 2n - k$ unknown bits of M (with success probability of $2^{-(2b-2n-k)}$) or guess the correct value of Z (with success probability of 2^{-n}) in order to be able to provide a correct value Z' at step 6 of A's differentiating attack. (Note that the success probability of

$\mathcal{S}$ in guessing the correct value of these unknown bits is independent of the time complexity of $\mathcal{S}$, i.e. $t_{\mathcal{S}}$, as $\mathcal{S}$ has no information about these bits.) So, we have $\mathrm{Adv}_H^{PRO}(A) = 1 - 2^{-min\{n,2b-2n-k\}}$. It remains to verify that $2b - 2n - k \geq n$. According to the construction of $rox - pad^{RO_2}$ it must be the case that $b \geq 2n$ and referring to [1], typical values for k and n are suggested as $k = 80$ bits and $n = 160$ bits for an 80-bit security level, i.e. we have $k \approx n/2$. Hence $min\{n, 2b - 2n - k\} = n$ and $\mathrm{Adv}_H^{PRO}(A) = 1 - 2^{-n}$ as claimed. □

MAC Preservation Analysis of the ROX. We show that the ROX transform does not preserve MAC (unforgeability). This is done by providing as a counterexample, a compression function h which is a secure MAC but for which the VIL hash function obtained by using ROX transform will be insecure in MAC sense.

Assume that there is a compression function $g : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^{n-1}$ which is $(t, q, \epsilon)-$MAC. Consider the following construction from [3] for a compression function $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$, where $s = \lceil log_2(n) \rceil$:

$$h_K(C||M) = \begin{cases} g_K(C||M)||C[i] & \text{if } M = \langle i \rangle_s \,||0^{b-s} \text{ for } i \in \{1, \cdots, n-1\} \\ g_K(C||M)||C[n] & \text{otherwise} \end{cases}$$

It is shown in [3] that if g is $(t, q, \epsilon)-$MAC then h will be $(t-cq, q, \epsilon)-$MAC, where c is a small constant. Note that h leaks the i-th bit of its chaining variable input C to the output if its input block M equals to $\langle i \rangle_s \,||0^{b-s}$, for $i \in \{1, \cdots, n-1\}$, and otherwise h leaks the n-th (i.e. the last) bit of C.

We use this counterexample to prove the following negative result.

Theorem 2 (Negative Result: MAC). *ROX domain extension transform does not preserve MAC.*

Proof. Consider the above counterexample function h as an FIL MAC. We show that the VIL function $H(K, .) = ROX^{h,RO_1,RO_2}(K, .)$ obtained by applying the ROX transform on this FIL MAC h will not be a secure MAC. This is done by describing an adversary A which can break $H(K, .)$ in MAC sense, with success probability $\approx \frac{1}{4}$ and whose computational resources are only: one query from the random oracle $RO_2(.)$, $2(n-1)$ queries from the function $H_K(.)$ with maximum length of each query $4b - 2n$ bits, and a constant time complexity proportional to n.

The idea behind the construction of the adversary A is the same "length reduction attacks" used in [3] to show that Sh transform is not MAC preserving. We adapt the attack for the case of ROX transform by considering the special padding function rox-pad^{RO_2} used by the ROX. The algorithm for the adversary A is shown in Fig. 3. It has oracle access to $H_K(.)$ and $RO_2(.)$. Note that in definition of the MAC security notion, the key K is considered secret from the adversary and hence A cannot compute $H_K(.)$ directly.

It selects an all zero string of length $2b - 2n$ bits as $M = 0^{2b-2n}$, for which it tries to return a valid tag T under $H_K(.)$. To this aim, A first queries RO_2 as $IP \leftarrow RO_2(M_{1...k}, \langle 2b - 2n \rangle_\lambda, \langle 1 \rangle)$ to get the 2n-bit response IP (IP stands for

'internal padding'). It then queries oracle $H_K(.)$ on $n-1$ messages, each of length $4b-2n$ bits, constructed as $QH_i = M||IP||\langle i \rangle_s ||0^{b-s}||0^{b-2n}$, and it receives the response $Y_i = H_K(QH_i)$, for $i \in \{1, \cdots, n-1\}$. Let y_i denote the last bit of Y_i, i.e. $y_i = Y_i[n]$. According to the ROX construction and the structure of the counterexample compression function h, the value of the bit y_i will be computed as $y_i = H_K(M)[i] \oplus K_0[i] \oplus K_2[n]$ with overwhelming probability of at least $1 - 2^{-min\{2n, b-s\}}$. This can be seen from (the Top-Right diagram in) Fig. 3 noting that the final block contains $2n$ bits of random padding string (denoted by OP) as its last bits and hence this final block input to the compression function h is not equal to $\langle i \rangle_s ||0^{b-s}$ with probability at least $1 - 2^{-min\{2n, b-s\}}$, for $i \in \{1, \cdots, n-1\}$, and therefore h will leak the last bit (i.e. n-th bit) of its chaining variable input to the output Y_i. Therefore with probability at least $1 - 2^{-min\{2n, b-s\}}$, the variable y_i (for $1 \leq i \leq n-1$) contains the i-th bit of the tag T for the message M (i.e. $T[i] = H_K(M)[i]$) masked with the unknown key bits $K_0[i]$ and $K_2[n]$. For typical values of b and n (say $b = 512, n = 160$) we can make $1 - 2^{-min\{2n, b-s\}} \approx 1$ and so we assume that this probability is approximately one, to prevent unnecessary complexity in the analysis.

Now A tries to peel off the unknown key bits $K_0[i]$, for $1 \leq i \leq n-1$. It queries $H_K(.)$ on $n-1$ messages, each of length $2b-2n$ bits, constructed as $qH_i = \langle i \rangle_s ||0^{b-s}||0^{b-2n}$ and receives the response $Z_i = H_K(qH_i)$, for $i \in \{1, \cdots, n-1\}$. Let z_i denote the last bit of the Z_i, i.e. $z_i = Z_i[n]$. According to the description of $H_K(.)$ and the structure of the counterexample function h, the value of bit z_i will be computed as $z_i = H_K(qH_i)[i] = IV[i] \oplus K_0[i] \oplus K_1[n]$ with overwhelming probability of at least $1 - 2^{-2n}$. This can be seen from (the Bottom-Right diagram in) Fig. 3 noting that the final block contains $2n$ bits of random padding string (denoted by OP'_i) as its last bits and hence this final block input to the compression function h is not equal to $\langle i \rangle_s ||0^{b-s}$ with probability at least $1 - 2^{-2n}$, for any $i \in \{1, \cdots, n-1\}$, and hence h will leak the last bit (i.e. n-th bit) of its chaining variable input to the output Z_i. For a typical value of n (say $n = 160$ as in SHA-1) we can make $1 - 2^{-2n} \approx 1$ and so we assume that this (overwhelming) probability is approximately one.

Now for each $i \in \{1, \cdots, n-1\}$ adversary builds a variable $t_i = y_i \oplus z_i \oplus IV[i]$ whose value will be $t_i = H_K(M)[i] \oplus K_1[n] \oplus K_2[n]$ (with overwhelming probability of at least $(1 - 2^{-2n})^2 \approx 1$ for typical values of n, say $n = 160$). Note that the value of the remaining unknown masking bit $K_1[n] \oplus K_2[n]$ is independent of the index i, i.e. it is the same for all t_i and therefore adversary can guess this unknown value with probability $1/2$. That is, for a random guess $\alpha \xleftarrow{\$} \{0,1\}$ with probability $1/2$ the value $t_i \oplus \alpha$ will be equal to $H_K(M)[i] = T[i]$ (i.e. the correct tag value), for all $i \in \{1, \cdots, n-1\}$. It just remains to compute the last bit of the tag T, i.e. $T[n]$, but A just guesses this one bit and with probability $1/2$ this guess will be correct. Hence the adversary A computes a correct MAC tag T for the message M under $H_K(.)$ with probability $1/4$ (or more precisely with probability $\frac{1}{4}(1-(n-1)2^{-2n+1})$, which for any typical value of hash size n, say $n = 160$, will be $\approx 1/4$). Note that the message M never is queried from $H_K(.)$ in the attack and so this is a valid forgery attack.

Algorithm $A^{H_K(.),RO_2(.)}$:
$M \leftarrow 0^{2b-2n}$
$IP \leftarrow RO_2(M_{1...k}, \langle 2b-2n \rangle_\lambda, \langle 1 \rangle)$
for $i = 1$ to $n-1$ do
  $Y_i \leftarrow H_K(M||IP||\langle i \rangle_s||0^{b-s}||0^{b-2n})$
  $y_i \leftarrow Y_i[n]$
for $i = 1$ to $n-1$ do
  $Z_i \leftarrow H_K(\langle i \rangle_s||0^{b-s}||0^{b-2n})$
  $z_i \leftarrow Z_i[n]$
$\alpha \xleftarrow{\$} \{0,1\}$
for $i = 1$ to $n-1$ do
  $t_i \leftarrow y_i \oplus z_i \oplus IV[i]$
  $T[i] \leftarrow t_i \oplus \alpha$
$T[n] \xleftarrow{\$} \{0,1\}$
$T \leftarrow T[1] \cdots T[n]$
return (M, T)

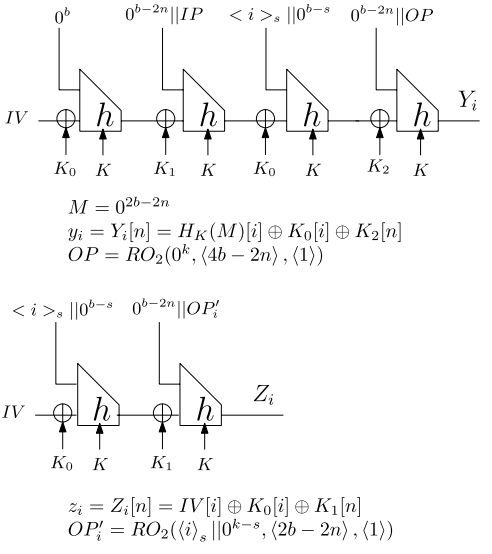


Fig. 3. (Left) Description of the algorithm for an adversary A against the VIL function $H_K(.) = ROX^{RO_1,RO_2}(K,.)$ based on the (counterexample) FIL MAC function h. **(Right)** The structure of the queries from $H_K(.)$ and computation of the responses according to the ROX construction.

Referring to the algorithm for A it is seen that A is quite efficient as its computational resources are: one query from the random oracle $RO_2(.)$, $2(n-1)$ queries from the function $H_K(.)$ with maximum length of each query $4b-2n$ bits, and a constant time complexity proportional to n which can be easily determined from the description of A. □

Theorem 3 (Positive Result: PRF). *ROX domain extension transform preserves PRF.*

Proof. This result is just a straightforward corollary of a theorem in [3] (Theorem 5, page 407) showing that *in the dedicated-key hash function setting* (where the compression function is a keyed hash function) Merkle-Damgård transform and all of its variants including Shoup are PRF preserving transforms. As shown in [3, 4] even if the key masks $(K_0, K_1, \cdots)$ in Shoup construction are made public and only the key (K) for the compression function is kept secret then Shoup transform will still be PRF preserving. Clearly a special case will be putting the value of all key masks to zero in which case Shoup iteration will be the same as Merkle-Damgård iteration which is a PRF preserving transform *in the dedicated-key hash setting* [4]. We note that the iteration function of the ROX transform is exactly the same as Shoup where the key masks are generated using a random oracle (Refer to Fig. 2). □

3.2 Analysis of the ESh Transform

The following theorems show our results about the ESh. We show both negative results and a positive result about property preservation capability of ESh.

Theorem 4 (Negative Results: Sec, aSec, Pre, and aPre). *ESh domain extension transform does not preserve any of the Sec, aSec, Pre, and aPre security properties.*

Proof. The proof is done by showing, as a counterexample, a compression function which is secure in xxx sense for xxx $\in$ {Sec, aSec, Pre, aPre} but for which the full-fledged hash function obtained using ESh domain extension transform is completely insecure in xxx sense for xxx $\in$ {Sec[δ], aSec[δ], Pre[δ], aPre[δ]} and for *any value of the parameter* $\delta < 2^{\lambda}$ (remember that 2^{λ} is the maximum input message length in bits).

Referring to the description of the strengthened chain shift padding function (padCS_s) used in ESh transform, we consider the following two cases depending on the sizes of the parameters b, n and λ (note that for ESh, $b \geq n$ and typical value of $\lambda = 64$):

- Case 1: if $b \geq n + \lambda$ then $\text{padCS}_s(M) = M||10^r||\ \langle |M| \rangle_{\lambda}$
- Case 2: if $b < n + \lambda$ then $\text{padCS}_s(M) = M||10^r||\ \langle |M| \rangle_{\lambda}\ ||0^{b-n}$

Assume that there is a compression function $g : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^{n-1}$ which is (t, ϵ)-xxx, where xxx is any of the four properties in {Sec, aSec, Pre, aPre}. For Case 1 and Case 2, respectively, consider the following two compression functions $h_1 : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ and $h_2 : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$:

$$h_1(K, M) = \begin{cases} 0^n & \text{if } M_{n+b-\lambda+1 \ldots n+b} = \langle \delta \rangle_{\lambda} \\ g(K, M)||1 & \textit{otherwise} \end{cases}$$

$$h_2(K, M) = \begin{cases} 0^n & \text{if } M_{2n+1 \ldots n+b} = 0^{b-n} \\ g(K, M)||1 & \textit{otherwise} \end{cases}$$

Construction of the counterexamples h_1 and h_2 are inspired from the counterexamples used in [5, 1] where we make some small modifications in the conditions defining these two functions to consider the effect of padCS_s padding in ESh transform. To complete the proof of the theorem we prove and combine the following two lemmas. The first lemma shows that the compression functions h_1 and h_2 inherit security properties xxx $\in$ {Sec, aSec, Pre, aPre} from the compression function g and the second lemma shows that ESh transform cannot preserve these four properties while extending the domain of h_1 or h_2 compression functions. Note that only one of these compression functions are used depending on which of the two conditions specified in Case 1 and Case 2 above are the case.

Lemma 1. *If g is (t, ϵ)-xxx then h_1 is $(t, \epsilon + 2^{-\lambda})$-xxx and h_2 is $(t, \epsilon + 2^{-(b-n)})$-xxx, for any of the notions* xxx $\in$ {*Sec, aSec, Pre, aPre*}.

Proof. The proof can be found in the full version of this paper [16].

Lemma 2. *For any* xxx $\in$ {*Sec[δ], aSec[δ], Pre[δ], aPre[δ]*}*, and for any value of* $\delta < 2^\lambda$*, there is a simple adversary which can break the domain extended hash function* $ESh(\mathbf{K}, M) = f_{ESh}(\mathbf{K}, padCS_s(M))$ *using* h_1 *or* h_2 *as the compression function.*

Proof. Considering the description of the padCS$_s$ and counterexample compression functions h_1 and h_2, we have $ESh(\mathbf{K}, M) = 0$ for any $M \in \{0,1\}^\delta$. Hence, in Pre[δ] and aPre[δ] attacks adversary A just needs to output any arbitrary $M' \in \{0,1\}^\delta$ and wins with probability one. Similarly, in Sec[δ] and aSec[δ] attacks A only needs to output any $M' \in \{0,1\}^\delta$ which is different from the challenge message $M \in \{0,1\}^\delta$ and wins with probability one. That is, the VIL hash function $ESh : \mathcal{K} \times \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$, defined as $ESh(\mathbf{K}, M) = f_{ESh}(\mathbf{K}, \text{padCS}_s(M))$ using h_1 or h_2 is completely insecure in xxx sense for xxx $\in$ {Sec[δ], aSec[δ], Pre[δ], aPre[δ]} and for any value of the parameter δ, where $\delta < 2^\lambda$. □

Theorem 5 (Positive Result: ePre). *If the compression function* $h : \{0,1\}^k \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$ *is* $(t, \epsilon)-ePre$ *then the full-fledged hash function* $ESh : \mathcal{K} \times \{0,1\}^{<2^\lambda} \rightarrow \{0,1\}^n$ *defined as* $ESh(\mathbf{K}, M) = f_{ESh}(\mathbf{K}, padCS_s(M))$ *will be* $(t', \epsilon)-ePre$*, where* $t' = t - c$*, for a small constant* c*.*

Proof. Assume that there is an adversary A against ePre property of the hash function ESh with time complexity t' and advantage ϵ'. We construct an adversary B which can break the compression function h in ePre sense with the same advantage (i.e. $\epsilon = \epsilon'$) and whose time complexity is that of A plus a small constant time (i.e. $t = t' + c$). Adversary B runs A and on receiving the value of Y from A outputs the same value Y as its own target hash value in the first phase of ePre game. B receives K (the random key for h), generates $K_0||\cdots||K_{t-1} \xleftarrow{\$} \{0,1\}^{t.n}$, where $t = \lceil log_2(2^\lambda/b) \rceil + 1$ and sends the key $K||K_0||\cdots||K_{t-1}$ to adversary A as the key for the full-fledged hash function ESh. Note that because B by this phase just knows the Y and does not know the length of the input message to the hash function ESh, it generates a key string of maximum required length for the XOR masks, i.e. $t.n$ bits, for $t = \lceil log_2(2^\lambda/b) \rceil + 1$ where $2^\lambda/b$ is the maximum possible input length in blocks and n is the hash size. Now on receiving the message M' from A (which is to be a preimage for Y under ESh, i.e. $Y = ESh(K||K_0||\cdots||K_{t-1},\ M')$), adversary B simply outputs the value $(IV_2 \oplus K_0)||(C_{L-1} \oplus K_\mu)||M'_L$ as a preimage for Y (refer to Fig. 2) which is the input to the final application of the compression function h in the construction of ESh hash function. Clearly B wins whenever A wins. The time complexity of B is that of A plus the time required to generate t random n-bit keys, where $t = O(\lambda)$ (typically $\lambda = 64$), and the time to compute the hash function ESh on a message of length $|M'|$. □

4 Can We Preserve All Properties?

In the previous section we showed that the ROX transform, which is a random oracle variant of Shoup, does not preserve MAC and PRO notions and also we showed that the Enveloped Shoup (ESh) transform does not preserve Sec, aSec, Pre, aPre notions. An immediate question arises from this analysis is that whether we can preserve all the ten properties simultaneously by a new domain extension transform.

Using FIL Random Oracles. If we are allowed to use some FIL Random Oracles in our construction in the same way that ROX does (i.e. uses FIL random oracles just for padding and generation of masking keys), then our analysis in the previous section hints us toward a candidate for such a ten-property-preserving transform by just mixing components from both ESh and ROX. We notice that, as shown in Fig. 2, ROX utilizes Shoup's iteration as its underlying iteration function and uses FIL random oracles for generation of masking keys and padding function. Hence the natural candidate for a ten property preserving transform in random oracle model can be a random oracle variant of ESh with some necessary adaptation in a similar way that ROX is obtained from Sh. We call such a transform as Random-Oracle Enveloped Shoup (RO-ESh). For more details and analysis of properties of RO-ESh we refer to the full version of this paper in [16].

Without Any Random Oracle. As it was shown in analysis of ESh, as a *standard model* transform, the four properties, namely; Pre, aPre, Sec, and aSec are not preserved by ESh. It appears to be a crux to preserve these four properties (simultaneously) in the standard model by an efficient transform.

Regarding the Sec property, Andreeva and Preneel [2] proposed a keyed transform to extend the domain of a keyless compression function. The proposed dedicated-key hash construction is CR and Sec secure provided that the underlying keyless compression function is, respectively, CR or Sec (here CR and Sec are defined for a dedicated-key hash function and a keyless compression function, respectively). Unfortunately the proposed scheme cannot be shown to be Pre secure in standard model and only a random oracle argument is provided in [2] for its Pre property. For Pre notion, the only transform in the standard model that is pointed out in [1] to be Pre preserving is the XOR Tree scheme, but it does not preserve aPre and aSec [1]. For aSec and aPre notions, there is currently no transform in the literature that can *preserve* aSec and aPre in the standard model.

We remind that a hash domain extension transform *preserves a property P* if the constructed VIL hash function provably possesses P assuming that the underlying compression function satisfies *the same property* P. This is different from a scenario where one proves that the VIL hash function has property P if its underlying compression function satisfies a different property P′, e.g. [19], or a collection of different assumptions, e.g. [10, 9].

5 Conclusion

In this paper, we analyzed two recently proposed MPP hash domain extension transforms, namely the Random-Oracle-XOR (ROX) transform and the Enveloped Shoup (ESh) transform. We showed that ROX *does not preserve* MAC and PRO notions, but it preserves PRF. We also showed that ESh *does not preserve* Sec, aSec, Pre, and aPre, but it preserves ePre. Our results complete the MPP analysis of both ROX and ESh transforms in regard to all ten security notions of interest, namely CR, Sec, aSec, eSec (TCR), Pre, aPre, ePre, MAC, PRF, PRO, and provide the full picture of their MPP capabilities. An interesting open question is that whether one can (simultaneously) preserve the remaining four properties, namely Pre, aPre, Sec, and aSec in the *standard model* with an efficient transform.

Acknowledgments. We would like to thank the anonymous reviewers of ACISP 2009 for their comments and suggestions.

References

[1] Andreeva, E., Neven, G., Preneel, B., Shrimpton, T.: Seven-Property-Preserving Iterated Hashing: ROX. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 130–146. Springer, Heidelberg (2007)

[2] Andreeva, E., Preneel, B.: A Three-Property-Secure Hash Function. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. Workshop Records, pp. 208–224 (2008)

[3] Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 399–410. Springer, Heidelberg (2007)

[4] Bellare, M., Ristenpart, T.: Hash Functions in the Dedicated-Key Setting: Design Choices and MPP Transforms. Cryptology ePrint Archive, Report 2007/271 (2007), http://eprint.iacr.org/

[5] Bellare, M., Ristenpart, T.: Multi-Property-Preserving Hash Domain Extension and the EMD Transform. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 299–314. Springer, Heidelberg (2006)

[6] Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)

[7] Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård Revisited: How to Construct a Hash Function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (2005)

[8] Damgård, I.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)

[9] Dodis, Y., Puniya, P.: Getting the Best Out of Existing Hash Functions; or What if We Are Stuck with SHA? In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 156–173. Springer, Heidelberg (2008)

[10] Halevi, S., Krawczyk, H.: Strengthening Digital Signatures Via Randomized Hashing. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 41–59. Springer, Heidelberg (2006)

[11] Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
[12] Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
[13] Mironov, I.: Collision-Resistant No More: Hash-and-Sign Paradigm Revisited. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 140–156. Springer, Heidelberg (2006)
[14] Mironov, I.: Hash Functions: From Merkle-Damgård to Shoup. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 166–181. Springer, Heidelberg (2001)
[15] Naor, M., Yung, M.: Universal One-Way Hash Functions and Their Cryptographic Applications. In: STOC 1989, pp. 33–43. ACM Press, New York (1989)
[16] Reyhanitabar, M.R., Susilo, W., Mu, Y.: Analysis of Property-Preservation Capabilities of the ROX and ESh Hash Domain Extenders. Cryptology ePrint Archive, Report 2009/170 (2009)
[17] Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
[18] Shoup, V.: A Composition Theorem for Universal One-Way Hash Functions. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 445–452. Springer, Heidelberg (2000)
[19] Yasuda, K.: How to Fill Up Merkle-Damgård Hash Functions. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 272–289. Springer, Heidelberg (2008)

Characterizing Padding Rules of MD Hash Functions Preserving Collision Security

Mridul Nandi

National Institute of Standards and Technology
mridul.nandi@gmail.com

Abstract. This paper characterizes collision preserving padding rules and provides variants of Merkle-Damgård (MD) which are having less or no overhead costs due to length. We first show that suffix-free property of padding rule is necessary as well as sufficient to preserve the collision security of MD hash function for an arbitrary domain $\{0,1\}^*$. Knowing this, we propose a simple suffix-free padding rule padding only $\log |M|$ bits for a message M, which is less than that of Damgard's and Sarkar's padding rules. We also prove that the length-padding is not absolutely necessary. We show that a simple variant of MD with 10^d-padding (or any injective padding) is collision resistant provided that the underlying compression function is collision resistant after chopping the last-bit. Finally, we design another variant of MD hash function preserving all three basic security notions of hash functions, namely collision and (2nd) preimage, which is an improvement over a recently designed (SAC-08) three-property preserving hash function.

Keywords: MD hash function, padding rule, suffix-free, collision resistant.

1 Introduction

Hash function has become an essential object in many cryptographic protocols [4] particularly in signature schemes [2,6,11]. It takes an input from a message space $\mathcal{M}$ (usually $\{0,1\}^*$ or $\{0,1\}^{\leq 2^s-1}$ for some s) and it outputs a t-bit string for a fixed t. The hash function plays role of preprocessor in many applications so that one can work with t-bit $H(M)$ instead of an arbitrary sized M, which essentially helps us to keep design of a protocol simple and efficient. In most cases, securities of these protocols rely on the *collision resistance* property (it is hard to find two different messages with same hash value) of the hash function. The most popular design of a hash function is Merkle-Damgard [5,10] or MD hash function where a compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$ is designed first. Given a message, some additional bits may be padded to it so that it can be partitioned into several blocks of size b. The compression function is then sequentially applied to an initial value and to all blocks of the padded message. It seems difficult to design a hash function, based on only simple logical or/and arithmetical operations, which can provide absolute collision security (or provable collision security like discrete log

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 171–184, 2009.

based hash function [7]). But it is well known that the MD hash functions with length strengthening (padding length of the input) preserves collision security (i.e. the hash function is collision resistant if so is the compression function). So we can at least reduce the infeasibility assumption of a hash function to a smaller domain compression function.

Padding rule is essential for MD hash function (proposed by both Damgard [5] and Merkle [10] independently in crypto-1989). However they used different padding rules. Merkle's padding rule can not handle arbitrary length messages. Sarkar [15] recently introduced a padding rule which can handle arbitrary messages and the number of padding bits is $O(\log |M| \log^* |M|)$ for some slowly growing function $\log^*$ defined in [15]. This is asymptotically less that that of Damgard's padding rule where $O(|M|)$ bits are padded. Note, if the size of padded bits is more, it may cost more invocations of the underlying compression function. So, in terms of efficiency of hash function, one should try to keep size of pad as small as possible. Any arbitrary padding may not be good as the hash function is desired to preserve collision security. Clearly, injectivity is a basic requirement of a padding rule. A padding rule is said to preserve collision security (for MD) if the hash function with this padding rule preserves collision security. So, it is worthwhile to characterize all padding rules preserving collision security.

Our Contribution. In this paper, we first show that *suffix-free property is both necessary and sufficient to preserve collision security for MD hash function.* Damgard in [5] mentioned prefix-free padding rules. Stinson [16], Bellare and Rogaway [3] mentioned suffix-free property while proving collision preserving of particular padding rules. Even though sufficiency of suffix-free padding rule seems intuitive, we do not know any paper proving it. Some observations on Merkle's padding rule can be found in [8]. On the other hand, the necessity of the suffix-free property is non-trivial. We propose *a simple efficient suffix-free padding rule, padding* $O(\log(|M|))$ *bits, which can handle arbitrary messages.* We see a comparison of new padding rules with known padding rules in Table 1.

Let a t-bit compression function f is collision resistant in the first $(t-1)$-bits (i.e. collision resistant after chopping the last bit). We show that a simple variant of MD hash function (converting 0^t chaining value (if any) into $0^{t-1}1$) without any length-padding (any injective padding such as 10^d-padding works) is collision resistant. We actually prove a stronger statement which says that any collision of the new hash function reduces to either a collision of f or a collision of the first $(t-1)$ bits of f **with the collision value** 0^{t-1}. Thus, we are able to remove overhead costs due to length.

We also provide *an improved three property (collision, (2nd) preimage) preserving salted hash function which is a variant of MD hash function and is more efficient than recently proposed hash function* [1] in terms of salt size.

Organization of the paper. We first give an overview of the security notions of a hash function and padding rules of MD hash functions in section 2. In section 3, we characterize the collision preserving padding rules for any fixed initial value. We also have provided simple examples of padding rule in the same

section. In the following section, we prove a simple variant can completely avoid length-padding and still have collision security under a reasonable additional assumption. In section 5, we study an improved variant of BCM (backward chaining mode) hash function which preserves all basic three security notions of a hash function.

2 Overview of MD Hash Function, Padding Rule

A hash function $H : \mathcal{M} \to \{0,1\}^t$ is called a *collision resistant* [14,17] hash functions if *it is "hard" to find a collision pair* (M, M') *i.e.,* $M \neq M'$ *such that* $H(M) = H(M')$. We define collision-advantage of an algorithm A as

$$\mathbf{Adv}_H^{\text{coll}}(A) := \mathsf{Pr}[A \to (M, M') : H(M) = H(M'), M \neq M']$$

where probability is calculated over the random coins of A. Informally, a hash function is called collision resistant if, for any efficient algorithm A, the collision advantage of A for H is negligible. Unfortunately, we can not rule out the existence of an efficient collision finding algorithm A outputting (M, M') which is eventually a collision pair of the hash function H. But nobody may know or write down this algorithm based on our current knowledge. Therefore, we can say that a hash function is *collision resistant* if no efficient collision finding algorithm is known for it. Rogaway formalized this approach by introducing human ignorance model [13]. Keeping this in mind, we use the following definition of preserving properties of hash securities.

Definition 1. *A hash family* $H := \{H_{\text{IV}}\}_{\text{IV}\in\{0,1\}^t}$ *based on a compression function* f *is said to preserve* (ϵ, ϵ')*-collision security if given an efficient algorithm* A *with at least* ϵ *collision advantage for* H*, we can construct (write down its code modulo the subroutine* A*) an efficient algorithm* A' *with at least* ϵ' *collision advantage for* f*.*

Merkle-Damgard Hash function. Markle-Damgard or MD hash function has three basic components namely, (1) an underlying compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$ for some $b > 0$, (2) an initial value $\text{IV} \in \{0,1\}^t$ and (3) an easily computable padding rule $\mathsf{pad} : \mathcal{M} \to (\{0,1\}^b)^+$ for some message space $\mathcal{M}$. We define the classical iterated function $f_{\text{IV}}^+ : (\{0,1\}^b)^+ \to \{0,1\}^t$ as $f_{\text{IV}}^+(M_1, \cdots, M_\ell) = f(f(\cdots f(\text{IV}, M_1), \cdots), M_\ell)$, $M_1, \cdots, M_\ell \in \{0,1\}^b$. The MD hash function $\mathsf{MD}_{\text{IV},\mathsf{pad}}^f$ is defined as the composition of the following maps:

$$\mathcal{M} \xrightarrow{\mathsf{pad}} (\{0,1\}^b)^+ \xrightarrow{f_{\text{IV}}^+} \{0,1\}^t \quad \text{(mapping as a function)}$$

Thus, for all $M \in \mathcal{M}$., $\mathsf{MD}_{\text{IV},\mathsf{pad}}^f(M) = f_{\text{IV}}^+(\mathsf{pad}(M))$. An illustration is given in figure 1. Padding rule is essential to make the message size compatible with the domain of f_{IV}^+ as well as to keep the hash function collision preserving. In this paper, we will mainly study padding rules of MD and its different variants.

Padding Rules. The simplest possible (must be injective) padding rule is $\mathsf{pad}_0(M) = M\|10^d$ where d is the smallest nonnegative integer such that $|M| +$

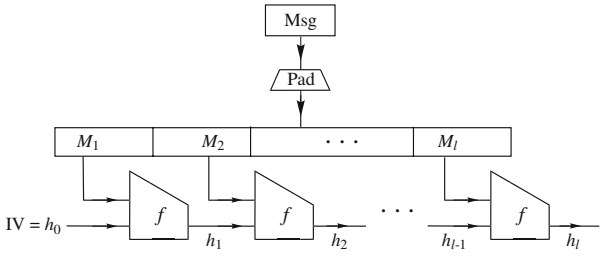


Fig. 1. The classical sequential iteration of a compression function

$1 + d$ is a multiple of b. However, pad_0 may not be sufficient to show the collision resistance property of MD under the only assumption that f is collision resistant. We show that if initial value is fixed then there is a compression function f which is collision resistant but we can actually construct a collision pair efficiently for $\mathsf{MD}^f_{\mathrm{IV},\mathsf{pad}_0}$. Moreover, the same result is true for any other "simply defined" padding rule which is not suffix-free (see Theorem 1 in section 3.3).

Definition 2. *Let $X, Y \in \{0,1\}^*$. We call X a suffix of Y if there exists a binary string Z such that $Y = Z\|X$. A padding rule* pad *is called* suffix-free *if, for any $M \neq M'$, $\mathsf{pad}(M)$ is not a suffix of $\mathsf{pad}(M')$.*

Here, we list some known padding rules. In crypto-1989, Damgard and Merkle independently proposed the classical MD iteration, but with different padding rules. Besides Merkle's and Damgard's padding rule, Sarkar defined a generalized version of Merkle's padding rule which has message space $\{0,1\}^*$ unlike Merkle's padding rule where message space is $\{0,1\}^{2^s-1}$ for some fixed s.

1. **Merkle's padding rule:** The message space $\mathcal{M} = \{0,1\}^{2^s-1}$ for some fixed s (we usually choose $s = 64$ or 128) and the padding rule is defined as $\mathsf{pad}_{\mathrm{merk}}(M) = M \parallel 10^d \parallel \mathrm{len}_s(M)$ where d is the smallest nonnegative integer such that $d + (|M| + 1 + s)$ is a multiple of b and $\mathrm{len}_s(M)$ represents the s bit binary representation of $|M|$. The classical MD hash function uses Merkle's padding rule. For example, SHA-2 hash family is nothing but MD hash function with Merkle's padding rule for $s = 64$. Note that Merkle's padding rule can hash messages of maximum possible size $2^s - 1$.

2. **Damgard's padding rule:** The message space for Damgard's padding rule is $\{0,1\}^*$. He used different padding rule $\mathsf{pad}_{\mathrm{damg}}$ which does not need to pad length of the message but it pads every message block by a single bit 0 or 1 depending on whether it is first block or not and finally, it pads one complete block representing the amount of 0-padding. More precisely, let $M\|0^d = x_1\|\cdots\|x_k$ where $|x_i| = b - 1$ and d is the smallest nonnegative integer such that $|M| + d$ is multiple of $b - 1$. Now,

$$\mathsf{pad}_{\mathrm{damg}}(M) = 0 \parallel x_1 \parallel 1 \parallel x_2 \parallel \cdots \parallel 1 \parallel x_k \parallel \mathrm{len}_b(d)$$

One disadvantage of using Damgard's padding is that for large messages it is not as efficient as Merkle's length padding as it needs more number of padded bits. However, unlike $\mathsf{pad}_{\mathrm{merk}}$, it can be applied to any arbitrary messages.

3. **Sarkar's padding rule:** Sarkar defined a new padding rule which can be applied to any tree based iteration which includes the classical sequential iteration. The padding rule is defined as

$$\mathsf{pad}_{\mathrm{sarkar}}(M) = 0 \parallel X_0 \parallel 0^{d_0} \parallel 1 \parallel X_1 \parallel 0^{d_1} \parallel \cdots \parallel 1 \parallel X_k \parallel 0^{d_k}$$

where $X_0 = M$, $X_i = \chi(X_{i-1})$, $1 \leq i \leq k$ and $\chi(x)$ denotes the smallest binary representation of $|x|$. Let k be the least positive integer such that $|X_k| \leq b$. The d_i's are smallest nonnegative integer so that $|X_i| + d_i$ is a multiple of $b - 1$. Note that, it can handle arbitrary messages and needs $\mathrm{O}(\log(|M|) \times \log^*(|M|))$ many padded bits where $\log^*$ is much slower function compare to log (see [15] for a precise definition).

It is straightforward to see that all these padding rules are suffix-free (we leave it for readers to verify). We provide a simple example of suffix-free padding rule $\mathsf{pad}_{\mathrm{length}}$ which needs $\mathrm{O}(\log(|M|))$ many bits. Basically, we apply Damgard's padding rule to the length of the message instead of applying it to the whole message. The precise definition of the padding rule can be found in section 3.2 and it is parameterized by a parameter s. Note that for any choice of s, the message space of this padding rule is $\{0,1\}^*$. This padding is so far the most efficient padding rule (in terms of the number of padding bits) and if the message size is less than 2^{63} then this padding rule with $s = 64$ is same as Merkle's padding rule. Therefore in practice, we can apply MD hash function as long as message size is less than 2^{63}. In table 1, we make a comparison for all these padding rules with respect to message space $\mathcal{M}$, length of the padded bits for a message M and how it preserves the collision security.

Table 1. A comparison table for different padding rules with an underlying compression function $f : \{0,1\}^{b+t} \to \{0,1\}^t$. pad_0 represents the 10^d padding rule defined in this section and $\mathsf{pad}_{\mathrm{length}}$ represents the suffix padding rule defined in the section 3.2. Padding length is given in terms of order. The last column represents when the hash function is collision secure.

padding rule	message space	padding length	assumption on f
$\mathsf{pad}_{\mathrm{merk}}$ [10]	$\{0,1\}^{\leq 2^s-1}$	b	collision-secure
$\mathsf{pad}_{\mathrm{damg}}$ [5]	$\{0,1\}^*$	$\lvert M\rvert$	collision-secure
$\mathsf{pad}_{\mathrm{sarkar}}$ [15]	$\{0,1\}^*$	$\log \lvert M\rvert \log^* \lvert M\rvert$	collision-secure
$\mathsf{pad}_{\mathrm{length}}$	$\{0,1\}^*$	$\log \lvert M\rvert$	collision-secure
pad_0	$\{0,1\}^*$	b	$(t-1)$-bit collision-secure

3 Characterization of Collision Preserving Padding Rules

In this section, we characterize all padding rules applied to MD hash function preserving the collision resistant property. We first show that suffix-free padding rule is sufficient to preserve collision resistant and then we provide some simple examples of suffix-free padding rules which are better than the known padding rules in terms of the padding size and the message space, in which padding rule can be applied. Finally, we show that suffix-free property is also necessary to preserve collision resistant property.

3.1 Sufficient Condition of Collision-Preserving Padding

For any $f : \{0,1\}^{b+t} \to \{0,1\}^t$ and $h \in \{0,1\}^t$ we have defined the iterated function $f_h^+ : (\{0,1\}^b)^+ \to \{0,1\}^t$. We can extend the definition to the domain $(\{0,1\}^b)^*$ by defining $f_h^+(\lambda) = h$ where λ is the empty bit string. It is easy to see that if $X_1 \in (\{0,1\}^b)^{k_1}$ and $X_2 \in (\{0,1\}^b)^{k_2}$ then

$$f_h^+(X_1 \| X_2) = f_{h'}^+(X_2) \text{ where } h' = f_h^+(X_1).$$

Now we provide basic intuitive lemma whose immediate corollary is that the suffix-free padding rule preserves collision resistant for MD hash function. The lemma says that if we have free-start collision for iterated hash f^+ (i.e., $f_h^+(X) = f_{h'}^+(X')$) with same length then there must be an intermediate collision during computations of $f_h^+(X)$ and $f_{h'}^+(X')$. A computation of $f_h^+(x_1, \cdots, x_k)$ means that the sequence of computations of $h_i = f(h_{i-1}, x_i)$, $1 \le i \le k$, where $h_0 = h$.

Lemma 1. (basic lemma)
Let $f : \{0,1\}^{b+t} \to \{0,1\}^t$ *and* $(h, x_1, \cdots, x_k) \neq (h', x'_1, \cdots, x'_k)$ *where* $h, h' \in \{0,1\}^t$ *and* $x_1, x'_1, \cdots, x_k, x'_k \in \{0,1\}^b$. *Then,*

$$f_h^+(x_1, \cdots, x_k) = f_{h'}^+(x'_1, \cdots, x'_k) \Rightarrow f(z, x_i) = f(z', x'_i),\ (z, x_i) \neq (z', x'_i)$$

where $1 \le i \le k$, $z = f_h^+(x_1, \cdots, x_{i-1})$ *and* $z' = f_{h'}^+(x'_1, \cdots, x'_{i-1})$.

Proof. Define $h_0 = h$, $h'_0 = h'$, $h_i = f_h^+(x_1, \cdots, x_i)$ and $h'_i = f_{h'}^+(x'_1, \cdots, x'_i)$, $1 \le i \le k$. Now we restate the statement of the lemma as follows.

> Given that $h_k = h'_k$ and $(h_0, x_1, \cdots, x_k) \neq (h'_0, x'_1, \cdots, x'_k)$ there must exist $0 \le i < k$ such that $(h_i, x_{i+1}) \neq (h'_i, x'_{i+1})$ but $h_{i+1} = h'_{i+1}$.

Thus, we have a collision $f(h_i, x_{i+1}) = h_{i+1} = h'_{i+1} = f(h'_i, x'_{i+1})$. To prove that there exists above such i, we use the contradiction method. So assume that, for all i, it is not true. Therefore, for all $1 \le i < k$, $h_{i+1} = h'_{i+1}$ implies that $(h_i, x_{i+1}) = (h'_i, x'_{i+1})$. Starting from $h_k = h'_k$ we have $(h_{k-1}, x_k) = (h'_{k-1}, x'_k)$. Since $h'_{k-1} = h_{k-1}$ we also have $(h_{k-2}, x_{k-1}) = (h'_{k-2}, x'_{k-1})$ and so on. Thus we must have $(h_0, x_1, \cdots, x_k) = (h'_0, x'_1, \cdots, x'_k)$ which is not true. Hence the claim is proved by contradiction. □

Recall that, $H^f_{\mathrm{IV},\mathsf{pad}}(M) = f^+_{\mathrm{IV}}(\mathsf{pad}(M))$ for a padding rule $\mathsf{pad} : \mathcal{M} \to (\{0,1\}^b)^+$. Here we fix an initial value IV and hence we are only interested in a single hash function and we write $H := \mathsf{MD}^f_{\mathsf{pad}}$. The computation of padding rule must be injective. Otherwise we will be able to find a collision of the hash function easily for any choice of underlying compression function. More precisely, if we have two messages $M \neq M'$ such that $\mathsf{pad}(M) = \mathsf{pad}(M')$ then clearly $H^f_{\mathrm{IV},\mathsf{pad}}(M) = H^f_{\mathrm{IV},\mathsf{pad}}(M')$ for any f. Since we usually choose a simply defined padding rule we can assume that we will be able to find efficiently a collision pair (M, M') of the padding rule if it is not injective. Now we want to classify all padding rules such that MD hash functions based on these padding rule preserves collision security for any choice of initial value. Here we show that this class is nothing but the set of all suffix-free padding rules.

Theorem 1. Sufficient Condition for collision-preserving padding *If* pad *is suffix-free padding rule then given any collision pair* (M, M') *for* $\mathsf{MD}^f_{\mathsf{pad}}$ *we can construct a collision pair of* f *efficiently. Thus,* $\mathsf{MD}^f_{\mathsf{pad}}$ *is preserving* (ϵ, ϵ)*-collision security for any* $\epsilon > 0$*.*

Proof. Let $\mathsf{pad}(M) = X$ and $\mathsf{pad}(M') = X'$. Without loss of generality we assume that $|X| \leq |X'|$. Let $X \in (\{0,1\}^b)^k$ and $X' = (Z, Y)$ where $Y \in (\{0,1\}^b)^k$. Define $h' = f^+_h(Z)$. As (M, M') is a collision pair, $f^+_h(X) = f^+_h(X')$ and hence $f^h(X) = f_{h'}(Y)$ where both X and Y are members of $(\{0,1\}^b)^k$ and $X \neq Y$ (since pad is suffix-free and hence X is not a suffix of X'). Thus, by using the above basic lemma 1 we must have a collision for f in the computation of $f^+_h(X)$ and $f^+_{h'}(Y)$. Since the computation of $f^+_h(X')$ includes the computation of $f^+_{h'}(Y)$, we are done. The collision advantage for f is at least the collision advantage of MD hash function. The efficiency of the collision finding for f is simple as we need to compute at most $(|M| + |M'|)/b$ computations of f where (M, M') is a collision pair generated from a collision finding algorithm for MD hash function. □

3.2 Simple Examples of Suffix-Free Padding Rules

We would like to note that known padding rules such as Damgard's padding rule, Merkle's padding rule, Sarkar's padding rule are all suffix-free.

Proposition 1. *The padding rules* $\mathsf{pad}_{\mathrm{merk}}$, $\mathsf{pad}_{\mathrm{damg}}$, *and* $\mathsf{pad}_{\mathrm{sarkar}}$ *are suffix-free. Hence the Merkle-Damgård hash function based on these padding rules preserves the collision security.*

It is straightforward to verify and so we leave it for readers to verify. Recall that Damgard and Sarkar's padding rules have domain $\{0,1\}^*$ whereas the Merkle's padding rule has domain $\{0,1\}^{2^d-1}$ for some fixed d. Damgard padding rule pads $\mathrm{O}(|M|)$ bits to the message M whereas Sarkar's padding rule needs roughly $\mathrm{O}(\log^* |M| \log |M|)$ many bits pad where $\log^*(|M|)$ is much slower function compared to $\log(|M|)$. Now we propose a much simpler padding rule which is suffix-free and which takes $\mathrm{O}(\log(|M|))$ many padded bits.

Let $\chi_s(|M|)$ denote the smallest multiple of s-bit binary representation of $|M|$. One can fix a suitable integer s, such as 8,32,64 etc. In other words, we first have a binary represent of $|M|$ and then add a sequence of 0's in the beginning of the length representation so that the size becomes multiple of s. Now we write $\chi_{s-1}(|M|) = p_1 \| \cdots \| p_{\ell-1} \| p_\ell$ where $|p_i| = s-1$ for $1 \le i \le \ell$. Let d be the smallest non-negative integer such that $(d + |M| + \ell s)$ is multiple of b. Now we define

$$\mathsf{pad}_{\mathrm{length}}(M) = M \parallel 0^d \parallel 0 \parallel p_1 \parallel 1 \parallel p_2 \parallel 1 \parallel \cdots \parallel p_\ell.$$

By definition of d the size of $\mathsf{pad}_{\mathrm{length}}$ becomes multiple of b. It is easy to see that this padding rule pads $d+\ell s$ many bits which is at most $b+\lceil \log(|M|) \rceil + \lceil \frac{\log(|M|)}{s-1} \rceil$. So the number of padding bits for this padding rule is $\mathrm{O}(\log |M|)$. Moreover, it can also be applied to any arbitrary message. Now we prove that it is a suffix-free padding rule. By using theorem 1, the MD hash function with this padding rule preserve collision security.

Lemma 2. *The padding rule* $\mathsf{pad}_{\mathrm{length}}$ *is suffix-free.*

Proof. Suppose $\mathsf{pad}_{\mathrm{length}}(M') = (X, \mathsf{pad}_{\mathrm{length}}(M))$ for some X, M and M'. Let us denote

$$\mathsf{pad}_{\mathrm{length}}(M) = M 0^d 0 p_1 1 p_2 \cdots 1 p_\ell, \quad \mathsf{pad}_{\mathrm{length}}(M) = M' 0^{d'} 0 p'_1 1 p'_2 \cdots 1 p'_{\ell'}.$$

Since $\mathsf{pad}_{\mathrm{length}}(M') = (X, \mathsf{pad}_{\mathrm{length}}(M))$ we must have $\ell' = \ell$ and $p'_i = p_i$, $1 \le i \le \ell$ by comparing the positions of 1 and 0 bits. So, $|M| = |M'|$ and hence $d = d'$. Thus, $|\mathsf{pad}_{\mathrm{length}}(M')| = |\mathsf{pad}_{\mathrm{length}}(M)|$. So, $X = \lambda$ and $\mathsf{pad}_{\mathrm{length}}(M') = \mathsf{pad}_{\mathrm{length}}(M)$ and hence $M = M'$. This proves that $\mathsf{pad}_{\mathrm{length}}$ is suffix-free. □

Remark 1. Note that for any messages of size up to 2^{63} the padding rule $\mathsf{pad}_{\mathrm{length}}$ with $s = 64$ is same as $\mathsf{pad}_{\mathrm{merk}}$. So we can use Merkle's padding rule and for any message longer than 2^{63} one can extend the definition by using $\mathsf{pad}_{\mathrm{length}}$ with $s = 64$. One may argue that the Merkle's padding rule is sufficient enough for all practical messages and the new padding rule only have of theoretical interest. However, in some applications (such as smart card) short messages appear more frequently (message sizes are usually less than 2^{15}). In those application, we can choose small value of s (such as 8 or 16). With this padding rule, we save at least 16 bit padding and as a result the hash function may be faster (since we can save sometimes one compression function invocation which is significant for short messages) than usual Merkle's padding rule. So, there are some applications where this padding rules are practically useful. Note that with $s = 16$, we can pad any message of arbitrary length. If we know that the message size can be large then $s = 32$ a reasonable choice.

Remark 2. Instead of padding length of the message, one can pad the number of message blocks. The padding rule capturing this notion is defined as follows:

$$\mathsf{pad}'_{\mathrm{length}}(M) = M \parallel 10^d \parallel 0 \parallel p_1 \parallel 1 \parallel p_2 \parallel 1 \parallel \cdots \parallel p_\ell$$

where $\chi_{s-1}(\lceil \frac{|M|}{b} \rceil) = p_1 \| \cdots \| p_{\ell-1} \| p_\ell$ and d is the smallest non-negative integer such that $(d + 1 + |M| + \ell s)$ is multiple of b. This variant actually pads the number of blocks of the message M instead of length of the message. In terms of order, both padding rule needs $\mathrm{O}(\log |M|)$ many bits. However, in most cases of message sizes, the later needs less number of padding bits.

3.3 A Necessary Condition for Collision-Preserving Padding Rule

We have shown that any suffix-free property is good for collision-preserving. Now we show that it is also a necessary condition. To show this let us fix a padding rule pad which is not suffix-free. Therefore, we also fix $M \neq M'$ such that $\mathsf{pad}(M)$ is a suffix of $\mathsf{pad}(M')$. We can do so since we assume that padding rule is simply defined function and hence it must be easy to find such a pair. Now we first construct a compression function f given a collision secure compression function f' such that (M, M') is a collision pair of H^f_{pad}. Then we prove that finding collision of f given M and M' and the oracle f is as hard as finding collision of f'. This proves that suffix-free padding rule is necessary to have a collision-preserving MD hash function for any compression function and for any initial value.

Theorem 2. Suffix-free is necessary condition
Let pad *be a fixed padding rule which is not suffix-free. Now, given a collision resistant compression function* f' *there is a collision resistant compression function* f*. Moreover, the Merkle-Damgård hash function based on* f *with the padding rule* pad *is not collision resistant (by providing a collision pair of it).*

Proof. Let us assume that there is a $(t-1)$-bit collision resistant compression function $f' : \{0,1\}^{b+t} \to \{0,1\}^{t-1}$, otherwise the question is moot. Without loss of generality, let the last bit of IV is 1. We have fixed a pair (M, M') such that $\mathsf{pad}(M)$ is a suffix of $\mathsf{pad}(M')$. Let $\mathsf{pad}(M') = (X, m) \| \mathsf{pad}(M)$, $m \in \{0,1\}^b$ and $X \in (\{0,1\}^b)^*$. For simplicity we first assume that $X = \lambda$. Define

$$f(x) = \begin{cases} f'(x) \parallel 0 & \text{if } x \neq \mathrm{IV} \| m \\ \mathrm{IV} & \text{if } x = \mathrm{IV} \| m \end{cases}$$

Thus, $f(\mathrm{IV}, m) = \mathrm{IV}$, a fixed point for f. Now it is easy to see that $H^f_{\mathsf{pad}}(M) = H^f_{\mathsf{pad}}(M')$. Now we have to show that f is collision resistant. Suppose there exists a collision finding algorithm A which finds collision for f. Let $x \neq x'$ such that $f(x) = f(x')$ where A returns the collision pair (x, x'). Now, it is easy to check that (x, x') is a collision pair of f' too. Since we do not know any collision algorithm for f', f must be collision resistant.

Now consider the case where $X \in (\{0,1\}^b)^+$. We first define $\widetilde{f}(x) = f'(x) \| 0$ and compute $\mathrm{IV}' = \widetilde{f}^+_{\mathrm{IV}}(X)$. Note that the last bit of IV' is 0 and hence it is different from IV. Since f' is a collision resistant compression function, IV' should be different from all other intermediate values in the computation of $\widetilde{f}^+_{\mathrm{IV}}(X)$. Otherwise, we will be able to find collision of f' easily. We define

$$f(x) = \begin{cases} f'(x) \parallel 0 & \text{if } x \neq \mathrm{IV}' \| m \\ \mathrm{IV} & \text{if } x = \mathrm{IV}' \| m \end{cases}$$

So, $f(x)$ and $\widetilde{f}(x)$ are same whenever $x \neq \mathrm{IV}' \| m$. Since IV' is different from all other intermediate values in the computation of $\widetilde{f}^{+}_{\mathrm{IV}}(X)$, we must have $f^{+}_{\mathrm{IV}}(X) = \widetilde{f}^{+}_{\mathrm{IV}}(X) = \mathrm{IV}$. Now we can show that (M, M') is a collision pair of H^{f}_{pad}. Note that, $H^{f}_{\mathsf{pad}}(M') = f^{+}_{\mathrm{IV}}(X, m, \mathsf{pad}(M)) = f^{+}_{\mathrm{IV}'}(m, \mathsf{pad}(M)) = f^{+}_{\mathrm{IV}}(\mathsf{pad}(M)) = H^{f}_{\mathsf{pad}}(M)$. Hence $H^{f}_{\mathsf{pad}}(M) = H^{f}_{\mathsf{pad}}(M')$. Now we show that the compression function f is also collision resistant. Suppose not, $x \neq x'$ and $f(x) = f(x')$ then clearly $f'(x) = f'(x')$ if $x, x' \neq \mathrm{IV}' \| m$. Moreover x or x' can not be $\mathrm{IV}' \| m$ otherwise the last bits of $f(x)$ and $f(x')$ will be different. Hence f is collision resistant as long as f' is collision resistant. The collision pair (M, M') also does not help to find a collision since M and M' are efficiently computable and we can use M and M' for any collision finding algorithm for f'. □

4 Length Padding is Redundant for a Variant of MD

In this section, we prove that the length padding is unnecessary if we use a small variant of Merkle-Damgård hash function and the underlying compression function is little more secure than collision resistant. We define the variant of the Merkle-Damgård hash function as follows.

Algorithm 1. A Variant of Merkle-Damgård Hash Function

Require: $f : \{0,1\}^t \times \{0,1\}^b \to \{0,1\}^t$, $M \in \{0,1\}^*$.
1: d is the remainder when we divide $t - |M| - 1$ by t.
2: partition $M10^d = M_1 \| \cdots \| M_\ell$, $M_1, \cdots, M_\ell \in \{0,1\}^b$
3: $h_0 = 0^t$
4: **for** $i = 1$ to ℓ **do**
5: $\quad h_i = f(h_{i-1}, M_i)$
6: $\quad$ **if** $h_i = 0^t$ **then**
7: $\quad\quad h_i = 0^{t-1}1$
8: $\quad$ **end if**
9: **end for**
10: **return** h_ℓ.

We simply apply MD hash function with pad_0 padding and with a checking in internal chaining value. If we ever come up with 0^t as a chaining value then we simply change the chaining value into $0^{t-1}1$. So we can think that it is MD hash function based on a compression function f' defined below.

$$f'(x) = \begin{cases} f(x) & \text{if } f(x) \neq 0^t \\ 0^{t-1}1 & \text{if } f(x) = 0^t \end{cases}$$

Note that $\mathbf{0} = 0^t$ is also the initial value of the Merkle-Damgård hash function. Suppose $\mathsf{MD}^{f'}_{\mathbf{0}}(M) = \mathsf{MD}^{f'}_{\mathbf{0}}(M)$ with $M \neq M'$. Then by using simple backward

induction on the padded messages, we can prove that either there is a collision of f' or there is a message $x \in \{0,1\}^{b+t}$ such that $f'(x) = \mathbf{0}$. By definition of f', there does not exist any such x. If (x, x') is a collision pair of f' then either it is also a collision for f or $f(x) = 0^t$ and $f(x') = 0^{t-1}1$. So either we have a collision of f or we have collision on the first $(t-1)$ bits of f with the collision value 0^{t-1}. Clearly, satisfying the second condition seems much harder than finding collision for any reasonable practical construction of a compression function. Thus, we have proved the following theorem.

Theorem 3. *Suppose it is hard to find collision of f or it is hard to find X_0 and X_1 such that $f(X_0) = 0^t$ and $f(X_1) = 0^{t-1}1$ then the hash function based on f defined in Algorithm 2. is collision resistant.*

Corollary 1. *Suppose it is hard to find collision on the first $(t-1)$-bits of a compression function f then the hash function based on f defined in Algorithm 2. is collision resistant.*

Remark 3. In the last section, we have proved that the suffix-free padding is necessary to preserve collision security for MD hash function. To do so, we provide a pathological example of the underlying compression function (easy to get the initial value 0^t even though it is collision secure). On the contrary, in this section, we show that any injective padding rule, not necessarily suffix-free, is sufficient to have collision security. Actually, we make sure that the pathological case does not appear by imposing the if condition (see step-6 of the Algorithm 2.1). This is the main reason, we do not contradict each other. Moreover, the variant actually do not preserve collision security according to the definition 1. But, it says that if we have little more security of the underlying compression function than collision security, then the hash function is collision secure. The additional security assumption seems to hold for any known practical secure hash function.

5 Three-Property Preserving Hash Function

Now we consider salted hash function which outputs the hash value with a salt $K \in \{0,1\}^s$ which is chosen randomly and keep it public (unlike message authentication code, hash function should be publicly computable). We denote the salted hash function as $H_K(\cdot)$ where K is the salt. We define three basic securities of a hash function for a salted hash function. These are collision, preimage and second-preimage and its corresponding advantages of adversaries is defined as

$$\begin{aligned}
\mathbf{Adv}_H^{\mathrm{coll}}(A_1) &= \mathsf{Pr}[A_1(K) \to (M, M') : H_K(M) = H_K(M'), M \neq M'] \\
\mathbf{Adv}_H^{\mathrm{2PI}}(A_3) &= \mathsf{Pr}[A_3(K, M) \to M' : H_K(M) = H_K(M'), M \neq M'] \\
\mathbf{Adv}_H^{\mathrm{PI}}(A_2) &= \mathsf{Pr}[A(K, z) \to M : H_K(M) = z]
\end{aligned}$$

where probabilities are computed over the internal randomness of the adversaries, uniform distribution of K, z and M chosen from $\{0,1\}^s$, $\{0,1\}^t$, and

Algorithm 2. A Variant of Merkle-Damgård Hash Function $\mathsf{BCM}^{f}_{\mathsf{pad}_0}$

Require: $f : \{0,1\}^t \times \{0,1\}^b \to \{0,1\}^t$ and $M \in \{0,1\}^*$.
1: d is the remainder when we divide $t - |M| - 1$ by t.
2: partition $M10^d = M_1 \| \cdots \| M_\ell$, $M_1, \cdots, M_\ell \in \{0,1\}^b$
3: $h_0 = 0^t$
4: **for** $i = 1$ to ℓ **do**
5: **if** $i = \ell$ **then**
6: $h_\ell = f(h_{\ell-1} \oplus K, M_\ell)$
7: **else**
8: $h_i = f(h_{i-1} \oplus M_{i+1}[t], M_i)$ \\ $X[t]$ represents the first t-bits of X.
9: **end if**
10: **if** $h_i = 0^t$ **then**
11: $h_i = 0^{t-1}1$
12: **end if**
13: **end for**
14: **return** h_ℓ.

$\{0,1\}^\ell$ for some ℓ, respectively. Now we define a variant of $\mathcal{BCM}$ [1] (backward chaining mode) which is simpler and needs less salt size. Recall that $\mathcal{BCM}$ uses the padding rule $\mathsf{pad}_{\mathrm{merk}}$ and its salt size is $b + 2t$. In this paper, we use salt of size t only. We denote the hash function as $\mathsf{BCM}^{f}_{\mathsf{pad}_0}$ since it uses the simplest injective padding rule pad_0. One may use some other padding rules.

A hash family $H := \{H_K\}_{K \in \{0,1\}^s}$ based on a compression function f is said to almost preserve (ϵ, ϵ')-collision security if given an efficient algorithm A with at least ϵ collision advantage for H_K, we can construct (write down its code modulo the subroutine A) an efficient algorithm A' with at least ϵ' collision advantage for f' (the first $t-1$ bits of f). Similarly, we define for the other two security notions (2nd) preimage. The proof of the following theorem is given in the full version of the paper and it hash similarity with proof given in [1].

Theorem 4. *$\mathsf{BCM}_{\mathsf{pad}_0}$ almost preserve (ϵ, ϵ)-collision, (2nd) preimage security.*

The preimage preserving is trivial since inverting $\mathsf{BCM}_{\mathsf{pad}_0}$ reduces to inverting the compression function by looking at the last invocation of the compression function. Note that preimage attacker for hash function and preimage attacker for compression function receives a target image which is uniformly distributed. The proof for second preimage is very much similar to the security proof for three property preserving hash function in [1] combined with the collision preserving security analysis for MD hash function with padding rule pad_0. We provide more details to the full version of the paper.

6 Conclusion

We study padding rules in different aspects which is very essential for designing hash function. Appending 1 followed by a suitable size sequence of 0 and binary

representation of length is a very standard way to have padding for hash function. But, this padding rule can only be applied for messages of size at most some specified large value. Both for theoretical and practical point of views, we can look for padding rules which can handle arbitrary messages. Padding rules by Damgard and Sarkar are some known examples for this. Here we have shown that suffix-free padding rule is sufficient to preserve collision resistant and as a result we construct a simple padding rule which is more efficient than padding rules both by Damgard and Sarkar. We have also proved that suffix is a necessary condition preserving collision security. Thus, it would be interesting to see that whether padding rule is optimum or not. So we propose the following open problem.

> **Open Problem:** Try to find suffix-free padding rule which needs less than logarithm number of bits. On the other hand, we can try to prove that asymptotically any suffix-free padding rule needs at least logarithm number of bits.

We believe that the second option is more feasible. We also have shown that the simplest padding such as padding 10^d only can be sufficient for collision preserving property if we restrict collision resistant assumption of the underlying compression function for the first $(t-1)$ bits. Thus, the simplest Merkle-Damgård hash function becomes collision resistant which does not have any overhead costs due to length of the message. We also study a simple variant of MD hash function which preserves collision resistance, second preimage as well as preimage resistance. Thus we believe that padding length is not needed if we choose initial value properly.

References

1. Andreeva, E., Preneel, B.: A Three-Property-Preserving Hash Function. To appear in: Selected Areas in Cryptography (2008)
2. Bellare, M., Rogaway, P.: Collision-Resistant Hashing: Towards Making UOWHFs Practical. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 470–484. Springer, Heidelberg (1997)
3. Bellare, M., Rogaway, P.: Introduction to Modern Cryptography, http://www-cse.ucsd.edu/~mihir/cse207/classnotes.html
4. Shoup, V.: Using Hash Functions as a Hedge against Chosen Ciphertext Attack. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 275–288. Springer, Heidelberg (2000)
5. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
6. Damgård, I.B.: Collision Free Hash Functions and Public Key Signature Schemes. In: Price, W.L., Chaum, D. (eds.) EUROCRYPT 1987. LNCS, vol. 304, pp. 203–216. Springer, Heidelberg (1988)
7. Gibson, J.K.: Discrete logarithm hash function that is collision free and one-way. IEE Proceedings-E 138, 407–410 (1991)
8. Don., B.J.: Improving Hash Function Padding. NIST hash workshop (2005), http://csrc.nist.gov/groups/ST/hash/documents/Johnson_Padding.pdf

9. Kelsey, J., Schneier, B.: Second Preimages on n-bit Hash Functions for Much Less than 2^n Work. Cryptology ePrint Archive (2004), http://eprint.iacr.org/2004/304
10. Merkle, R.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
11. Naor, M., Yung, M.: Universal one-way hash functions and their cryptographic applications. In: Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing, pp. 33–43. ACM Press, New York (1989)
12. NIST/NSA. FIPS 180-2 Secure Hash Standard (August 2002), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
13. Rogaway, P.: Formalizing Human Ignorance: Collision-Resistant Hashing without the Keys. Eprint archive (2006), http://eprint.iacr.org/2006/281.pdf
14. Rogaway, P., Shrimpton, T.: Cryptographic Hash Function Basics: Definitions, Implications, and Separations for Pre-image Resistance, Second Pre-image Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
15. Sarkar, P.: Domain Extender for Collision Resistant Hash Functions: Improving Upon Merkle-Damgard Iteration. Discrete Applied Mathematics 157(5), 1086–1097 (2009)
16. Stinson, D.R.: Cryptography: Theory and Practice, 2nd edn. CRC Press, Inc., Boca Raton
17. Stinson, D.R.: Some observations on the theory of cryptographic hash functions. ePrint Archive Report (2001), http://eprint.iacr.org/2001/020/

Distinguishing Attack on the Secret-Prefix MAC Based on the 39-Step SHA-256

Hongbo Yu[1] and Xiaoyun Wang[2,⋆]

[1] Center for Advanced Study, Tsinghua University, Beijing 100084, China
yuhongbo@mail.tsinghua.edu.cn
[2] Tsinghua University and Shandong University, China
xiaoyunwang@tsinghua.edu.cn, xywang@sdu.edu.cn

Abstract. In this paper, we present the first distinguishing attack on the LPMAC based on step-reduced SHA-256. The LPMAC is the abbreviation of the secret-prefix MAC with the length prepended to the message before hashing and it's a more secure version of the secret-prefix MAC. In [19], Wang *et* al. give the first distinguishing attack on HMAC/NMAC-MD5 without the related key, then they improve the techniques to give a distinguishing attack on the LPMAC based on 61-step SHA-1 in [23]. In this paper, we utilize the techniques in [23] combined with our differential path on step-reduced SHA-256 to distinguishing the LPMAC based on 39-step SHA-256 from the LPMAC with a random function. The complexity of our attack is about $2^{184.5}$ MAC queries.

Keywords: SHA-256, distinguishing attack, MAC.

1 Introduction

Hash function play an important role in modern cryptography. One main application of their use is for the message authentication. A message authentication code (MAC) is a function which takes a message and a secret key as inputs and produces an output called an *authentication tag*. Three earlier MACs based on hash functions are constructed by the *secret prefix method*, *secret suffix method* and *envelope method*. The two other popular MACs based on the existing hash functions are MDx-MAC [13] and HMAC/NMAC [2].

Recent years, cryptanalysis on hash function MDx-family and SHA-family has been extensively studied. The attacks on hash functions [5,20,21,22,24,25,27,3] have shown that the prevailing hash functions such as MD4, HAVAL, MD5, SHA-0, SHA-1 are not collision resistant. Therefore research on the MACs based on

⋆ Supported by 973 Project(No.2007CB807902), 863 Project(No.2006AA01Z420) and the National Natural Science Foundation of China(NSFC Grant No.60803125).

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 185–201, 2009.

those hash functions has become a hot topic. There are three kinds of attacks on MACs: *distinguishing attack*, *forgery attack* and *key-recovery attack*. In [9], Kim *et al.* defined two kinds of distinguishing attack on the hash-based MACs, named *distinguishing-R* and *distinguishing-H* attacks. The distinguishing-R attacks means distinguishing a MAC with a random function, and distinguishing-H attack checks an instantiated MAC (in which is embedded a specific hash function) from a MAC with a random function. The distinguishing-R attack is useful when the cryptanalyst wants to check whether output strings are produced from a specific MAC algorithm or a random function, while the distinguishing-H attack is to check which cryptographic hash is embedded in a specific MAC algorithm (in this case, the cryptanalyst somehow already knew that the output producing MAC algorithm, for instance, using the distinguishing-R attack, but does not know the underlying hash function). For a MAC based on an iterated compression function, the distinguishing-R attack requires about $2^{n/2}$ messages queries by the birthday paradox [13], where n is the length of the MAC output. But for the distinguishing-H attack, it has no constraint of $2^{n/2}$ message queries because there does not exist a general attack based on the birthday paradox which distinguishing a MAC with existing hash functions from a MAC with a random function. So the idea complexity for the distinguishing-H attack should be the exhaustive attack. In this paper, we only focus on the distinguishing-H attack. For simplicity, we call it as distinguishing attack.

Most of the attacks on hash-based MACs make use of collision or near-collision differential path for the underlying compression function with probability higher than 2^{-n}. For MD4, HAVAL and SHA-0, it's easy to find the differential paths with high probability [20,26,22,3], so the distinguishing, forgery and key-recovery (or partial key recovery) attacks can be implemented successfully [4,6,18]. For MD5, the only differential path with high probability is the pseudo-collision differential found by den Boer and Bosselaers [1], and we call it as a dBB difference which consists of different IVs and the same massage. So the previous attacks on MD5 are all in related key setting [4,6,18,16]. For the MACs based on the SHA-1, distinguishing, forgery and partial or full key recovery attacks on NMAC/HMAC-SHA-1 with a reduced number of steps (up to 62 out of 80) are given in [16]. For the SHA-2, the first studies on the unmodified SHA-2 is given by Mendel *et al.* [10], which give a 18-step collision and a 22-step pseudo-near-collision for SHA-256. A series of analysis results on the step-reduced SHA-2 can be obtained in [14,11,8,15] subsequently.

Recently, Wang *et al.* give the first distinguishing attack on HMAC and NMAC based on MD5 without related key in Eurocrypt 2009 [19]. They use a distinguisher with a pair of two-block messages, the first block is used to guarantee the appearance of the dBB difference by the birthday attack and the second block uses a dBB difference to make a collision. The aims of their distinguishing is to detect a dBB-collision from other types of collisions. Furthermore in FSE 2009 [23], Wang *et al.* improve their techniques to distinguish the LPMAC based on 61-step SHA-1. The distinguisher also consists of a pair of two-block messages, the first 960 bits of the messages are used to ensure the emergence of a specific

difference in the 14-th step of the second block, and the last 64 bits are used to ensure the establishment of the 34-step differential path. Because the outputs of the first block are unknown, it's more complex to detect the specific difference than that of MD5 and it requires that the probability of last 34-step differential path is higher than $2^{-n/4}$ where n is the length of the MAC output.

In this paper, we apply the distinguishing techniques of Wang's [23] to the LPMAC based on step-reduced SHA-256. We find a 25-step differential path of SHA-256 with probability 2^{-41}. Based on it, we can distinguish a LPMAC based on 39-step SHA-256 with a LPMAC from a random function with complexity $2^{184.5}$. The difficulty of our attack is to deal with a large number of linear message equations because of the non-linear message expansion of SHA-256.

This paper is organized as follows. In section 2, we define some notations and give a brief description of SHA-256 and LPMAC. In section 3, we give an overview of Wang *et al.*'s distinguishing attack on SHA-1. In section 4, we describe our distinguishing attack on the LPMAC based on 39-step SHA-2. Finally we conclude the paper in section 5.

2 Backgrounds and Definitions

In this section, we define some notations used in this paper, and give a brief description of the hash function SHA-256 and the secret-prefix MAC.

2.1 Notations

$x\|y$: concatenation of the two bitstrings x and y
$x_{i,j}$: the j-th bit of x_i, where x_i is a 32-bit word and $x_{i,32}$ is the most significant bit
$+,-$: addition and subtration modular 2^{32}
$\wedge,\vee,\oplus$: bitwise AND, OR, and exclusive OR
Δx : the XOR difference $x \oplus x'$
$\Delta^{-} x$: the integer modular subtration difference $x - x'$, where x and x are two 32-bit words
S^n : right-rotation by n-bit
R^n : right-shift by n-bit

2.2 Description of SHA-256

The SHA-2 family was standardized by NIST in 2002 [12]. It consists of the hash functions SHA-256, SHA-224, SHA-512 and SHA-384. For our purpose attack, here we only describe SHA-256.

SHA-256 is an iterated cryptographic hash function based on compression function which takes a message of length less than 2^{64} and produces a 256-bit hash value. The compression function takes a 256-bit chaining value $H_i = (a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0)$ and a 512-bit message block M^i as inputs and outputs another 256-bit chaining value H_{i+1}. The final chaining value is the hash value by iterating over all the message blocks.

Each 512-bit block of the padded message M^i is divided into sixteen 32-bit words which are denoted as $m_0, m_1, ..., m_{15}$. The message words are expanded to sixty-four 32-bit words w_0, w_1,..., w_{63}:

$$w_j = \begin{cases} m_j, & 0 \le j \le 15; \\ \sigma_1(w_{j-2}) + w_{j-7} + \sigma_0(w_{j-15}) + w_{j-16}, & 16 \le j \le 63. \end{cases}$$

The compression function of SHA-256 consists of 4 rounds and each involves 16 steps, which is defined as follows:

- Input: sixty-four 32-bit words w_0, w_1,...,w_{63} and a 160-bit chaining value $H_i = (a_0, b_0, c_0, d_0, e_0, f_0, g_0, h_0)$.
- Step update: for $j = 1$ to 64,

$$\begin{aligned} T_1 &= h_{j-1} + \Sigma_1(e_{j-1}) + Ch(e_{j-1}, f_{j-1}, g_{j-1}) + k_{j-1} + w_{j-1} \\ T_2 &= \Sigma_0(a_{j-1}) + Maj(a_{j-1}, b_{j-1}, c_{j-1}) \\ a_j &= T_1 + T_2 \\ b_j &= a_{j-1} \\ c_j &= b_{j-1} \\ d_j &= c_{j-1} \\ e_j &= d_{j-1} + T_1 \\ f_j &= e_{j-1} \\ g_j &= f_{j-1} \\ h_j &= g_{j-1} \end{aligned}$$

- Output: $H_{i+1} = (a_0 + a_{64}, b_0 + b_{64}, c_0 + c_{64}, d_0 + d_{64}, e_0 + e_{64}, f_0 + f_{64}, g_0 + g_{64}, h_0 + h_{64})$.

The Boolean functions $Ch(x, y, z)$ and $Maj(x, y, z)$ employed in the SHA-256 are defined as follows:

$$\begin{aligned} Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z) \\ Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \end{aligned}$$

The functions $\Sigma_0(x)$, $\Sigma_1(x)$, $\sigma_0(x)$ and $\sigma_1(x)$ are defined as follows:

$$\begin{aligned} \Sigma_0(x) &= S^2(x) \oplus S^{13}(x) \oplus S^{22}(x) \\ \Sigma_1(x) &= S^6(x) \oplus S^{11}(x) \oplus S^{25}(x) \\ \sigma_0(x) &= S^7(x) \oplus S^{18}(x) \oplus R^3(x) \\ \sigma_1(x) &= S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x) \end{aligned}$$

The constants k_j in each step can be referred to [12].

2.3 Secret-Prefix MAC and LPMAC

The secret prefix method consists of prepending a secret key k to the message M before the hashing operation: $MAC(M) = h(k\|M)$ for h an unkeyed hash function. If the key consists of (maybe padded) a complete block, this corresponds to a hash function with a secret IV. This method was suggested for MD4 independently by Tsudik [17] and by the Internet Security and Privacy Working Group for use in the Simple Network Management Protocol (SNMP) [7]. But this kind of MAC is insecure: a single text-MAC pair contains information essentially equivalent to the secret key, independent of its size. An attacker may append any blocks to the message and update the MAC accordingly, using the old MAC as the initial chaining variable. An countermeasure for this type of attack was proposed by prepending the length of the unpadded message before hashing [17]. We denoted this type of MAC as $LPMAC_k(M) = h(k\|length\|\overline{M})$, where $\overline{M}$ is the padded message of M. Suppose that $k\|length$ consists of a complete block. The padding method we selected for the LPMAC in this paper is: if the length of M is the multiple of 512 bits, it no need to pad; Otherwise, appending minimum number of zeros to make a whole number of blocks.

3 Overview of Wang *et al.*'s Distinguishing Attack on the LPMAC Based on 61-Step SHA-1 [23]

The distinguishing attack for the LPMAC based on step-reduced SHA-1 in [23] utilized a 47-step differential path P of SHA-1 from step 15 to 61 with probability 2^{-34} (See Table 2 of [23]). The basic idea of Wang *et.al*'s technique in [23] is to detect a specific difference which consists of a two-block message pair $(x\|y\|z, x'\|y'\|z')$, where x and x' are one-block messages, y and y' are 448-bit (14 words) messages of the second block, z and z' are the remaining 64-bit messages, $\Delta y = y' \oplus y$ and $\Delta z = z' \oplus z$ are the fixed differences which are determined by the differential path P.

Let cv and cv' denote the output values after the first block of $LPMAC_k$ $(x\|y\|z)$ and $LPMAC_k(x'\|y'\|z')$, and ch_i and ch'_i denote the outputs after step i of the second block respectively. Then the specific difference satisfies the following conditions:

1. The XOR difference $\Delta ch_{14} = ch_{14} \oplus ch'_{14}$ is a fixed difference determined in advance and ch_{14} satisfies 9 fixed conditions.
2. The outputs of the message pair $(x\|y\|z, x'\|y'\|z')$ satisfies

$$LPMAC_k(x'\|y'\|z') - LPMAC_k(x\|y\|z) = \delta_1 + \delta_2 \tag{1}$$

where $\delta_1 = cv' - cv$ and $\delta_2 = ch'_{61} - ch_{61}$.

For the random message pair $(x\|y\|z, x'\|y'\|z')$, the condition 1 holds with probability 2^{-169}. Under the condition 1, if the LPMAC is based on 61-step SHA-1, the condition 2 holds with probability 2^{-34}. Otherwise, if the LPMAC is based

on a random function, the condition 2 is satisfied with the average probability 2^{-160}.

Because the output difference δ_1 of the first block is unknown, the specific difference can not be detected similar as the distinguishing attack on HMAC/NMAC-MD5 in [19]. So it needs another two 64-bit messages z_1 and $z_1' = z_1 + \Delta z$ that also satisfy

$$LPMAC_k(x'\|y'\|z_1') - LPMAC_k(x\|y\|z_1) = \delta_1 + \delta_2 \tag{2}$$

Combine the equations (1) and (2), we can get

$$LPMAC_k(x'\|y'\|z_1') - LPMAC_k(x'\|y'\|z')$$

$$= LPMAC_k(x\|y\|z_1) - LPMAC_k(x\|y\|z) \tag{3}$$

So if the LPMAC is based on 61-step SHA-1, the message quadruple $(x\|y\|z$, $x\|y\|z_1$, $x'\|y'\|z'$, $x'\|y'\|z_1')$ satisfies the equation (3) with probability 2^{-68} under the condition 1 ; Otherwise, the equation (3) holds with probability 2^{-160}. According to this probability advantage, the main idea of the distinguishing attack in [23] is summarized as follows:

Choose about 2^{68} sets S_i randomly and each consists of $2^{84.5}$ different one-block messages. For each message $x \in S_i$ $(0 \le i \le 2^{68})$, query the MACs with $x\|y\|z$, $x\|y\|z_1$, $x\|y'\|z'$ and $x\|y'\|z_1'$ where $y\|z$, $y\|z_1$, $y'\|z'$ and $y'\|z_1'$ are four fixed messages and compute the following two structures of differences

$$S_{i,1} = \{LPMAC_k(x\|y\|z_1) - LPMAC_k(x\|y\|z) | x \in S_i\},$$
$$S_{i,2} = \{LPMAC_k(x\|y'\|z_1') - LPMAC_k(x\|y'\|z') | x \in S_i\}.$$

Find all the collisions (x, x') in $S_{i,1}$ and $S_{i,2}$ $(0 \le i \le 2^{68})$ so that the equation (3) holds. The total number of MAC queries are about $2^{154.5}$. It expects about 2^{68} collisions where each set contains one collision on average. For each collision pair (x, x'), do:

- Compute $\delta = LPMAC_k(x'\|y'\|z') - LPMAC_k(x\|y\|z)$.
- Replace the fixed message pair (z, z') with 2^{34} different 64-bit message pairs $(\overline{z}, \overline{z}')$ where each $y\|\overline{z}$ also satisfies the fix message conditions and $\overline{z}' = \overline{z} + \Delta z$. Query all the 2^{34} message pairs $(x\|y\|\overline{z}, x'\|y'\|\overline{z}')$ and get their MACs. For each message pair, compute $\overline{\delta} = LPMAC_k(x'\|y'\|\overline{z}') - LPMAC_k(x\|y\|\overline{z})$ and compare $\overline{\delta}$ to δ.

Once a $\overline{\delta}$ is found so that $\overline{\delta} = \delta$, then conclude that the target MAC is a LPMAC based on 61-step SHA-1; Otherwise, conclude that the MAC is LPMAC based on a random function. The complexity of the distinguishing attack is about $2^{154.5}$.

4 Distinguishing Attack on the LPMAC Based on 39-Step SHA-256

In this section, we apply the distinguisher in [23] to distinguish the LPMAC based on 39-step SHA-256. The key step for our attack is to find a specific differential path P_1 of 39-step SHA-256 which is shown in Table 3. The differential path in Table 3 can be divided into two parts: steps $1 \sim 14$ and steps $15 \sim 39$. For the first part, we neglect the exact output differences in the first 13 steps and only focus on the output difference after step 14 which serves as the input of the the second part. The differential path of second part from steps 15 to 39 is a pseudo-near-collision differential path with probability 2^{-41}. The *pseudo-near-collision* for the compression function can be defined as follows:

Definition. We say that (h, x) and (h', x') is a *pseudo-near-collision* for the compression function f: $\{0,1\}^n \times \{0,1\}^m \mapsto \{0,1\}^n$, if the Hamming distance between $f(h, x)$ and $f(h', x')$ is small (typically, a few bits) and $(h, x) \neq (h', x')$.

The distinguisher for the LPMAC based on 39-step SHA-1 also consists of a two block messages pair $(x\|y\|z, x'\|y'\|z')$, where x and x' are one-block messages, y and y' are 448-bit truncated messages of the second block, z and z' are the remaining 64-bit messages, and the difference $\Delta y = y \oplus y'$ and $\Delta z = z \oplus z'$ are determined by the path P_1. The purpose of our attack is to distinguish the output difference occurred after step 14 of the second block.

4.1 The Specific Differential Path for 39-Step SHA-256

The message difference $\Delta M = M \oplus M' = (\Delta m_0, ..., \Delta m_{15})$ for the 39-step SHA-256 is selected are as follows (See Table 1):

Table 1. The message difference

$\Delta m_0 \sim \Delta m_3$	0x92844891 0x50824014 0x84521222 0x4a901104
$\Delta m_4 \sim \Delta m_7$	0x55410245 0xa1114484 0x90904111 0x04444441
$\Delta m_8 \sim \Delta m_{11}$	0x12022882 0x00104011 0x00880080 0x81544280
$\Delta m_{12} \sim \Delta m_{15}$	0x0a020000 0x11002000 0x80000000 0

Let $W = (w_0, w_1, ..., w_{38})$ and $W' = (w'_0, w'_1, ..., w'_{38})$ be the expanded message of M and M' respectively. To make $\Delta w_i = w_i \oplus w'_i$ $(16 \leq i \leq 38)$ meet the fixed message differences in Table 3, i.e.,

$$\begin{aligned} &\Delta w_i = 0, 16 \leq i \leq 37, i \neq 23, \\ &\Delta w_{23} = 0x80000000, \\ &\Delta^- w_{38} = 2^{13} + 2^{24} + 2^{28}, \end{aligned}$$

we need to deduce a group of sufficient conditions on message W. For example, according to the message expansion algorithm of SHA-256,

$$w_{16} = w_0 + \sigma_0(w_1) + w_9 + \sigma_1(w_{14}),$$

$$w'_{16} = w'_0 + \sigma_0(w'_1) + w'_9 + \sigma_1(w'_{14}).$$

Let $\Delta w_{16} = 0$, then

$$w_0 + \sigma_0(w_1) + w_9 + \sigma_1(w_{14}) = w'_0 + \sigma_0(w'_1) + w'_9 + \sigma_1(w'_{14}) \tag{4}$$

From the XOR difference

$$\begin{aligned}
&\Delta\sigma_0(w_1) = \sigma_0(\Delta w_1) = \sigma_0(0x50824014) = 0xb2b458a2,\\
&\Delta\sigma_1(w_{14}) = \sigma_1(\Delta w_{14}) = \sigma_1(0x8000000) = 0x00205000,\\
&\Delta w_0 = 0x92844891,\\
&\Delta w_9 = 0x00104011,
\end{aligned}$$

we can deduce a set of sufficient conditions to guarantee the equation (4) hold:

$$\begin{cases}
w_{9,1} = w_{0,1},\\
\sigma_0(w_1)_2 = w_{0,1} \oplus 1,\\
w_{9,5} = w_{0,5},\\
\sigma_0(w_1)_6 = w_{0,5} \oplus 1,\\
\sigma_0(w_1)_8 = w_{0,8} \oplus 1,\\
\sigma_0(w_1)_{12} = w_{0,12} \oplus 1,\\
\sigma_0(w_1)_{13} = \sigma_1(w_{14})_{13} \oplus 1,\\
\sigma_0(w_1)_{15} = \sigma_1(w_{14})_{15} \oplus 1,\\
w_{9,15} = w_{0,15} \oplus 1,\\
\sigma_0(w_1)_{19} = w_{0,19} \oplus 1,\\
\sigma_0(w_1)_{21} = w_{0,21} \oplus 1,\\
\sigma_0(w_1)_{22} = \sigma_1(w_{14})_{22} \oplus 1,\\
\sigma_0(w_1)_{24} = w_{0,24} \oplus 1,\\
\sigma_0(w_1)_{26} = w_{0,26} \oplus 1,\\
\sigma_0(w_1)_{29} = w_{0,29},\\
\sigma_0(w_1)_{30} = w_{0,29} \oplus 1.
\end{cases} \tag{5}$$

Because $\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$ and $\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$, then the j-the bit of $\sigma_0(x)$ and $\sigma_1(x)$ can be expressed as

$$\sigma_0(x)_j = \begin{cases} x_{(j+7)mod32} \oplus x_{(j+18)mod32} \oplus x_{(j+3)mod32}, & 1 \le j \le 29,\\ x_{(j+7)mod32} \oplus x_{(j+18)mod32}, & 30 \le j \le 32. \end{cases}$$

$$\sigma_1(x)_j = \begin{cases} x_{(j+17)mod32} \oplus x_{(j+19)mod32} \oplus x_{(j+10)mod32}, & 1 \le j \le 22,\\ x_{(j+17)mod32} \oplus x_{(j+19)mod32}, & 23 \le j \le 32. \end{cases}$$

Take $\sigma_0(x)_j$ and $\sigma_1(x)_j$ into the equation system (5) and arrange the items in each equation from the higher bit to lower bit, we can get a group of message equations:

$$\begin{cases} w_{1,16} = w_{1,5} \oplus w_{0,29} \oplus 1, \\ w_{1,20} = w_{1,9} \oplus w_{1,5} \oplus w_{0,1} \oplus 1, \\ w_{1,22} = w_{1,15} \oplus w_{1,11} \oplus w_{1,5} \oplus w_{0,8} \oplus w_{0,19}, \\ w_{1,24} = w_{1,13} \oplus w_{1,9} \oplus w_{0,5} \oplus 1, \\ w_{1,26} = w_{1,15} \oplus \oplus w_{1,11} \oplus w_{0,8} \oplus 1, \\ w_{1,29} = w_{1,12} \oplus w_{1,1} \oplus w_{0,26} \oplus 1, \\ w_{1,30} = w_{1,19} \oplus w_{1,15} \oplus w_{0,12} \oplus 1, \\ w_{1,31} = w_{1,27} \oplus w_{1,10} \oplus w_{0,24} \oplus 1, \\ w_{1,32} = w_{1,15} \oplus w_{1,4} \oplus w_{0,29}, \\ w_{9,1} = w_{0,1}, \\ w_{9,5} = w_{0,5}, \\ w_{9,15} = w_{0,15} \oplus 1, \\ w_{9,21} = w_{1,28} \oplus w_{1,24} \oplus w_{1,7} \oplus 1, \\ w_{14,25} = w_{14,7} \oplus w_{14,9} \oplus w_{14,2} \oplus w_{1,29} \oplus w_{1,25} \oplus w_{1,22} \oplus w_{1,8} \oplus w_{1,1} \oplus w_{1,18}, \\ w_{14,30} = w_{14,23} \oplus w_{14,9} \oplus w_{14,7} \oplus w_{1,31} \oplus w_{1,29} \oplus w_{1,25} \oplus w_{1,20} \oplus w_{1,16} \oplus w_{1,8}, \\ w_{14,32} = w_{14,7} \oplus w_{14,9} \oplus w_{1,29} \oplus w_{1,25} \oplus w_{1,8} \oplus 1, \end{cases} \tag{6}$$

For each message word difference of $\Delta w_{16} \sim \Delta w_{38}$ in Table 3, we can deduce a group of sufficient conditions similar to the equation system (6). There are total 150 conditions on $W = (w_0, ..., w_{38})$ which are shown in Table 5 and 5.

For the pseudo-near-collision differential path from steps 15 to 39 of Table 3, the input chaining variable difference $\Delta ch_{14} = ch'_{14} \oplus ch_{14}$ is selected as (in hexadecimal expression)

$$(0, 22140240, 80000000, 0, 00082200, 20040000, 80000000, 4411008c),$$

and the output difference $\Delta^- ch_{39} = ch'_{39} - ch_{39}$ is

$$(2^{13} + 2^{24} + 2^{28}, 0, 0, 0, 2^{13} + 2^{24} + 2^{28}, 0, 0, 0).$$

It's easy to deduce the sufficient conditions on the chaining variables ch_{14} to ch_{39} according to the properties of the Boolean functions Ch and Maj. There are 25 conditions in ch_{14} and 41 conditions in $ch_{15} \sim ch_{39}$ altogether (See Table 4), and each condition can be fulfilled with probability 1/2. So if the 25 sufficient conditions in ch_{14} and 150 conditions in message words $w_0 \sim w_{38}$ are fulfilled, the differential path from steps 15 to 39 in Table 3 holds with probability 2^{-41}.

It deserves to note that we use the XOR difference in the first 38 steps in Table 3, and use the integer modular subtration difference in the last step (step 39).

4.2 The Distinguishing Algorithm on LPMAC from 39-Step SHA-256

In this section, we give the distinguishing attack on the LPMAC based on 39-step SHA-2 utilizing the technique in [23] combined with our new differential path.

The first step of our attack is to select four fixed one-block messages $y\|z$, $y\|z_1$, $y'\|z'$, and $y'\|z'_1$ which satisfy

- The messages $y\|z$ and $y\|z_1$ satisfies the 150 message conditions in Table 5 and 5.
- $\Delta y = y' \oplus y$ and $\Delta z = z' \oplus z = z_1 \oplus z'_1$, $\Delta y \| \Delta z$ are the target message differences in Table 1.

The purpose of our attack is to find a 512-bit message pair (x, x') so that the message quadruple $(x\|y\|z, x\|y\|z_1, x'\|y'\|z', x'\|y'\|z'_1)$ satisfy the following two conditions:

1. Let ch_{14} and ch'_{14} denote the 14-th step outputs of the second block of $LPMAC_k(x\|y\|z)$ and $LPMAC_k(x'\|y'\|z')$ respectively. Δch_{14} satisfies the target input difference of step 14 in Table 3 and ch_{14} satisfies the 25 sufficient conditions in Table 4.
2. The LPMACs of the message quadruple satisfy the equation

$$\begin{aligned} &LPMAC_k(x\|y\|z_1) - LPMAC_k(x\|y\|z) \\ &= LPMAC_k(x'\|y'\|z'_1) - LPMAC_k(x'\|y'\|z') \end{aligned}$$

We call the message pair (x, x') that satisfies the condition 1 and 2 above as a *W-Collision*.

For a random one-block message pair (x, x'), the condition 1 above is satisfied with probability $2^{-256} \times 2^{-25} = 2^{-281}$. If the LPMAC is based on 39-step SHA-256, the condition 2 can be satisfied when the two message pairs $(x\|y\|z, x\|y'\|z')$ and $(x\|y\|z_1, x\|y'\|z'_1)$ both follow the differential path from step 15 to 39 in Table 3, and the probability is 2^{-82}; Otherwise, if the LPMAC is based on a random function, the condition 2 holds with the average probability 2^{-256}.

Overall, if the MAC is a LPMAC based on 39-step SHA-256, the messages quadruple $(x\|y\|z, x\|y\|z_1, x'\|y'\|z', x'\|y'\|z'_1)$ satisfy the condition 1 and 2 with probability $2^{-281} \times 2^{-82} = 2^{-363}$, i.e., (x, x') consists of a *W-Collision* with probability 2^{-363}.

Then the distinguishing algorithm for the LPMAC based on 39-step SHA-256 can be described as follows:

1. Generate a message set S randomly, which consist of $2^{182.5}$ one-block messages. For each $x \in S$, query the MACs with $x\|y\|z$, $x\|y'\|z'$, $x\|y\|z_1$ and $x\|y'\|z'$ respectively, and compute the following two sets of differences:

$$\begin{aligned} S_1 &= \{LPMAC_k(x\|y\|z_1) - LPMAC_k(x\|y\|z) | x \in S\}, \\ S_2 &= \{LPMAC_k(x\|y'\|z'_1) - LPMAC_k(x\|y'\|z') | x \in S\}. \end{aligned}$$

   Find all the collisions (x, x') between the sets S_1 and S_2 by the birthday attack so that

$$\begin{aligned} &LPMAC_k(x\|y\|z_1) - LPMAC_k(x\|y\|z) \\ &= LPMAC_k(x'\|y'\|z'_1) - LPMAC_k(x'\|y'\|z'). \end{aligned}$$

   The set of all collisions is recorded as S_3.

2. For each collision $(x, x') \in S_3$, compute $LPMAC_k(x'\|y'\|z') - LPMAC_k(x\|y\|z)$ and denote it as δ. Choose 2^{42} different 64-bit messages pairs $(\overline{z}, \overline{z'})$ so that $(y\|\overline{z}, y'\|\overline{z'})$ satisfy the message differences in Table 3 and $y\|\overline{z}$ satisfy the message conditions in Table 5 and 5. Query the MACs of all the 2^{42} message pairs $(x'\|y'\|\overline{z'}, x\|y\|\overline{z})$ and observe whether $LPMAC_k(x'\|y'\|\overline{z'}) - LPMAC_k(x\|y\|\overline{z})$ is equivalent to δ.
3. Once a pair $(\overline{z}, \overline{z'})$ are found to match the difference δ, we conclude that the LPMAC is based on 39-step SHA-256; Otherwise, output the MAC is LPMAC based on a random function.

Complexity evaluation
In step 1, the number of queries is $2^{184.5}$ for all the $2^{182.5}$ messages. In addition, it needs a table look up with the table size $2^{182.5}$. For the $2^{182.5}$ messages, it can form about 2^{364} message pairs , so expected number of collisions in S_3 is about $2^{364-256} = 2^{108}$ in which about $2^{364-363} = 2$ *W-collisions* are included. In step 2, for each collision, it needs 2^{42} message pairs to verify whether the collision is a *W-collision* for a reasonable success rate, so the complexity in step 2 is about 2^{150} MAC queries. Therefore, the total time complexity of this attack is dominant in step 1 and it's about $2^{184.5}$ MAC queries.

Success rate:
We divide the success rate into two parts:

- If the MAC is LPMAC based on 39-step SHA-256, the attack succeeds when we distinct a *W-collision* from 2^{108} other collisions.
  The probability that there exists a *W-collision* among 2^{108} collisions is:

$$1 - (1 - \frac{1}{2^{363}})^{2^{364}} = 1 - \frac{1}{e^2} \approx 0.86$$

  The probability that the *W-collision* can be detected in step 3 is about

$$1 - (1 - \frac{1}{2^{41}})^{2^{42}} = 1 - \frac{1}{e^2} \approx 0.86$$

  This way, if the MAC is LPMAC based on 40-step SHA-256, a *W-collision* can be detected with probability $0.86 \times 0.86 \approx 0.72$.
- If the MAC is LPMAC based on a random function, the attack succeeds when no *W-collision* is detected. The success probability is about:

$$((1 - \frac{1}{2^{256}})^{2^{42}})^{2^{108}} \approx 1.$$

Therefore, the success rate of the whole attack is about

$$\frac{1}{2} \times 0.72 + \frac{1}{2} \times 1 = 0.87.$$

The success probability can be improved by increase the number of selected messages.

5 Conclusions

In this paper, we give the first distinguishing attack on the LPMAC based on 39-step SHA-256, and the attack is also applicable to the LPMAC based on step-reduced SHA-512. Our distinguishing attack on the LPMAC is not useful for the HMAC/NMAC because the non-zero output difference of the inner function is overshadowed by the outer function of HMAC/NMAC so that the *W-Collision* cannot be detected.

References

1. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
2. Bellare, M., Canetti, R., Krawczyk, H.: Keying Hash Functions for Message Authentication. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 1–15. Springer, Heidelberg (1996)
3. Chabaud, F., Joux, A.: Differential Collisions in SHA-0. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 56–71. Springer, Heidelberg (1998)
4. Contini, S., Yin, Y.L.: Forgery and Partial Key-Recovery Attacks on HMAC and NMAC Using Hash Collisions. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 37–53. Springer, Heidelberg (2006)
5. Dobbertin, H.: Cryptanalysis of MD4. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 53–69. Springer, Heidelberg (1996)
6. Fouque, P.-A., Leurent, G., Nguyen, P.Q.: Full Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 13–30. Springer, Heidelberg (2007)
7. Galvin, J.M., McCloghrie, K., Davin, J.R.: Secure management of SNMP networks. Integrated Network Management 11, 703–714 (1991)
8. Indesteege, S., Mendel, F., Preneel, B., Rechberger, C.: Collisions and other Non-Random Properties for Step-Reduced SHA-256. SAC 2008 (2008), http://eprint.iacr.org/2008/131.pdf
9. Kim, J., Biryukov, A., Preneel, B., Hong, S.: On the Security of HMAC and NMAC Based on HAVAL, MD4, MD5, SHA-0, and SHA-1. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 242–256. Springer, Heidelberg (2006)
10. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: Analysis of Step-Reduced SHA-256. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 126–143. Springer, Heidelberg (2006)
11. Nikolić, I., Biryukov, A.: Collisions for Step-Reduced SHA-256. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 1–16. Springer, Heidelberg (2008)
12. National Institute of Standards and Technology (NIST). FIPS- 180-2: Secure Hash Standard (August. 2002), http://www.itl.nist.gov/fipspubs/
13. Preneel, B., Oorschot, P.: MDx-MAC and building fast MACs from hash functions. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 1–14. Springer, Heidelberg (1995)
14. Sanadhya, S., Sarkar, P.: New Local Collisions for the SHA-2 Hash Family. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 193–205. Springer, Heidelberg (2007)

15. Sanadhya, S., Sarkar, P.: New Collision attacks Against Up To 24-step SHA-2. In: Chowdhury, D.R., Rijmen, V., Das, A. (eds.) INDOCRYPT 2008. LNCS, vol. 5365, pp. 91–103. Springer, Heidelberg (2008), http://eprint.iacr.org/2008/270.pdf
16. Rechberger, C., Rijmen, V.: On Authentication with HMAC and Non-Random Properties. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 39–57. Springer, Heidelberg (2007)
17. Tsudik, G.: Message authentication with one-way hash functions. ACM Computer Communications Review 22(5), 29–38 (1992)
18. Wang, L., Ohta, K., Kunihiro, N.: New Key-Recovery Attacks on HMAC/NMAC-MD4 and NMAC-MD5. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 237–253. Springer, Heidelberg (2008)
19. Wang, X.Y., Yu, H.B., Wang, W., Zhang, H.N., Zhan, T.: Cryptanalysis on HMAC/NMAC-MD5 and MD5-MAC. In: Eurocrypt 2009 (to appear, 2009)
20. Wang, X.Y., Lai, X.J.: Cryptanalysis for Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
21. Wang, X.Y., Yu, H.B.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)
22. Wang, X.Y., Feng, D.G., Yu, X.Y.: An attack on HAVAL function HAVAL-128. Science in China Ser. F Information Sciences 48(5), 1–12 (2005)
23. Wang, X.Y., Wang, W., Jia, K.T., Wang, M.Q.: New Distinguishing Attack on MAC using Secret-Prefix Method. In: FSE 2009 (to appear, 2009)
24. Wang, X.Y., Yu, H.B., Yin, Y.L.: Efficient Collision Search Attacks on SHA-0. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 1–16. Springer, Heidelberg (2005)
25. Wang, X.Y., Yin, Y.L., Yu, H.B.: Finding collisions on the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
26. Yu, H.B., Wang, G.L., Zhang, G.Y., Wang, X.Y.: The Second-Preimage Attack on MD4. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS, vol. 3810, pp. 1–12. Springer, Heidelberg (2005)
27. Yu, H.B., Wang, X.Y., Yun, A., Park, S.: Cryptanalysis of the Full HAVAL with 4 and 5 Passes. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 89–110. Springer, Heidelberg (2006)

Appendix

Table 2. The property for the round function $Ch(x, y, z)$ and $Maj(x, y, z)$ of SHA-256. $\Delta x = 1$ denotes x changes from 0 to 1, $\Delta x = -1$ denotes x changes from 1 to 0.

Δx	Δy	Δz	$\Delta Ch = 0$	$\Delta Ch = 1$	$\Delta Ch = -1$	$\Delta Maj = 0$	$\Delta Maj = 1$	$\Delta Maj = -1$
0	0	0	1	–	–	1	–	–
0	0	1	$x = 1$	$x = 0$	–	$x = y$	$x \neq y$	–
0	0	-1	$x = 1$	–	$x = 0$	$x = y$	–	$x \neq y$
0	1	0	$x = 0$	$x = 1$	–	$x = z$	$x \neq z$	–
0	1	1	–	1	–	–	1	–
0	1	-1	–	$x = 1$	$x = 0$	1	–	–
0	-1	0	$x = 0$	–	$x = 1$	$x = z$	–	$x \neq z$
0	-1	1	–	$x = 0$	$x = 1$	1	–	–
0	-1	-1	–	–	1	–	–	1
1	0	0	$y = z$	$y = 1$, $z = 0$	$y = 0$, $z = 1$	$y = z$	$y \neq z$	–
1	0	1	$y = 0$	$y = 1$	–	–	1	–
1	0	-1	$y = 1$	–	$y = 0$	1	–	–
1	1	0	$z = 1$	$z = 0$	–	–	1	–
1	1	1	–	1	–	–	1	–
1	1	-1	1	–	–	–	1	–
1	-1	0	$z = 0$	–	$z = 1$	1	–	–
1	-1	1	1	–	–	–	1	–
1	-1	-1	–	–	1	–	–	1
-1	0	0	$y = z$	$y = 0$, $z = 1$	$y = 1$, $z = 0$	$y = z$	–	$y \neq z$
-1	0	1	$y = 1$	$y = 0$	–	1	–	–
-1	0	-1	$y = 0$	–	$y = 1$	–	–	1
-1	1	0	$z = 0$	$z = 1$	–	1	–	–
-1	1	1	–	1	–	–	1	–
-1	1	-1	1	–	–	–	–	1
-1	-1	0	$z = 1$	–	$z = 0$	–	–	1
-1	-1	1	1	–	–	–	–	1
-1	-1	-1	–	–	1	–	–	1

Table 3. Differential path for the 39-step SHA-256

step	Δw	Δa	Δb	Δc	Δd	Δe	Δf	Δg	Δh
IV		-	-	-	-	-	-	-	-
1	92844891	-	-	-	-	-	-	-	-
2	50824014	-	-	-	-	-	-	-	-
3	84521222	-	-	-	-	-	-	-	-
4	4a901104	-	-	-	-	-	-	-	-
5	55410245	-	-	-	-	-	-	-	-
6	a1114484	-	-	-	-	-	-	-	-
7	90904111	-	-	-	-	-	-	-	-
8	04444441	-	-	-	-	-	-	-	-
9	12022882	-	-	-	-	-	-	-	-
10	00104011	-	-	-	-	-	-	-	-
11	00880080	-	-	-	-	-	-	-	-
12	81544280	-	-	-	-	-	-	-	-
13	0a020000	-	-	-	-	-	-	-	-
14	11002000	0	22140240	80000000	0	00082200	20040000	80000000	4411008c
15	80000000	0	0	22140240	80000000	0	00082200	20040000	80000000
16	0	80000000	0	0	22140240	0	0	00082200	20040000
17	0	0	80000000	0	0	02100040	0	0	00082200
18	0	0	0	80000000	0	0	02100040	0	0
19	0	0	0	0	80000000	0	0	02100040	0
20	0	0	0	0	0	80000000	0	0	02100040
21	0	0	0	0	0	0	80000000	0	0
22	0	0	0	0	0	0	0	80000000	0
23	0	0	0	0	0	0	0	0	80000000
24	80000000	0	0	0	0	0	0	0	0
25-38	0	0	0	0	0	0	0	0	0
39	$2^{13}+2^{24}+2^{28}$	$\Delta^- w_{38}$	0	0	0	$\Delta^- w_{38}$	0	0	0

Table 4. Sufficient conditions for steps $14 \sim 39$ on SHA-256

ch_{14}	$f_{14,10} = g_{14,10}$, $f_{14,14} = g_{14,14} \oplus 1$, $f_{14,19} = b_{14,19} \oplus 1$, $f_{14,20} = g_{14,20}$, $f_{14,30} = b_{14,30} \oplus 1$, $a_{14,7} = c_{14,7}$, $a_{14,10} = c_{14,10}$, $a_{14,19} = c_{14,19}$, $a_{14,21} = c_{14,21}$, $a_{14,26} = c_{14,26}$, $a_{14,30} = c_{14,30}$, $a_{14,32} = b_{14,32}$, $e_{14,10} = b_{14,10} \oplus 1$, $e_{14,14} = e_{14,1} \oplus h_{14,8}$, $e_{14,15} = e_{14,1} \oplus e_{14,6} \oplus e_{14,2} \oplus h_{14,8} \oplus h_{14,27}$, $e_{14,19} = 0$, $e_{14,20} = e_{14,15} \oplus e_{14,2} \oplus h_{14,8} \oplus 1$, $e_{14,23} = e_{14,9} \oplus e_{14,14} \oplus h_{14,3} \oplus e_{14,10} \oplus h_{14,17}$, $e_{14,24} = e_{14,5} \oplus e_{14,10} \oplus h_{14,31} \oplus 1$, $e_{14,25} = e_{14,20} \oplus e_{14,7} \oplus e_{14,14} \oplus g_{14,14} \oplus 1$, $e_{14,27} = e_{14,14} \oplus h_{14,21} \oplus 1$, $e_{14,28} = e_{14,9} \oplus e_{14,14} \oplus h_{14,3} \oplus 1$, $e_{14,29} = e_{14,10} \oplus e_{14,15} \oplus h_{14,4} \oplus 1$, $e_{14,30} = 0$, $e_{14,32} = 0$
ch_{15}	$a_{15,32} = a_{14,32}$, $a_{15,7} = a_{14,7}$, $a_{15,10} = a_{14,10}$, $a_{15,19} = a_{14,19}$, $a_{15,21} = a_{14,21}$, $a_{15,26} = a_{14,26}$, $a_{15,30} = a_{14,30}$, $e_{15,10} = 0$, $e_{15,14} = 0$, $e_{15,20} = 0$, $e_{15,19} = 1$, $e_{15,30} = 1$
ch_{16}	$a_{16,32} = a_{16,12} \oplus a_{16,23} \oplus c_{15,10}$, $a_{16,23} = a_{16,21} \oplus a_{16,12} \oplus a_{16,9} \oplus c_{15,10} \oplus c_{15,19}$, $a_{16,21} = a_{16,9} \oplus a_{16,11} \oplus a_{16,20} \oplus c_{15,30} \oplus c_{15,19}$, $e_{16,7} = e_{15,7}$, $e_{16,10} = 0$, $e_{16,14} = 1$, $e_{16,20} = 1$, $e_{16,21} = e_{15,21}$, $e_{16,26} = e_{15,26}$
ch_{17}	$e_{17,7} = c_{15,7}$, $e_{17,16} = e_{17,21} \oplus e_{17,3} \oplus f_{15,10} \oplus 1$, $e_{17,21} = c_{15,21}$, $e_{17,25} = e_{17,20} \oplus e_{17,7} \oplus f_{15,14} \oplus 1$, $e_{17,26} = c_{15,26}$, $e_{17,31} = e_{17,26} \oplus e_{17,13} \oplus f_{15,20} \oplus 1$, $a_{17,32} = a_{15,32}$
$ch_{18} \sim ch_{19}$	$a_{18,32} = a_{17,32}$, $e_{18,7} = 0$, $e_{18,21} = 0$, $e_{18,26} = 0$, $e_{19,7} = 1$, $e_{19,21} = 1$, $e_{19,26} = 1$, $e_{19,32} = e_{18,32}$
$ch_{20} \sim ch_{22}$	$e_{20,19} = e_{20,13} \oplus e_{20,18} \oplus e_{20,7} \oplus e_{20,5} \oplus e_{17,26}$, $e_{20,28} = e_{20,13} \oplus e_{20,18} \oplus e_{20,7} \oplus e_{20,14} \oplus e_{17,21}$, $e_{20,32} = e_{20,13} \oplus e_{20,18} \oplus e_{17,7} \oplus 1$, $e_{21,32} = 0$, $e_{22,32} = 0$

Table 5. Conditions on messages for steps $1 \sim 39$ on SHA-256

w_i	Conditions	Numbers
w_1	$w_{1,16} = w_{1,5} \oplus w_{0,29} \oplus 1$, $w_{1,20} = w_{1,9} \oplus w_{1,5} \oplus w_{0,1} \oplus 1$, $w_{1,22} = w_{1,15} \oplus w_{1,11} \oplus w_{1,5} \oplus w_{0,8} \oplus w_{0,19}$, $w_{1,24} = w_{1,13} \oplus w_{1,9} \oplus w_{0,5} \oplus 1$, $w_{1,26} = w_{1,15} \oplus w_{1,11} \oplus w_{0,8} \oplus 1$, $w_{1,29} = w_{1,1} \oplus w_{1,12} \oplus w_{0,26} \oplus 1$, $w_{1,30} = w_{1,19} \oplus w_{1,15} \oplus w_{0,12} \oplus 1$, $w_{1,31} = w_{1,10} \oplus w_{1,27} \oplus w_{0,24} \oplus 1$, $w_{1,32} = w_{1,4} \oplus w_{1,15} \oplus w_{0,29}$	9
w_2	$w_{2,14} = w_{2,4} \oplus w_{2,6} \oplus w_{1,3} \oplus w_{1,5} \oplus w_{1,18}$, $w_{2,15} = w_{2,11} \oplus w_{2,13} \oplus w_{2,6} \oplus w_{2,4} \oplus w_{1,29} \oplus w_{1,31} \oplus w_{1,5} \oplus w_{1,24}$, $w_{2,17} = w_{2,6} \oplus w_{1,31}$, $w_{2,21} = w_{2,10} \oplus w_{2,6} \oplus w_{1,3} \oplus 1$, $w_{2,22} = w_{2,1} \oplus w_{2,18} \oplus w_{1,15} \oplus 1$, $w_{2,23} = w_{2,12} \oplus w_{2,8} \oplus w_{1,5}$, $w_{2,24} = w_{2,13} \oplus w_{2,9} \oplus w_{1,5}$, $w_{2,25} = w_{2,14} \oplus w_{2,10} \oplus w_{1,5}$, $w_{2,27} = w_{2,16} \oplus w_{2,12} \oplus w_{1,5}$, $w_{2,28} = w_{2,17} \oplus w_{2,13} \oplus w_{1,5}$, $w_{2,29} = w_{2,18} \oplus w_{2,14} \oplus w_{1,5} \oplus 1$, $w_{2,32} = w_{2,4} \oplus w_{2,15} \oplus w_{1,29} \oplus 1$	12
w_3	$w_{3,20} = w_{3,9} \oplus w_{3,5} \oplus w_{2,2}$, $w_{3,21} = w_{3,10} \oplus w_{3,6} \oplus w_{2,2} \oplus 1$, $w_{3,24} = w_{3,13} \oplus w_{3,9} \oplus w_{2,6} \oplus 1$, $w_{3,25} = w_{3,4} \oplus w_{3,21} \oplus w_{2,18} \oplus 1$, $w_{3,27} = w_{3,20} \oplus w_{3,16} \oplus w_{3,10} \oplus w_{2,23} \oplus w_{2,13} \oplus 1$, $w_{3,30} = w_{3,2} \oplus w_{3,13} \oplus w_{2,27} \oplus 1$, $w_{3,31} = w_{3,20} \oplus w_{3,16} \oplus w_{2,13}$, $w_{3,32} = w_{3,21} \oplus w_{3,17} \oplus w_{2,13}$	8
w_4	$w_{4,13} = w_{4,10} \oplus w_{4,12} \oplus w_{4,6} \oplus w_{4,7} \oplus w_{4,3} \oplus w_{3,24} \oplus w_{3,31} \oplus w_{3,21} \oplus 1$, $w_{4,17} = w_{4,6} \oplus w_{3,31}$, $w_{4,18} = w_{4,11} \oplus w_{4,7} \oplus w_{3,3} \oplus w_{4,1} \oplus w_{3,13}$, $w_{4,19} = w_{4,12} \oplus w_{4,8} \oplus w_{3,3} \oplus w_{4,2} \oplus w_{3,13} \oplus 1$, $w_{4,20} = w_{3,13} \oplus w_{4,10} \oplus w_{4,12} \oplus w_{3,9} \oplus w_{3,24} \oplus 1$, $w_{4,21} = w_{4,10} \oplus w_{4,6} \oplus w_{3,3}$, $w_{4,22} = w_{4,11} \oplus w_{4,7} \oplus w_{3,3}$, $w_{4,23} = w_{4,12} \oplus w_{4,8} \oplus w_{3,3} \oplus 1$, $w_{4,24} = w_{4,3} \oplus w_{4,20} \oplus w_{3,13} \oplus 1$, $w_{4,27} = w_{4,16} \oplus w_{4,12} \oplus w_{3,9}$, $w_{4,28} = w_{4,17} \oplus w_{4,13} \oplus w_{3,9}$, $w_{4,29} = w_{4,18} \oplus w_{4,14} \oplus w_{3,9} \oplus 1$, $w_{4,31} = w_{4,20} \oplus w_{4,16} \oplus w_{3,13}$, $w_{4,32} = w_{4,21} \oplus w_{4,17} \oplus w_{3,13}$	14

Table 5. (*continued*)

w_5	$w_{5,17} = w_{5,6} \oplus w_{4,31} \oplus 1$, $w_{5,19} = w_{5,8} \oplus w_{5,4} \oplus w_{4,1} \oplus 1$, $w_{5,21} = w_{5,10} \oplus w_{5,6} \oplus w_{4,3}$, $w_{5,22} = w_{5,11} \oplus w_{5,7} \oplus w_{4,3}$, $w_{5,23} = w_{5,12} \oplus w_{5,8} \oplus w_{4,3} \oplus 1$, $w_{5,24} = w_{5,3} \oplus w_{5,20} \oplus w_{4,17} \oplus 1$, $w_{5,25} = w_{5,14} \oplus w_{5,10} \oplus w_{4,7} \oplus 1$, $w_{5,26} = w_{5,2} \oplus w_{5,13} \oplus w_{5,9} \oplus w_{4,27} \oplus w_{4,23} \oplus 1$, $w_{5,28} = w_{5,17} \oplus w_{5,13} \oplus w_{4,10} \oplus 1$, $w_{5,30} = w_{5,2} \oplus w_{5,13} \oplus w_{4,27}$, $w_{5,31} = w_{5,3} \oplus w_{5,14} \oplus w_{4,27} \oplus 1$	11
w_6	$w_{6,15} = w_{6,14} \oplus w_{6,11} \oplus w_{6,10} \oplus w_{6,8} \oplus w_{6,6} \oplus w_{6,5} \oplus w_{6,1} \oplus w_{6,4} \oplus w_{5,15} \oplus w_{5,8} \oplus w_{5,11} \oplus w_{5,3} \oplus w_{5,21}$, $w_{6,16} = w_{6,5} \oplus w_{5,30} \oplus 1$, $w_{6,18} = w_{6,15} \oplus w_{6,11} \oplus w_{6,5} \oplus w_{6,1} \oplus w_{5,15} \oplus w_{5,17} \oplus w_{5,8} \oplus 1$, $w_{6,19} = w_{6,9} \oplus w_{6,11} \oplus w_{5,11} \oplus w_{5,8} \oplus w_{5,21} \oplus 1$, $w_{6,21} = w_{6,10} \oplus w_{6,6} \oplus w_{5,3} \oplus 1$, $w_{6,22} = w_{6,1} \oplus w_{6,18} \oplus w_{5,15} \oplus 1$, $w_{6,24} = w_{6,3} \oplus w_{6,20} \oplus w_{5,17}$, $w_{6,25} = w_{6,4} \oplus w_{6,21} \oplus w_{5,17}$, $w_{6,26} = w_{6,15} \oplus w_{6,11} \oplus w_{5,8} \oplus 1$, $w_{6,28} = w_{6,7} \oplus w_{6,24} \oplus w_{5,21}$, $w_{6,29} = w_{6,14} \oplus w_{6,18} \oplus w_{5,11}$, $w_{6,30} = w_{6,19} \oplus w_{6,15} \oplus w_{5,11} \oplus 1$, $w_{6,32} = w_{6,11} \oplus w_{6,28} \oplus w_{5,25} \oplus 1$	13
w_7	$w_{7,19} = w_{7,8} \oplus w_{7,4} \oplus w_{6,1} \oplus 1$, $w_{7,22} = w_{7,18} \oplus w_{7,1} \oplus w_{6,15} \oplus 1$, $w_{7,23} = w_{7,12} \oplus w_{7,8} \oplus w_{6,5} \oplus 1$, $w_{7,24} = w_{7,4} \oplus w_{7,15} \oplus w_{7,11} \oplus w_{7,7} \oplus w_{6,29} \oplus w_{6,21} \oplus w_{6,24}$, $w_{7,27} = w_{7,16} \oplus w_{7,12} \oplus w_{6,9} \oplus 1$, $w_{7,28} = w_{7,7} \oplus w_{7,24} \oplus w_{6,21} \oplus 1$, $w_{7,29} = w_{7,1} \oplus w_{7,12} \oplus w_{6,24} \oplus 1$, $w_{7,31} = w_{7,10} \oplus w_{7,27} \oplus w_{6,24}$, $w_{7,32} = w_{7,4} \oplus w_{7,15} \oplus w_{6,29} \oplus 1$	9
w_8	$w_{8,15} = w_{8,2} \oplus w_{8,11} \oplus w_{8,13} \oplus w_{8,9} \oplus w_{7,23} \oplus w_{7,27} \oplus w_{7,7} \oplus 1$, $w_{8,18} = w_{8,2} \oplus w_{8,13} \oplus w_{8,9} \oplus w_{8,5} \oplus w_{8,1} \oplus w_{7,15} \oplus w_{7,19} \oplus w_{7,23} \oplus w_{7,27}$, $w_{8,19} = w_{8,8} \oplus w_{8,4} \oplus w_{7,1} \oplus 1$, $w_{8,22} = w_{8,1} \oplus w_{8,18} \oplus w_{7,15}$, $w_{8,23} = w_{8,2} \oplus w_{8,19} \oplus w_{7,15} \oplus 1$, $w_{8,25} = w_{8,14} \oplus w_{8,10} \oplus w_{7,7}$, $w_{8,26} = w_{8,5} \oplus w_{8,22} \oplus w_{7,19} \oplus 1$, $w_{8,27} = w_{8,16} \oplus w_{8,12} \oplus w_{7,7} \oplus 1$, $w_{8,29} = w_{8,18} \oplus w_{8,14} \oplus w_{7,11} \oplus 1$, $w_{8,30} = w_{8,9} \oplus w_{8,26} \oplus w_{7,23} \oplus 1$, $w_{8,31} = w_{8,3} \oplus w_{8,14} \oplus w_{7,27} \oplus 1$	11
w_9	$w_{9,1} = w_{0,1}$, $w_{9,5} = w_{0,5}$, $w_{9,10} = w_{9,6} \oplus w_{8,2} \oplus w_{1,28} \oplus w_{1,7} \oplus w_{1,24}$, $w_{9,15} = w_{0,15} \oplus 1$, $w_{9,16} = w_{9,5} \oplus w_{8,29} \oplus 1$, $w_{9,17} = w_{9,10} \oplus w_{9,6} \oplus w_{9,4} \oplus w_{9,15} \oplus w_{8,2} \oplus w_{8,14} \oplus w_{8,29} \oplus 1$, $w_{9,18} = w_{9,1} \oplus w_{9,15} \oplus w_{9,11} \oplus w_{9,5} \oplus w_{8,8} \oplus w_{8,18} \oplus w_{8,14} \oplus 1$, $w_{9,20} = w_{9,9} \oplus w_{9,5} \oplus w_{8,2}$, $w_{9,21} = w_{1,28} \oplus w_{1,7} \oplus w_{1,24} \oplus 1$, $w_{9,22} = w_{9,1} \oplus w_{9,18} \oplus w_{8,14} \oplus 1$, $w_{9,25} = w_{9,4} \oplus w_{9,21} \oplus w_{8,18}$, $w_{9,26} = w_{9,15} \oplus w_{9,11} \oplus w_{8,8} \oplus 1$, $w_{9,29} = w_{9,1} \oplus w_{9,12} \oplus w_{8,26} \oplus 1$, $w_{9,30} = w_{9,19} \oplus w_{9,15} \oplus w_{8,12} \oplus 1$, $w_{9,32} = w_{9,21} \oplus w_{9,17} \oplus w_{8,14}$	15
w_{10}	$w_{10,8} = w_{1,5}$, $w_{10,9} = w_{2,27} \oplus w_{10,5} \oplus w_{2,6} \oplus w_{2,23} \oplus w_{9,1}$, $w_{10,13} = w_{10,9} \oplus w_{9,5} \oplus w_{1,24} \oplus 1$, $w_{10,19} = w_{10,8} \oplus w_{10,4} \oplus w_{9,1}$, $w_{10,20} = w_{2,27} \oplus w_{2,6} \oplus w_{2,23} \oplus 1$, $w_{10,23} = w_{10,12} \oplus w_{10,8} \oplus w_{9,5}$, $w_{10,24} = w_{1,24}$, $w_{10,28} = w_{10,7} \oplus w_{10,24} \oplus w_{9,21} \oplus 1$	8
w_{11}	$w_{11,8} = w_{3,26} \oplus w_{3,15} \oplus w_{3,11} \oplus 1$, $w_{11,10} = w_{2,10} \oplus 1$, $w_{11,15} = w_{2,13} \oplus 1$, $w_{11,19} = w_{3,26} \oplus w_{3,5} \oplus w_{3,22} \oplus 1$, $w_{11,21} = w_{2,21} \oplus 1$, $w_{11,23} = w_{2,23}$, $w_{11,25} = w_{3,32} \oplus w_{3,11} \oplus w_{3,28} \oplus 1$, $w_{11,26} = w_{11,15} \oplus w_{11,11} \oplus w_{10,8} \oplus 1$, $w_{11,27} = w_{11,6} \oplus w_{11,23} \oplus w_{10,20} \oplus 1$, $w_{11,31} = w_{11,10} \oplus w_{11,27} \oplus w_{10,24}$, $w_{11,32} = w_{11,11} \oplus w_{11,28} \oplus w_{10,24} \oplus 1$	11
w_{12}	$w_{12,5} = w_{12,1} \oplus w_{11,15} \oplus w_{11,19} \oplus w_{4,25} \oplus w_{4,4} \oplus w_{4,21} \oplus w_{3,26}$, $w_{12,15} = w_{12,11} \oplus w_{11,8} \oplus w_{3,26}$, $w_{12,17} = w_{12,13} \oplus w_{11,10} \oplus w_{3,28} \oplus 1$, $w_{12,18} = w_{4,25} \oplus w_{4,4} \oplus w_{4,21} \oplus 1$, $w_{12,22} = w_{12,1} \oplus w_{12,18} \oplus w_{11,15} \oplus 1$, $w_{12,24} = w_{12,17} \oplus w_{12,13} \oplus w_{12,7} \oplus w_{11,10} \oplus w_{11,21} \oplus 1$, $w_{12,26} = w_{3,26} \oplus 1$, $w_{12,28} = w_{3,28} \oplus 1$, $w_{12,29} = w_{12,18} \oplus w_{12,14} \oplus w_{11,10} \oplus 1$, $w_{12,30} = w_{12,9} \oplus w_{12,26} \oplus w_{11,23} \oplus 1$, $w_{12,32} = w_{12,11} \oplus w_{12,28} \oplus w_{11,25} \oplus 1$	11
w_{13}	$w_{13,12} = w_{13,1} \oplus w_{12,26} \oplus w_{4,29}$, $w_{13,14} = w_{5,32} \oplus w_{5,21} \oplus w_{5,17} \oplus 1$, $w_{13,21} = w_{13,4} \oplus w_{12,18} w_{4,25}$, $w_{13,25} = w_{4,25} \oplus 1$, $w_{13,29} = w_{4,29} \oplus 1$, $w_{13,31} = w_{13,3} \oplus w_{13,14} \oplus w_{12,28} \oplus 1$	6
w_{14}	$w_{14,15} = w_{14,7} \oplus w_{14,9} \oplus w_{14,4} \oplus w_{13,29} \oplus w_{1,29} \oplus w_{1,8} \oplus w_{1,25}$, $w_{14,21} = w_{14,4} \oplus w_{14,15} \oplus w_{14,17} \oplus w_{13,14} \oplus w_{13,29}$, $w_{14,25} = w_{14,7} \oplus w_{14,9} \oplus w_{14,2} \oplus w_{1,29} \oplus w_{1,8} \oplus w_{1,25} \oplus w_{1,22} \oplus w_{1,1} \oplus w_{1,18}$, $w_{14,28} = w_{14,21} \oplus w_{14,17} \oplus w_{14,11} \oplus w_{13,14} \oplus w_{13,25}$, $w_{14,30} = w_{14,7} \oplus w_{14,9} \oplus w_{14,23} \oplus w_{1,8} \oplus w_{1,25} \oplus w_{1,20} \oplus w_{1,31} \oplus w_{1,16} \oplus w_{1,29}$, $w_{14,32} = w_{14,21} \oplus w_{14,17} \oplus w_{13,14} \oplus 1$	6
w_{23}	$w_{23,15} = w_{23,7} \oplus w_{23,9} \oplus w_{23,4} \oplus w_{10,29} \oplus w_{10,8} \oplus w_{10,25} \oplus 1$, $w_{23,21} = w_{23,4} \oplus w_{23,15} \oplus w_{23,17}$, $w_{23,25} = w_{23,4} \oplus w_{23,15} \oplus w_{23,2} \oplus w_{9,15} \oplus 1$, $w_{23,28} = w_{23,4} \oplus w_{23,15} \oplus w_{23,11}$, $w_{23,30} = w_{23,2} \oplus w_{23,25} \oplus w_{23,23} \oplus w_{10,20} \oplus w_{10,31} \oplus w_{10,16} \oplus w_{9,15}$, $w_{23,32} = w_{23,4} \oplus w_{23,15}$	6

Inside the Hypercube

Jean-Philippe Aumasson[1,⋆], Eric Brier[3], Willi Meier[1,⋆⋆], María Naya-Plasencia[2,⋆⋆⋆], and Thomas Peyrin[3]

[1] FHNW, Windisch, Switzerland
[2] INRIA project-team SECRET, France
[3] Ingenico, France

Some force inside the Hypercube occasionally manifests itself with deadly results.
http://www.staticzombie.com/2003/06/cube_2_hypercube.html

Abstract. Bernstein's CubeHash is a hash function family that includes four functions submitted to the NIST Hash Competition. A CubeHash function is parametrized by a number of rounds r, a block byte size b, and a digest bit length h (the compression function makes r rounds, while the finalization function makes $10r$ rounds). The 1024-bit internal state of CubeHash is represented as a five-dimensional hypercube. The submissions to NIST recommends $r = 8$, $b = 1$, and $h \in \{224, 256, 384, 512\}$.

This paper presents the first external analysis of CubeHash, with

- improved standard generic attacks for collisions and preimages
- a multicollision attack that exploits fixed points
- a study of the round function symmetries
- a preimage attack that exploits these symmetries
- a practical collision attack on a weakened version of CubeHash
- a study of fixed points and an example of nontrivial fixed point
- high-probability truncated differentials over 10 rounds

Since the first publication of these results, several collision attacks for reduced versions of CubeHash were published by Dai, Peyrin, et al. Our results are more general, since they apply to any choice of the parameters, and show intrinsic properties of the CubeHash design, rather than attacks on specific versions.

1 CubeHash

Bernstein's CubeHash is a hash function family submitted to the NIST Hash Competition. A CubeHash function is parametrized by a number of rounds r, a block byte size b, and a digest bit length h; the 1024-bit internal state of CubeHash is viewed as a five dimensional hypercube. The submissions to NIST recommends $r = 8$, $b = 1$, and $h \in \{224, 256, 384, 512\}$.

CubeHash computes a message digest as follows:

⋆ Supported by the Swiss National Science Foundation under project no. 113329.
⋆⋆ Supported by GEBERT RÜF STIFTUNG, project no. GRS-069/07.
⋆⋆⋆ Supported in part by the French Agence Nationale de la Recherche under contract ANR-06-SETI-013-RAPIDE.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 202–213, 2009.

- initialize a 1024-bit state as a function of (h, b, r)
- append to the message a 1 bit and enough 0 bits to reach a multiple of $8b$ bits
- for each b-byte message block:
  - xor the block into the first b bytes of the state
  - transform the state through the r-round T function
- xor a 1 bit with the 993rd bit of the state
- transform the state through $10r$-round T
- output the first h bits of the state

Let $x[0], \ldots, x[31]$ represent the 1024-bit state as an array of 32-bit words. The transform function T makes r identical rounds, where each round computes (see also Fig. 1):

for $i = 0, \ldots, 15$: $x[i + 16] = x[i + 16] + x[i]$
for $i = 0, \ldots, 15$: $y[i \oplus 8] = x[i]$
for $i = 0, \ldots, 15$: $x[i] = y[i] \lll 7$
for $i = 0, \ldots, 15$: $x[i] = x[i] \oplus x[i + 16]$
for $i = 0, \ldots, 15$: $y[i \oplus 2] = x[i + 16]$
for $i = 0, \ldots, 15$: $x[i + 16] = y[i]$
for $i = 0, \ldots, 15$: $x[i + 16] = x[i + 16] + x[i]$
for $i = 0, \ldots, 15$: $y[i \oplus 4] = x[i]$
for $i = 0, \ldots, 15$: $x[i] = y[i] \lll 11$
for $i = 0, \ldots, 15$: $x[i] = x[i] \oplus x[i + 16]$
for $i = 0, \ldots, 15$: $y[i \oplus 1] = x[i + 16]$
for $i = 0, \ldots, 15$: $x[i + 16] = y[i]$

See [5] for a more detailed description of CubeHash.

This paper presents the first external analysis of CubeHash, with

- improved standard generic attacks for collisions and preimages
- a multicollision attack that exploits fixed points
- a study of the round function symmetries
- a preimage attack that exploits these symmetries
- a practical collision attack on a weakened version of CubeHash
- a study of fixed points and an example of nontrivial fixed point
- high-probability truncated differentials over the 10-round transform

After the first publication of this article [2], Dai, Peyrin, et al. presented a series of collision attacks [1, 8, 7, 6] on reduced versions of CubeHash. Their best results (as of Feb. 6) are an example of collision on CubeHash3/64 and a collision attack on CubeHash4/3 in about 2^{207} simple operations [6]. Our results, however, are more general, since they apply to any choice of the parameters, and show intrinsic properties of the CubeHash design, rather than attacks on specific versions.

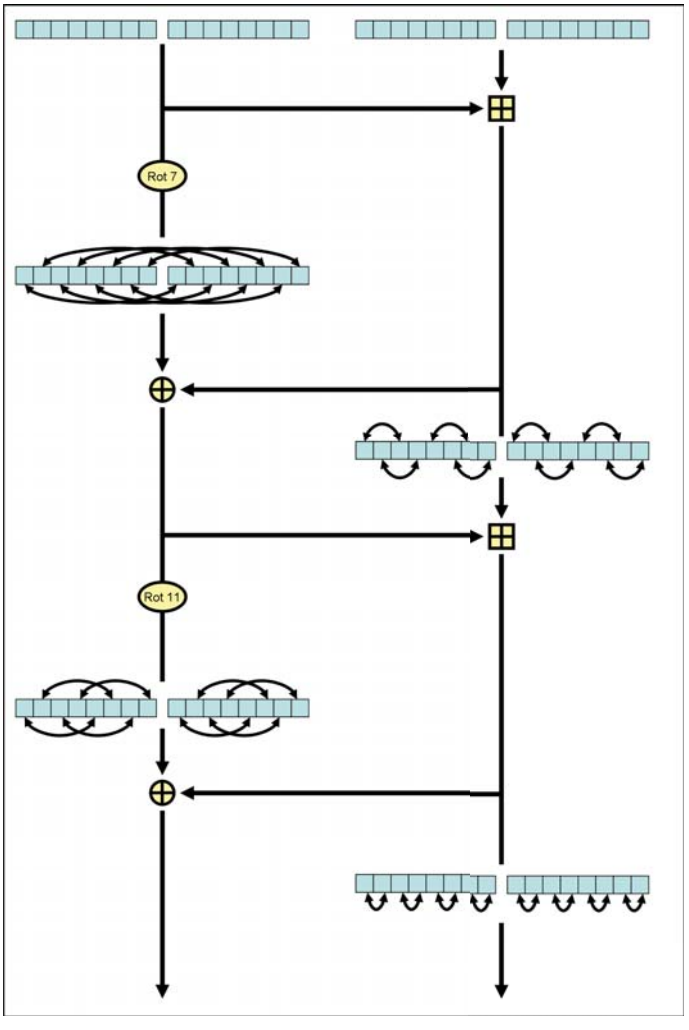


Fig. 1. Schematic view of a CubeHash round

2 Improved Standard Generic Attacks

The author of CubeHash presented [3] the following "standard preimage attack":

- from (h, b, r) compute the initial state S_0
- from the h-bit image plus some arbitrary $(1024-h)$ bits, invert $10r$ rounds and the "xor 1" to get a state S_f before finalization
- find two n-block sequences that map S_0 (forward) and S_f (backward), respectively, to two states that share the last $(1024-8b)$ bits

There are 2^{nb} possible n-block inputs and one looks for a collision over $(1024-8b)$ bits. For a success chance $1-1/e \approx 0.63$ one thus requires 2^{512-4b} trials in each

direction, that is, $2nb > 1024 - 8b$, i.e., $n > 512/b - 4$. In total the number of evaluations of T is approximately

$$2 \times \left(\frac{512}{b} - 4 \right) \times 2^{512-4b} \approx 2^{522-4b-\log b} .$$

Furthermore, [3] estimates that each round of T needs 2^{11} "bit operations"; the above formula gives about $2^{533-4b-\log b+\log r}$ bit operations.

A speed-up of the above attack can be obtained by searching a collision not only in the states resulting of a n-block computation, but in every distinct state reached (i.e. also with the intermediate states). This is made possible by the absence of message length padding. Each call to T gives a new candidate for the collision search; we thus get rid of the $(512/b - 4)$ multiplicative factor in the cost estimate. This gives a cost of

$$2 \times 2^{512-4b} = 2^{513-4b}$$

evaluations of T, i.e. $2^{524-4b+\log r}$ bit operations.

The proposed CubeHash-512 has $(h, b, r) = (512, 1, 8)$, our attack thus makes 2^{523} bit operations, against 2^{532} with the original attack. If $r = 8$, our attack needs $b > 3$ to make less than 2^{512} bit operations, against $b > 5$ with the original preimage attack. It is to note that these estimates exclude the non-negligible communication costs.

One can use the same trick to speed-up the standard collision attack [3]; the cost in T evaluations then drops from $2^{521-4b-\log b}$ to 2^{512-4b}.

3 Narrow-Pipe Multicollisions

Based on the "narrow-pipe" attacks in [4], we show a multicollision attack on CubeHash faster than Joux's [10] or birthday [9,12] methods (for large b's). Our attack requires the same amount of computation as narrow-pipe collisions. It exploits the fact that the null state is a fixed point for the compression function T (regardless of r), and that the message padding does not include the message length.

Starting from an initial state S_0 derived from (h, b, r), one finds two n-block sequences m and m' that map S_0 (forward) and the zero state (backward), respectively, to two states that share the last $(1024 - 8b)$ bits. One finds a connection of the form

$$\begin{aligned} S_0 \oplus m_1 &\xrightarrow{T} S_1 \\ S_1 \oplus m_2 &\xrightarrow{T} \cdots \\ &\cdots \\ \cdots &\xrightarrow{T} S'_1 \\ S'_1 \oplus m'_2 &\xrightarrow{T} 0 \oplus m'_1 \end{aligned}$$

Once a path to the zero state is found, one can add an arbitrary number of zero message blocks to maintain a zero state. Colliding messages are of the form

$$m\|m'\|0\|0\|\ldots\|0\|\bar{m},$$

where $\bar{m}$ is an arbitrary sequence of blocks.

Using the technique of §2, this multicollision attack requires approximately 2^{513-4b} evaluations of T. In comparison, a birthday attack finds a k-collision in $(k! \times 2^{n(k-1)})^{1/k}$ trials, and Joux's attacks in $\log k \times 2^{4(128-b)}$. For example, with $h = 512$ and $b = 112$, our attack finds 2^{64}-collisions within 2^{65} calls to T, against $> 2^{512}$ for a birthday attack and 2^{70} for Joux's.

4 State Symmetries

The documentation of CubeHash mentions [5, p.3] the existence of symmetries through the round function, and states that the initialization of CubeHash was designed to avoid them. However [5] gives no detail on those symmetries. In this section, we provide a reasoning that finds all symmetries inherent in the transformation T. In total we are able to show 15 symmetry classes of 2^{512} states each, and show how to exploit these.

4.1 Symmetry Classes

If a 32-word state x satisfies $x[0] = x[1]$, $x[2] = x[3]$, ..., $x[30] = x[31]$, then this property is preserved through the transformation T, with probability equal to 1, for any number of rounds. One can represent this symmetry with the pattern (each letter stands for a 32-bit word):

AABBCCDD EEFFGGHH IIJJKKLL MMNNOOPP .

In total we found 15 classes of symmetry:

C_1 : AABBCCDD EEFFGGHH IIJJKKLL MMNNOOPP
C_2 : ABABCDCD EFEFGHGH IJIJKLKL MNMNOPOP
C_3 : ABBACDDC EFFEGHHG IJJIKLLK MNNMOPPO
C_4 : ABCDABCD EFGHEFGH IJKLIJKL MNOPMNOP
C_5 : ABCDBADC EFGHFEHG IJKLJILK MNOPNMPO
C_6 : ABCDCDAB EFGHGHEF IJKLKLIJ MNOPOPMN
C_7 : ABCDDCBA EFGHHGFE IJKLLKJI MNOPPONM
C_8 : ABCDEFGH ABCDEFGH IJKLMNOP IJKLMNOP
C_9 : ABCDEFGH BADCFEHG IJKLMNOP JILKNMPO
C_{10} : ABCDEFGH CDABGHEF IJKLMNOP KLIJOPMN
C_{11} : ABCDEFGH DCBAHGFE IJKLMNOP LKJIPONM
C_{12} : ABCDEFGH EFGHABCD IJKLMNOP MNOPIJKL
C_{13} : ABCDEFGH FEHGBADC IJKLMNOP NMPOJILK
C_{14} : ABCDEFGH GHEFCDAB IJKLMNOP OPMNKLIJ
C_{15} : ABCDEFGH HGFEDCBA IJKLMNOP PONMLKJI

Each class contains 2^{512} states. If a state belongs to several classes, then its image under T also belongs to these classes; for example if $S \in (C_i \cap C_j)$, then $T(S) \in (C_i \cap C_j)$. We have

$$|C_i \cap C_j| \leq 2^{256} \ .$$

By the inclusion-exclusion principle, the number of distinct symmetric states is

$$\left|\cup_{i=1}^{15} C_i\right| = 15 \times 2^{512} - 70 \times 2^{256} + 120 \times 2^{128} - 64 \times 2^{64} \approx 2^{516} \ .$$

Note that symmetry is not preserved by the finalization procedure of CubeHash (the "xor 1" breaks any of the above symmetries).

4.2 Finding All Symmetry Classes

Now we prove that the classes $C_1, \ldots, C_{15}$ capture all the possible symmetries of CubeHash's transform T. A symmetry class can be represented as a set of pairs (i, j), where each (i, j) means $x[i] = x[j]$. For example, C_1 can be described by the set

(0,1) (2,3) (4,5) (6,7) (8,9) (10,11) (12,13) (14,15)
(16,17) (18,19) (20,21) (22,23) (24,25) (26,27) (28,29) (30,31)

We want a symmetry class to propagate through one round of the scheme with probability equal to one. It is easy to see that this condition imposes that the equality constraints at the left and at the right branch of the scheme must be the same (because of the intra-word rotations that are only present in the left branch of the scheme). That is, for any relation (i, j) with $0 \leq i, j \leq 15$, we must also have the relation $(i + 16, j + 16)$. In other words, a symmetry pattern is the same for the left and for the right branch. We thus only need to consider 16-word symmetry patterns.

To describe all possible symmetries, we start by fixing $(0, k)$, for a fixed k in $\{1, \ldots, 15\}$. We then compute T backwards to indentify the relations implied by $(0, k)$: the first substitution and xor encountered force us to have

$$(0, k)\ (4, k \oplus 4).$$

Then, the second substitution and the modular addition force to have (note that the intra-word rotations can be omitted since they leave the symmetry pattern unchanged)

$$(0, k)\ (4, k \oplus 4)\ (1, k \oplus 1)\ (5, k \oplus 5).$$

The third substitution and xor yield

$$(0, k) \quad (4, k \oplus 4)\ (1, k \oplus 1)\ (5, k \oplus 5)$$
$$(2, k \oplus 2)\ (6, k \oplus 6)\ (3, k \oplus 3)\ (7, k \oplus 7).$$

Finally, the last substitution and the modular addition imply

$$\begin{array}{cccc} (0,k) & (4,k\oplus 4) & (1,k\oplus 1) & (5,k\oplus 5) \\ (2,k\oplus 2) & (6,k\oplus 6) & (3,k\oplus 3) & (7,k\oplus 7) \\ (8,k\oplus 8) & (12,k\oplus 12) & (9,k\oplus 9) & (13,k\oplus 13) \\ (10,k\oplus 10) & (14,k\oplus 14) & (11,k\oplus 11) & (15,k\oplus 15). \end{array}$$

Eventually, each symmetry that contains the relation $(0,k)$—i.e., $x[0] = x[k]$—also has the relations $(i,k\oplus i)$, $1,\ldots,15$. Therefore, we have 15 distinct wordwise symmetry classes, of the form

$$(i,k\oplus i), i = 0,\ldots,15$$

for $k \in \{1,\ldots,15\}$. Each class contains 2^{512} states. For example, the case $k = 1$ provides directly C_1, and more generally $k = i$ corresponds to C_i.

4.3 Exploiting Symmetric States for Finding Preimages

Given a target digest, one can make a preimage attack similar to that in §2, and exploit symmetric states for the connection. The attack goes as follows:

- from the initial state, reach a symmetric state (of any class) by using $2^{1024-516-8} = 2^{500}$ message blocks
- from a state before finalization, reach (backwards) another symmetric state (not necessarily of the same class)
- from these two symmetric states in classes C_i and C_j, use null message blocks in both directions to reach two states in $C_i \cap C_j$
- find a collision by trying $\sqrt{|C_i \cap C_j|}$ messages in each direction

Complexity of steps 1 and 2 is about 2^{501} computations of T. The cost of steps 3 and 4 depends on i and j; but it is upper bounded by 2×2^{256} operations.

Thus, in any case, the total complexity is about 2^{501} calls to T. This attack, however, finds messages of unauthorized size (more than 2^{256} bytes!).

One can find preimages of reasonable size by using a variant of the above attack: suppose $b > 4$, from the initial state reach a state in a given class C_i, do the same backwards from a state before finalization. For a given b, the complexity of reaching a symmetric state depends on the C_i considered. Then one seeks a collision within C_i by trying messages preserving the symmetry: for example, if $b = 5$ and $C_i = C_1$, then one has to preserve the equality $x[0] = x[1]$ and shall thus pick 5-byte messages of the form `X000X` (each digit stands for a byte). Since any C_i contains 2^{512} states, the cost of finding a collision within C_i is about 2^{256} trials in each direction.

Below we give a class example C_i that is the easiest to reach, depending on the value of b:

- $5 \leq b < 9$: one of the best classes is C_1, which gives $(1024-2\times 4\times 8)/2 = 480$ equations to verify
- $9 \leq b < 17$: one of the best classes is C_2, which give $(1024 - 2 \times 8 \times 8)/2 = 448$ equations to verify

- $17 \leq b < 33$: one of the best classes is C_4, which gives $(1024-2\times16\times8)/2 = 384$ equations to verify
- $33 \leq b < 65$: one of the best classes is C_8, which gives $(1024-2\times32\times8)/2 = 256$ equations to verify

If n equations have to be verified, the cost of reaching a symmetric state is about 2^n evaluations of T. Compared to the preimage attack in §2, the best speed-up obtained from a given C_i is when $b = 4d + 1$, where d is the number of 32-bit words that separate the first repetition of two words.
To illustrate this attack, let's study in more detail the case of C_1:

- if $b \equiv 0 \bmod 8$, there are $(1024 - 8b)/2 = 512 - 4b$ equations to satisfy, thus about 2^{512-4b} calls to T are necessary
- if $b \equiv 4 \bmod 8$, there are only $(1024 - 8b - 32)/2 = 496 - 4b$ equations to satisfy, because one has no condition on the first state word not xored with the message block
- generalizing, when $b \bmod 8 \leq 4$, about $2^{512-4(b+(b \bmod 4))}$ calls to T are necessary
- when $b \bmod 8 > 4$, there are $(1024 - 8b - 32 + 8(b \bmod 4))/2$ equations to satisfy, which gives a cost $2^{496-4(b-(b \bmod 4))}$

The general formula for the number of equations is

$$512 - 32\lfloor b/8 \rfloor - 32\lfloor (b \bmod 8)/4 \rfloor - [(\lfloor (b \bmod 8)/4 \rfloor + 1) \bmod 2] \times 8(b \bmod 4) \ .$$

In the best case ($b \equiv 4 \bmod 8$), the attack is 2^{15} times faster than that in §2 (in the worst case, $b \equiv 0 \bmod 8$, it has the same complexity). Note that when $b = 5$, the attack makes about 2^{481} calls to T, against 2^{493} with the attack in §2.

4.4 Exploiting Symmetric States for Finding Collisions

We present a technique to find collisions for a weakened version of CubeHash, in which we modify the IV (initial state). The initialization of CubeHash never leads to a symmetric initial state. Here we present a practical collision attack that would apply if the initial state were symmetric, and in $C_1 \cap C_2 \cap C_4 \cap C_8$.

Suppose that the initial state of CubeHashr/b-h is in $C_1 \cap C_8$, i.e. is of the form

```
AAAAAAAA AAAAAAAAA BBBBBBBBB BBBBBBBB .
```

If one hashes the $b2^{33}$-byte message that contain only zeros, then each of the 2^{33} intermediate states is an element of $C_1 \cap C_2 \cap C_4 \cap C_8$. Assuming that T acts like a random permutation over this set, one will find two identical states with probability about 0.63, which directly gives a collision.

5 On the Fixed Points of T

In this section we let T be the 1-round transform of CubeHash. A fixed point for T^r, $r > 0$, is a state x that is left unchanged by T^r, i.e., $T^r(x) = x$. Recall that the average number of cycles of length k is $1/k$ for a random permutation. If T were a random permutation, T would thus have one fixed point. Noting that a cycle of length r gives r fixed points for T^r, we have that T^2 would have two fixed points (the one of T and one due to an average of $2 \times 1/2$ fixed points from cycles of length two); T^4 would have three fixed points (one for each cycle length in $1, 2, 4$), etc. More generally, the average number of fixed points for T^n would be the number of divisors of n, if T were a random permutation.

Note that each symmetry class represents a class of cycles of T, and that the 15 symmetry classes give 67 distinct subsets. Modeling T as a random permutation over each of those subsets, one expects 67 fixed points. This gives for T^8 $1+4\times 67 = 269$ fixed points, where 4 is the number of divisors of 8, i.e., of cycles length that give fixed points for T^8. Note that this results assumes a random behavior of T with respect to fixed points over the 67 subsets considered.

Finding examples of fixed points seems difficult, however: the zero state $x[0] = \cdots = x[31] = 0$ is a trivial fixed point for T, and thus also for T^n, $n \geq 0$. Among the states of the form $x[0] = \cdots = x[15]$, $x[16] = \cdots = x[31]$, the only nontrivial fixed point is the state with $x[0] =$ `54E5FC8A` and $x[8] =$ `84FE49D2`.

6 Truncated Differentials over T

This section shows how to detect non-randomness over the 10-round T transform. We start from a weight-64 difference to reach a weight-1 difference after 3 rounds with high probability; this *nonlinear* differential was discovered by simply computing backwards from the weight-1 difference.

We consider the input difference `80000000` in $x[16]$. The word $x[16]$ was chosen because $x[16] \cdots x[31]$ diffuse less in the first rounds than $x[0] \cdots x[15]$. We set a difference `80000000` to minimize the impact of carries.

We consider the following nonlinear differential. Input difference (weight-64):

```
18000000 10000000 08000000 30000000
00000040 00000080 00000000 00000000
00400000 00000000 00400000 01000404
00000003 80802002 00000001 81802004
40000000 08000000 00000000 E8020600
00000000 00000100 00000080 41F001C0
00400008 00000008 00400000 01000404
00000005 80802002 00000001 8080200C
```

Difference after one round (weight-26):

000E0000 00000000 00000000 00000000
00000000 00000040 00000080 00000040
01000004 00000000 00000004 00000000
00000000 00000000 00002000 00000000
800E0200 00000000 00000000 00000000
00000000 000000C0 00000080 000001C0
00000000 00000004 00000000 00000004
00000000 00002000 0000C000 00000000

Difference after two rounds (weight-9):

00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 01000000
00000000 00002000 00000000 00002000
00000000 00000000 00000000 80000000
00000000 00000000 00000000 00100000
00000000 00000000 00000000 03000000
00000000 00002000 00000000 00002000

Difference after three rounds (weight-1):

00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
80000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

which after another round gives with probability 1 the difference

80000000 00000000 80000000 00000000
00000400 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 80000000 00000000 80000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000

This differential holds with negligible probability for a random input. But it holds for the input

```
DFB7AA11 7B2872F1 2848B142 64CB0AF9
17DA36E7 320A7AB2 27621CD8 B6E23031
3BCE90DB 0E496C61 AF4156BD 0B4D857F
4379D4C0 D495EAC9 038BD6E5 72A114CC
29065395 824774C3 F0923C34 28F3B2DD
74251DF6 1A562265 BD8EE5E3 DEFDD839
2804D3BE 89417DC3 F001CE4A 6A5328A8
2BEC024E B2306F17 1F2A7C6C 14BC37B6
```

For 32 random bits in $x[25]$ and $x[26]$ (at positions $4, \ldots, 19$ in both), the differential is satisfied with probability approximately 0.985.

Note that in the linear model (i.e. when additions are replaced by xors), a differential path starting from the weight-1 difference cycles over 47 rounds. That is, it comes back to the difference `80000000` in $x[16]$ after 47 rounds.

Based on the above differential, we empirically looked for high-probability truncated differentials, based on the weight-64 input difference, and applying to each output bit a frequency test similar to that in [11, §2.1], with decision threshold 0.001 and 2^{20} samples. We found 4 output bits with p-value less than 0.001, at positions 579, 778, 841, and 842. Over 11 rounds and more, no bias was detected.

This observation is consistent with the fact that, when starting from the weight-1 difference, we could detect non-randomness on up to 7 rounds (now this difference is introduced three rounds later). Note that in a previous version of this article [2], we reached 8 rounds by starting one round before the weight-1 difference.

These observations indicate that 10-round T does not act as a random permutation, and that 10 rounds may not be overkill, as suggested in [4]. But note that the settings used don't correspond to a realistic attack scenario. Furthermore, if we restrict ourselves to differences in the first state byte, and put random bits in the rest of the state, then we observe non-randomness after up to 5 rounds.

References

1. Aumasson, J.-P.: Collision for CubeHash2/120-512. NIST mailing list (December 4, 2008), http://ehash.iaik.tugraz.at/uploads/a/a9/Cubehash.txt
2. Aumasson, J.-P., Meier, W., Naya-Plasencia, M., Peyrin, T.: Inside the hypercube. Cryptology ePrint Archive, Report 2008/486, version 20081124:132635 (2008)
3. Bernstein, D.J.: CubeHash appendix: complexity of generic attacks. Submission to NIST (2008)
4. Bernstein, D.J.: CubeHash attack analysis (2.B.5). Submission to NIST (2008)
5. Daniel, J.B.: CubeHash specification (2.B.1). Submission to NIST (2008)

6. Brier, E., Khazaei, S., Meier, W., Peyrin, T.: Attack for CubeHash-2/2 and collision for CubeHash-3/64. NIST mailing list (local link) (2009), http://ehash.iaik.tugraz.at/uploads/3/3a/Peyrin_ch22_ch364.txt
7. Brier, E., Peyrin, T.: Cryptanalysis of CubeHash (2009), http://thomas.peyrin.googlepages.com/BrierPeyrinCubehash.pdf
8. Dai, W.: Collisions for CubeHash1/45 and CubeHash2/89 (2008), http://www.cryptopp.com/sha3/cubehash.pdf
9. Diaconis, P., Mosteller, F.: Methods for studying coincidences. Journal of the American Statistical Association 84(408), 853–861 (1989)
10. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
11. NIST. SP 800-22, a statistical test suite for random and pseudorandom number generators for cryptographic applications (2001)
12. Suzuki, K., Tonien, D., Kurosawa, K., Toyota, K.: Birthday paradox for multicollisions. In: Rhee, M.S., Lee, B. (eds.) ICISC 2006. LNCS, vol. 4296, pp. 29–40. Springer, Heidelberg (2006)

Meet-in-the-Middle Preimage Attacks on Double-Branch Hash Functions: Application to RIPEMD and Others

Yu Sasaki and Kazumaro Aoki

NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo, 180-8585 Japan
sasaki.yu@lab.ntt.co.jp

Abstract. We describe preimage attacks on several double-branch hash functions. We first present meet-in-the-middle preimage attacks on RIPEMD, whose output length is 128 bits and internal state size is 256 bits. With this internal state size, a straightforward application of the meet-in-the-middle attack will cost the complexity of at least 2^{128}, which gives no advantage compared to the brute force attack. We show two attacks on RIPEMD. The first attack finds pseudo-preimages and preimages of the first 33 steps with complexities of 2^{121} and $2^{125.5}$, respectively. The second attack finds pseudo-preimages and preimages of the intermediate 35 steps with complexities of 2^{96} and 2^{113}, respectively. We next present meet-in-the-middle preimage attacks on full Extended MD4, reduced RIPEMD-256, and reduced RIPEMD-320. The best known attack for these is the brute force attack. We show how to find preimages more efficiently on these hash functions.

Keywords: RIPEMD, double branch, preimage, meet-in-the-middle.

1 Introduction

Hash functions are cryptographic primitives used for various purposes. They are required to satisfy several security properties: preimage resistance, second preimage resistance, collision resistance, and so on. Usually, if the length of the hash is n-bit, the required security for these properties is $2^n, 2^n$, and $2^{n/2}$, respectively. Note that in the SHA-3 competition [22] conducted by NIST, 2^n security is required for the preimage resistance.

Various hash functions have been designed. A list of hash function types is shown in Fig. 1. The most widely used hash functions, e.g., MD5 [18], SHA-1, and SHA-2 [23], have a structure where the initial value, whose length is the same as the hash value, is iteratively updated by using messages. Hereafter, we call such a structure "single-path." In contrast, some hash functions update two copies of the initial value in parallel, merge each result, and finally output the merged value as the hash value of the message. Hereafter, we call such a structure "double-branch." For example, RIPEMD [16], RIPEMD-128, and RIPEMD-160 [7], are double-branch hash functions.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 214–231, 2009.

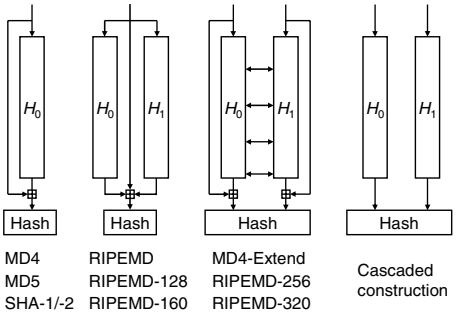


Fig. 1. Types of hash function structures

Hash functions are sometimes required to output longer hash values. For this purpose, some hash functions define an extension to output the double-length hash value, e.g., MD4 [17], RIPEMD-128, and RIPEMD-160. For a given input message, two hash values are computed by using different initial values and round constants. Finally, the concatenated value of two hash values, which is the double-length of the original hash value, is output. Such extensions of MD4, RIPEMD-128, and RIPEMD-160 are called Extended MD4[1], RIPEMD-256, and RIPEMD-320, respectively. Efforts to strengthen the security are made in these extensions. Intermediate chaining values after each round are swapped between computations of two hash values so that a stronger interaction between two computations can be achieved.

A cascaded construction, which was analyzed by Joux [10], produces a long hash value from two short hash values. It computes two hash functions and outputs the concatenated value.

The security of the double-branch hash functions is unclear. Intuitively, if two hash functions are ideal, the security will be a product of two hash functions. However, if two hash computations are similar, some attacks might be performed because of unwanted dependencies. The designers of RIPEMD-256 and -320 consider this situation. Although the known best preimage attack on RIPEMD-256 and -320 is the brute force attack, which costs 2^{256} and 2^{320}, their security is claimed to be 2^{128} and 2^{160}, respectively. Similarly, the security of Extended MD4 is not described by its designer[2]. Reference [14, Fact 9.27] claims the security of the cascaded construction is a product of each hash function; however, Joux showed the security is damaged if iterated hash functions are instantiated [10].

1.1 Attack History

Several papers have been published on finding collisions or variants of collisions on RIPEMD, RIPEMD-128, and RIPEMD-160 [6,5,4,25,12]. However, the

[1] Rivest, the designer of MD4, did not name this extension. Dobbertin, the first cryptanalyst of this extension, called it "Extended MD4."

[2] Dobbertin, in his analysis paper [5], introduced Extended MD4, which was proposed for highest security requirements.

preimage resistance of double-branch hash functions has not been studied much. Since 2008, several meet-in-the-middle preimage attacks on hash functions whose structures are similar to MD4 have been proposed [11,2,1,20,21]. The targets of these attacks are single-path hash functions, hence the attack techniques cannot be applied to double-branch hash functions directly. Mendel presented preimage attacks [13] on HAS-V [15], which is a double-branch hash function with a swapping function. The attack exploits a weakness of the HAS-V step-function, which cannot be applied to other hash functions. Saarinen [19] presented a preimage attack on FORK-256 [8], which is a 4-branch hash function. The attack has some similarity with ours; however, its success lies in the small number of steps in each branch and the weak message schedule of FORK-256. At ISPEC 2009, a preimage attack on 29 steps of RIPEMD with a complexity of $2^{115.2}$ was presented by Wang et al. [24]. Note that our work is independent of Ref. [24].

At CRYPTO'04, Joux analyzed the cascaded construction [10]. He showed that the cascaded construction does not provide the security of each product if an iterated hash function is used. Joux also explained that his technique cannot be applied to RIPEMD-256 and -320 due to the swapping function of intermediate values. This shows that the swapping function strengthens the security at some point. However, whether or not it can prevent other attacks is unclear.

1.2 Our Contribution

We present preimage attacks on several double-branch hash functions. We first present preimage attacks on step-reduced RIPEMD and then show how to find preimages of Extended MD4, RIPEMD-256, and RIPEMD-320 faster than the brute force attack does. The second result shows that using the swapping function does not provide the ideal security for double-branch hash functions. Details of each result are as follows.

1. We describe two preimage attacks on RIPEMD. The first attack, with a complexity of 2^{121}, provides pseudo-preimages of the *first* 33 out of 48 steps of RIPEMD. This can be converted to a preimage attack with a complexity of $2^{125.5}$. Our attack is based on the meet-in-the-middle attack on MD5 and MD4 [1]. However, because RIPEMD runs two MD4 computations, the size of the internal state is also doubled. Therefore, the straightforward application of the meet-in-the-middle attack does not give any advantage. We focus on the differentials of two MD4 computations, and efficiently perform the meet-in-the-middle attack.

   The second approach, with a complexity of 2^{96}, provides pseudo-preimages of the *intermediate* 35 steps of RIPEMD. This can be converted to a preimage attack with a complexity of 2^{113}. Technically, we use the meet-in-the-middle attack and the idea of local collision. This strategy is partially similar to the preimage attack on 1-block MD4 [1].
2. Extended MD4 provides 256-bit hash values. Our preimage attack on full Extended MD4 finds pseudo-preimages and preimages with complexities of 2^{229} and $2^{243.5}$, respectively. We also show that pseudo-preimages and preimages

of the *first* 62 out of 64 steps of RIPEMD-256 are found with complexities of 2^{240} and 2^{249}, respectively, and pseudo-preimages and preimages of the *intermediate* 64 out of 80 steps of RIPEMD-320 are found with complexities of 2^{304} and 2^{313}, respectively.

From a technical viewpoint, we show how to avoid the swapping functions. In Extended MD4, the message schedules of both sides are identical. We show that using swapping functions in such a structure does not prevent our attack. In RIPEMD-256 and -320, message schedules are different on each side. However, the attacker might be able to attack even if the swapping function is used. Then, by combining this idea with the splice-and-cut, partial-matching, and partial-fixing techniques proposed in Ref. [1,21], we attack those hash functions.

Although results in this paper do not contradict the security claims of these hash functions, the known best attack on these hash functions is the brute force attack, and no one knows whether or not better attacks exist. Therefore, we believe that our analyses contribute to better understanding of the security of double-branch hash functions.

2 Description of Hash Functions

2.1 MD4

MD4 takes arbitrary length messages as input and outputs 128-bit hash values. MD4 was proposed in 1990 by Rivest [17] and is a basic component of RIPEMD and Extended MD4. MD4 has the Merkle-Damgård structure. The input message is padded to be multiples of 512-bit. First, a single bit '1' is appended, then bit '0's are appended until the message length becomes 448 mod 512. Finally, the binary expression of the input message length is appended to the last 64 bits. The message is divided into 512-bit message blocks M_i. Then, the hash value is computed as follows:

$$\begin{cases} H_0 \leftarrow IV, \\ H_{i+1} \leftarrow \mathrm{md4}(H_i, M_i) & \text{for } i = 0, 1, \ldots, n-1, \end{cases}$$

where IV is the initial value defined in the specification, H_n is the output hash value, and md4: $\{0,1\}^{128} \times \{0,1\}^{512} \rightarrow \{0,1\}^{128}$ is the compression function of MD4 computed as follows.

1. M_i is divided into 32-bit message words m_j $(j = 0, 1, \ldots, 15)$.
2. p_0 is set to H_i.
3. Compute the following: $p_{j+1} \leftarrow R_j(p_j, m_{\pi(j)})$ for $j = 0, 1, \ldots, 47$.
4. H_{i+1} $(= p_{48} + H_i)$ is output, where "+" denotes 32-bit word-wise addition. In this paper, we similarly use "−" to denote 32-bit word-wise subtraction.

R_j is the step function for Step j. Let a_j, b_j, c_j, and d_j be 32-bit values that satisfy $p_j = (a_j, b_j, c_j, d_j)$. $R_j(p_j, m_{\pi(j)})$ is defined as follows:

$$\begin{aligned} a_{j+1} &= d_j, & b_{j+1} &= (a_j + \Phi_j(b_j, c_j, d_j) + m_{\pi(j)} + k_j) \lll s_j, \\ c_{j+1} &= b_j, & d_{j+1} &= c_j, \end{aligned}$$

where Φ_j, k_j, and $\lll s_j$ are the bitwise Boolean function, constant value, and left rotation defined in the specification, respectively. $\pi(j)$ is the MD4 message schedule. Note that $R_j^{-1}(p_{j+1}, m_{\pi(j)})$ can be computed at almost the same complexity as that of R_j.

2.2 RIPEMD

RIPEMD [16] is an extension of MD4, whose compression function consists of two parallel copies of MD4's compression function. These functions are identical but for the constant number in each step. We describe chaining variables for one side $p_j = (a_j, b_j, c_j, d_j)$ and for the other side $p'_j = (a'_j, b'_j, c'_j, d'_j)$. The message schedule $\pi(j)$, the constant numbers k_j and k'_j, and the rotation number s_j are different from those for MD4. These values are shown in Table 1. Finally, the out-

Table 1. Message schedule, constants, and rotation numbers of RIPEMD

$\pi(0), \pi(1), \ldots, \pi(15)$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(16), \pi(17), \ldots, \pi(31)$	7	4	13	1	10	6	15	3	12	0	9	5	14	2	11	8
$\pi(32), \pi(33), \ldots, \pi(47)$	3	10	2	4	9	15	8	1	14	7	0	6	11	13	5	12

	$0 \le i \le 15$	$16 \le i \le 31$	$32 \le i \le 47$
k_i	0x00000000	0x5a827999	0x6ed9eba1
k'_i	0x50a28be6	0x00000000	0x5c4dd124

$s_0, s_1, \ldots, s_{15}$	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
$s_{16}, s_{17}, \ldots, s_{31}$	7	6	8	13	11	9	7	15	7	12	15	9	7	11	13	12
$s_{32}, s_{33}, \ldots, s_{47}$	11	13	14	7	14	9	13	15	6	8	13	6	12	5	7	5

put of RIPEMD's compression function $H_{n+1} = (H_a, H_b, H_c, H_d)$ is computed by using $H_n = (IV_a, IV_b, IV_c, IV_d)$, p_{48}, and p'_{48} as follows.

$$H_a = IV_b + c_{48} + d'_{48}, \qquad H_b = IV_c + d_{48} + a'_{48},$$
$$H_c = IV_d + a_{48} + b'_{48}, \qquad H_d = IV_a + b_{48} + c'_{48}.$$

2.3 RIPEMD-128, RIPEMD-160

RIPEMD-128 and RIPEMD-160, which output 128-bit and 160-bit hash values respectively, were proposed by Dobbertin et al. in 1996 [7]. They have been standardized by the International Organization for Standardization (ISO) [9].

A branch of RIPEMD-160 uses five 32-bit chaining variables. It computes 80 steps to produce the output value. Let the chaining variables in step j be $p_j = (a_j, b_j, c_j, d_j, e_j)$. Step function $R_j(p_j, m_\pi(j))$ is as follows.

$$a_{j+1} = e_j, \quad c_{j+1} = b_j, \quad d_{j+1} = c_j \lll 10, \quad e_{j+1} = d_j,$$
$$b_{j+1} = ((a_j + \Phi_j(b_j, c_j, d_j) + m_{\pi(j)} + k_j) \lll s_j) + e_j.$$

Table 2. Message schedules of RIPEMD-160

r	$\pi(r), \pi(r+1), \ldots, \pi(r+15)$	$\pi'(r), \pi'(r+1), \ldots, \pi'(r+15)$
0	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	5 14 7 0 9 2 11 4 13 6 15 8 1 10 3 12
16	7 4 13 1 10 6 15 3 12 0 9 5 2 14 11 8	6 11 3 7 0 13 5 10 14 15 8 12 4 9 1 2
32	3 10 14 4 9 15 8 1 2 7 0 6 13 11 5 12	15 5 1 3 7 14 6 9 11 8 12 2 10 0 4 13
48	1 9 11 10 0 8 12 4 13 3 7 15 14 5 6 2	8 6 4 1 3 11 15 0 5 12 2 13 9 7 10 14
64	4 0 5 9 7 12 2 10 14 1 3 8 11 6 15 13	12 15 10 4 1 5 8 7 6 2 13 14 0 3 9 11

Between two copies of the compression function, the order of the Boolean functions, message schedules, constants, and rotation numbers are different. The message schedules are shown in Table 2. The output of RIPEMD-160 is computed in the same manner as that of RIPEMD.

A branch of RIPEMD-128 uses 4 chaining variables and consists of 64 steps. The step function is the same as that of MD4 and RIPEMD. The Boolean functions and rotation numbers used in RIPEMD-128 are the same as those of the first 64 steps for RIPEMD-160, but the order is different. The message schedule for RIPEMD-128 is the same as that of the first 64 steps for RIPEMD-160, which is shown in Table 2. Computation for the final output is also the same as that of RIPEMD.

2.4 Extended MD4, RIPEMD-256, and RIPEMD-320

Extended MD4 is an optional extension of MD4 proposed by Rivest to obtain 256-bit hash values [17]. Two copies of MD4 are run in parallel over the input. The first copy is the same as MD4. The second copy is computed with different IV and constants. To strengthen the data dependency between two copies, a swapping function is introduced, which exchanges the values of a_{16} and a'_{16}, a_{32} and a'_{32}, and a_{48} and a'_{48}. The final output is obtained by concatenating both results.

RIPEMD-256 and RIPEMD-320 are extensions of RIPEMD-128 and RIPEMD-160 for obtaining the double length hash values without needing a higher security level [7]. The output is achieved by computing the feedforward of IV in each branch and concatenating the results at the end of every application of the compression function. Interaction between two copies is introduced by a swapping function, which exchanges the values of a_{16} and a'_{16}, b_{32} and b'_{32}, etc.

3 Related Works

3.1 Converting Pseudo-preimage Attack to Preimage Attack

Given a hash value H_n, a pseudo-preimage is a pair of (H_{n-1}, M) such that $\mathrm{Hash}(H_{n-1}, M) = H_n$. In x-bit iterated hash functions, a pseudo-preimage attack whose complexity is $2^y, y < x - 2$ can be converted to a preimage attack with a complexity of $2^{\frac{x+y}{2}+1}$ [14, Fact9.99].

3.2 Meet-in-the-Middle Preimage Attack

Aoki and Sasaki proposed a preimage attack based on the meet-in-the-middle attack [1]. They proposed three techniques named *splice-and-cut*, *partial-matching*, and *partial-fixing*.

The splice-and-cut technique considers the first and last steps of the compression function as consecutive steps. Then, the compression function is divided into two *chunks* of steps so that each chunk includes independent message words, which are called *neutral words*. Then, a pseudo-preimage is computed by the meet-in-the-middle attack.

The partial-matching technique can skip messages in several steps when checking the match in the meet-in-the-middle attack. It focuses on the property where not all the chaining variables are updated in each step. With this idea, we can partially compare two results, even if values of message words in several steps are not known.

The partial-fixing technique enables an attacker to skip more steps. The idea is to fix a part of the neutral words so that an attacker can partially compute a chunk even if a neutral word for the other chunk appears. For example, consider the inversion of MD4: $a_j = (b_{j+1} \ggg s_j) - \Phi_j(c_{j+1}, d_{j+1}, a_{j+1}) - m_{\pi(j)} - k_j$. If the lower n bits of $m_{\pi(j)}$ are fixed and thus known to the attacker and if other variables are fully known, the lower n bits of a_j can be computed independently of the higher $32 - n$ bits.

Since the internal state size of RIPEMD is double the output size, the straightforward application of the meet-in-the-middle attack does not give any advantage.

3.3 Analysis on Double-Branch Hashes and Cascaded Construction

At CRYPTO 2004, Joux proposed attacks on cascaded construction [10]. Joux showed how to generate multi-collisions of iterated hash functions and how to find collisions and preimages of the cascaded construction by using multi-collisions. The success of Joux's attack lies in the independency of the two hash functions in the cascaded construction, namely, multi-collisions of the iterated hash functions can be generated independently of the others. Joux explained that his technique would not be applied to RIPEMD-256 and -320 due to the dependency of the two compression functions caused by swapping functions. In conclusion, if swapping functions are used, no attack is known to break the preimage resistance of double-branch hash functions.

4 Preimage Attacks on RIPEMD

We present here two preimage attacks on RIPEMD. The first attack targets the first 33 steps and the second attack targets the intermediate 35 steps.

4.1 Attacks on First 33 Steps

Outline of Attack. Our attack is based on the meet-in-the-middle attack introduced in Section 3.2. However, since the internal state size is double the

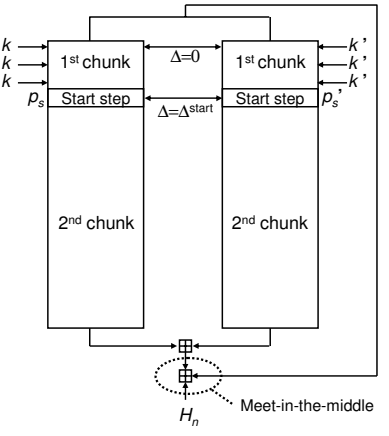


Fig. 2. Outline of strategy 1

output size, a direct application of the meet-in-the-middle attack cannot exceed the brute force attack. Our strategy to solve this problem is shown in Fig. 2. We separate the attack target so that one chunk is located in the first several steps and the other chunk is located in the last relatively long steps, and then we compare the results of the two chunks at the last feedforward operation which is performed in 128 bits.

Assume m_a and m_b are neutral words, where m_a is included in the first chunk but is not included in the second chunk, and m_b is vice versa. Remember that message schedules for both sides are identical, hence if one side can be separated into two chunks, the other side can always be separated in the same manner. First, we fix all message words but m_a and m_b and fix chaining variables at the border of two chunks, e.g., fix p_s and p'_s to compute the first and second chunks independently. We then inversely compute the first chunk with $R_j^{-1}(p_{j+1}, m_j)$ and ${R'_j}^{-1}(p'_{j+1}, m_j)$ for $j = s-1, s-2, \ldots, 0$ by trying all values of m_a and store the results in a table. Finally, we compute the second chunk with $R_j(p_j, m_j)$ and $R'_j(p'_j, m_j)$ for $j = s, s+1, \ldots, 32$ by trying all values of m_b and then check whether the result matches items in the table.

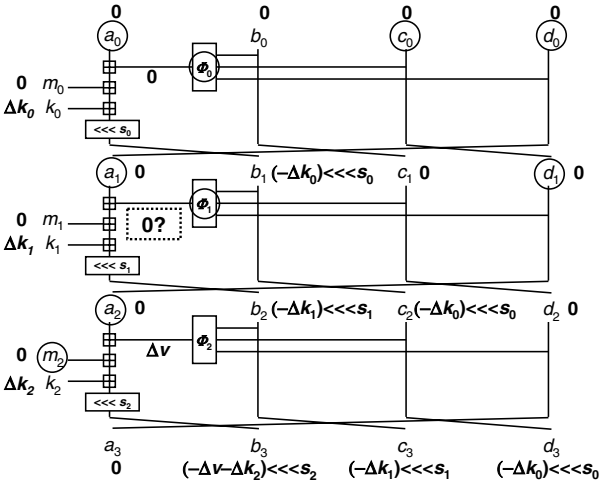
For consistency with the specification of RIPEMD, the values of p_0 and p'_0 computed in the backward computation must be identical, because they are originally computed from the same IV. The differences of the computations for both sides are differences of the constant Δk only. Therefore, we fix intermediate chaining variables p_s and p'_s to have a specific difference Δ^{start} so that Δ^{start} and Δk can be cancelled out in the backward computation.

Set Up of Attack. We separate the 33 steps into 2 chunks as shown in Fig. 3. The border of the two chunks is between Steps 2 and 3; we therefore fix p_3 and p'_3 so that their difference Δ^{start} can cancel Δk in Steps 0–2. Now we trace the differentials in Steps 0–2 of both sides to determine the appropriate Δ^{start}. The analysis is shown in Fig. 4.

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	0	1	**2**	3	4	5	6	7	8	9	10	11	**12**	13	14	15
	first chunk			second chunk												
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	7	4	13	1	10	6	15	3	**12**	0	9	5	14	**2**	11	8
	second chunk													skip		
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	3	10	**2**	4	9	15	8	1	14	7	0	6	11	13	5	**12**
	skip	Excluded from the attack target														

In RIPEMD, the message schedules of the two compression functions are identical. We separate them into two chunks in the same manner.

Fig. 3. Chunks for first 33 steps of RIPEMD



Bold font represents the value of differences.

Fig. 4. Differences propagation in first three steps

The difference of a variable X is defined as $\Delta X = X' - X$. The goal is determining $\Delta a_3, \Delta b_3, \Delta c_3$, and Δd_3 so that $\Delta a_0, \Delta b_0, \Delta c_0$, and Δd_0 become 0. Since message schedules for both sides are identical, $\Delta m_{\pi(j)}$ is always 0. In the first chunk, m_2 is the neutral word. Therefore, in every trial of the first chunk, the values of m_2 and corresponding chaining variables are changed. In Fig. 4, we circled such variables. The analysis is as follows.

$\Delta b_0 = 0$**:** This can be easily achieved by setting $\Delta a_3 = 0$.

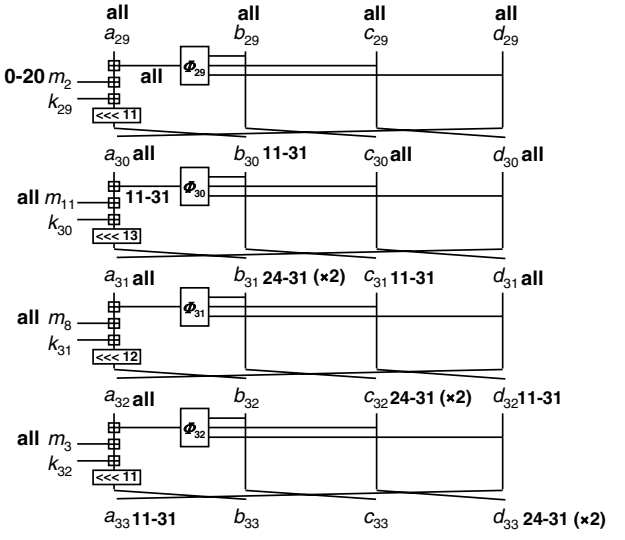
$\Delta a_0 = 0$**:** Assume we can achieve $\Delta b_0 = \Delta c_0 = \Delta d_0 = 0$. Then, $\Delta a_0 = 0$ can be achieved by setting $\Delta d_3 = (-\Delta k_0) \lll s_0$.

$\Delta c_0 = 0$**:** Assume we have finished fixing the values of $a_3, a'_3, c_3, c'_3, d_3$, and d'_3. Then, the value and difference of the output of Φ_2 are fixed. Let this difference be Δv. Finally, $\Delta c_0 = 0$ is achieved by setting $\Delta b_3 = (-\Delta v - \Delta k_2) \lll s_2$.

$\Delta d_0 = 0$**:** Achieving $\Delta d_0 = 0$ is complicated. We want to guarantee $\Delta a_1 = 0$ regardless of the value of the neutral variable m_2. To achieve this, we need to fix the value of Φ_1 independently of m_2. Since the function Φ_1 is $\Phi_1(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$ and $\Delta c_1 = \Delta d_1 = 0$, $\Delta\Phi_1$ can be fixed to 0 by setting $b_1 = b'_1$. However, this is impossible since b_1 and b'_1 must have differences. Consequently, we search for the exact value of b_1 and b'_1 to minimize the Hamming weight of $(b_1 \oplus b'_1)$ so that the probability of $\Delta\Phi_1 = 0$ is maximized. Remember that for each bit where $b_1 \oplus b'_1 = 1$, the equation $\Delta\Phi_1 = 0$ is satisfied with a probability of $1/2$.

As we will explain later, we fix the lower 21 bits of m_2 for the partial-fixing technique. Consequently, the lower 21 bits of $\Delta\Phi_1$ are fixed. Therefore, minimizing the Hamming weight of the upper 11 bits (HW^{11}) of $(b_1 \oplus b'_1)$ is enough. We tried 2^{32} values of b_1 and confirmed that no value achieves $HW^{11}(b_1 \oplus b'_1) \leq 3$ and many values achieve $HW^{11}(b_1 \oplus b'_1) = 4$. For example, $b_1 =$ `0xffffffff`, $b'_1 =$ `0x50a28be5` is the case. Finally, by choosing the value of $b_1 (= d_3)$ to minimize the Hamming weight, the probability of $\Delta\Phi_1 = 0$ is 2^{-4}.

Partial-Matching and Partial-Fixing Techniques. As a result of computing the second chunk, we obtain p_{29} and p'_{29}. By fixing the lower 21 bits of m_2, we can perform the meet-in-the-middle attack on a further 4 steps in forward computation, namely, up to Step 32. The equation we use for matching is $H_b = IV_c + d_{33} + a'_{33}$, where H_b is given and the upper 11 bits of IV_c are



Bold font represents the bit positions of known bits.

Fig. 5. Partial-matching and partial-fixing techniques

produced from the first chunk. Therefore, we need to compute d_{33} from p_{29} and a'_{33} from p'_{29}. How we compute d_{33} and a'_{33} is shown in Fig. 5. Since m_2 is identical on both sides, the fixed bit positions in m_2 are also identical on both sides.

In Fig. 5, since the lower 21 bits of m_2 are fixed, we can uniquely compute bits 11–31 of b_{30}. This produces bits 11–31 of Φ_{30}. In the addition of Φ_{30}, we cannot determine the carried number from bit position 10 to 11. Therefore, we consider both carried number patterns and obtain two candidates for bits 24–31 of b_{31} after the $s_{30}(= 13)$-bit left rotation. Then, we compute the upper 8 bits of $d_{33} + a'_{33}$ with consideration of the two candidates of carried number patterns from bit position 23 to 24. In conclusion, the second chunk produces 4 candidates for the upper 8 bits of $d_{33} + a'_{33}$ for given p_{29} and p'_{29}, and thus, we can perform the 8-bit matching in 33-step RIPEMD.

Remark: From a designer's view, when results of two compression functions are merged, adding two values from different registers seems to be a good strategy against our attack. In fact, with only the partial-matching technique, attackers can skip only two steps in RIPEMD, whereas attackers can skip three steps in MD4. This is because the attacker needs to know the values of two different registers to compute each word of the output of RIPEMD.

Attack Procedure. Our attack first finds pseudo-preimages and converts them to preimages. Therefore, our attack finds a 2-block preimage. Hence, we fix m_{13}, m_{14}, and m_{15} to satisfy padding for 2-block messages. Given a hash value $H_2 = (H_a, H_b, H_c, H_d)$, the attack procedure is as follows.

1. Fix m_i $(i \notin \{2, 12, 13, 14, 15\})$ and the lower 21 bits of m_2 to randomly chosen values.
2. Fix a_3, b_3, c_3 to randomly chosen values and d_3 to `0xffffffff`. Then, compute a'_3, b'_3, c'_3, and d'_3 to make Δ^{start}, shown in Fig. 4.
3. (a) For all upper 11 bits of m_2, compute $R_j^{-1}(p_{j+1}, m_{\pi(j)})$ and $R_j'^{-1}(p'_{j+1}, m_{\pi(j)})$ for $j = 2, 1, 0$.
   (b) If $\Delta p_0 = 0$, compute $H_b - c_0$ and store $(m_2, p_0, H_b - c_0)$s in a table.
4. Compute $R_j(p_j, m_{\pi(j)})$ and $R'_j(p_{j'}, m_{\pi(j)})$ for $j = 3, 4, \ldots, 11$, and store p_{12} and p'_{12}.
5. (a) For all m_{12}, compute $R_j(p_j, m_{\pi(j)})$ and $R'_j(p'_j, m_{\pi(j)})$ for $j = 12, 13, \ldots 28$.
   (b) Compute bit positions 11 to 31 of b_{30} and b'_{30}, then compute bit positions 24 to 31 of b_{31} for both carried number patterns as shown in Fig. 5. Then, compute bits 24–31 of $d_{33} + a'_{33}$ by considering both carried number patterns from bit 23 to 24.
   (c) For each item in the table, check whether bits 24–31, in total 8 bits, of $d_{33} + a'_{33}$ are matched with $H_b - c_0$.
   (d) If matched, compute p_{30} to p_{33} by the corresponding m_i, and check whether or not all values are matched.
   (e) If all bits are matched, the corresponding message and p_0 is a pseudo-preimage.

Complexity Evaluation. Assume the complexity for computing 1 step is 1/33 33-step RIPEMD computations. The computational complexity of the above procedure is as follows. Step 3a takes $2^{11} \cdot \frac{3}{33}$. Since the success probability of 3b is 2^{-4}, $2^7 (= 2^{11} \cdot 2^{-4})$ items are stored in the table. Step 4 is negligible. Steps 5a and 5b take $2^{32} \cdot (\frac{17}{33} + \frac{3}{33})$. In 5b, $2^{34} (= 2^{32} \cdot 2 \cdot 2)$ items are produced. Therefore, in 5c, $2^{41} (= 2^{34} \cdot 2^7)$ pairs are compared, and after 8-bit matching for both carried number patterns, $2^{33} (= 2^{41} \cdot 2^{-8})$ pairs will remain. In 5d, we compute p_{30} and p_{31} at the complexity of $2^{28} (= 2^{33} \cdot \frac{2}{64})$, and by applying additional 56-bit match and checking the correctness of the guess for the carried number patterns, $2^{-25} (= 2^{33} \cdot 2^{-56} \cdot 2^{-2})$ pair will remain. Furthermore, by computing p_{31} and p_{32} at negligible complexity, we obtain $2^{-89} (= 2^{-25} \cdot 2^{-64})$ pair that is matched with 128 bits. The dominant complexity so far is 2^{32} of 5a and 5b. Therefore, by repeating the attack 2^{89} times, we obtain a pseudo-preimage at the complexity of $2^{121} (= 2^{32} \times 2^{89})$. Finally, by applying the technique in Section 3.1[3], this pseudo-preimage attack is converted to the preimage attack with a complexity of $2^{125.5}$. In the above procedure, a memory is used to store 2^7 $(m_2, p_0, H_b - c_0)$s at step 3b. Therefore, the memory complexity of this attack is approximately $2^7 \times 6$ words.

4.2 Attack on Intermediate 35 Steps

Similar to the attack on the first 33 steps, this attack is a meet-in-the-middle attack, but the approach is different. The strategy is shown in Fig. 6. In this approach, we start the meet-in-the-middle attack from an intermediate step of either two copies of the compression functions. Let us start from the left side. We separate the compression function so that one chunk includes two neutral messages that can form a local collision. This strategy was first used for the 1-block preimage attack on MD4 [1]. Due to the property of the local collision, the value of a pseudo-preimage is always fixed to a constant value. Therefore, we can consider the feedforward as constant addition and can compute the second chunk independently of the first chunk. Finally, we perform the meet-in-the-middle attack at the right side. The chunk we use is shown in Fig. 7.

The attack procedure is similar to Aoki and Sasaki's one-block MD4 preimage attack [1], and how to construct a local collision in the second round is explained in Leurent's MD4 preimage attack [11]. Therefore, because of the limited space, we omit the detailed attack procedure. Since both of the first and second chunks have 2^{32} free bits, the complexity of finding the pseudo-preimage is 2^{96}, and this attack, at the complexity of 2^{113}, is converted to the preimage attack. The memory complexity is approximately $2^{32} \times 5$ words.

[3] Several techniques converting partial-pseudo-preimages to preimages have been proposed [11,3]. However, since our attack does not find partial-pseudo-preimages efficiently, these techniques cannot be applied.

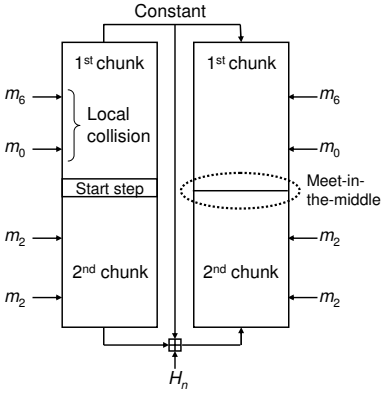


Fig. 6. Outline of strategy 2

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index	**0**	1	**2**	3	4	5	**6**	7	8	9	10	11	12	13	14	15
	excluded							first chunk								
Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index	7	4	13	1	10	**6**	15	3	12	**0**	9	5	14	**2**	11	8
	first chunk													2nd chunk		
Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index	3	10	**2**	4	9	15	8	1	14	3	**0**	**6**	11	13	5	12
	second chunk										excluded					

Chunk separations are identical on both sides.

Fig. 7. Chunks for intermediate 35 steps of RIPEMD

5 Cryptanalyses on Double-Branch Hash Functions

In this section, we analyze the preimage resistance of double length parallel hash functions. Specifically we give a study of relations of the splice-and-cut technique and swapping functions.

5.1 Extended MD4

In Extended MD4, two copies of MD4 with different IV and constants are computed. The swapping function of Extended MD4 exchanges the values of a_{16} and a'_{16}, a_{32} and a'_{32}, and a_{48} and a'_{48}.

MD4 has already been broken by using the splice-and-cut technique [1]. In this research, we found that the swapping function of Extended MD4 cannot prevent the splice-and-cut technique, namely, preimages are generated by almost the same approach as MD4. This is caused by the fact that the message schedules of two MD4 computations are exactly the same as original MD4. Due to this

fact, when we compute chunks in one side, we can also compute the value for the other side. Since we know the values for both sides, we can exchange them according to the swapping function. This means the swapping functions do not contribute to prevent our attack.

The chunk separation is the same as that shown in Ref. [1, Fig.5]. However, the partial-fixing technique can be improved for Extended MD4. In this attack, since we compare the results of two chunks on both compression functions, the number of bits matched by the meet-in-the-middle attack can increase. This enables us to reduce the fixed bits in neutral words, hence free bits in neutral words increase and the meet-in-the-middle attack becomes efficient.

By the partial-fixing technique, we fix the lower 5 bits of the neutral word and examine the 30-bit matching (= 5 bits × 6 words). This results in the pseudo-preimage attack[4] with a complexity of 2^{229}. This, with a complexity of $2^{243.5}$, is converted to a preimage attack with the algorithm explained in Section 3.1. The memory complexity is approximately $2^{27} \times 11$ words.

5.2 RIPEMD-256 and RIPEMD-320

In RIPEMD-256 and -320, the message orders of two copies of the compression functions are different. Therefore, different from Extended MD4, the attack cannot be applied in a straightforward manner. Note that the swapping function exchanges the value of a_{16} and a'_{16}, b_{32} and b'_{32}, c_{48} and c'_{48}, and so on.

To attack RIPEMD-256, we first search for a pair of neutral words that can attack as many steps as possible on either side. Then, on the other side, we check if we can divide the steps into two chunks so that the intermediate chaining variables that are used in the swapping function can also be computed. Selected neutral words and chunks are shown in Fig. 8.

As shown in Fig. 8, we skip eight steps when we attack the right side of MD4 by using the partial-matching and partial-fixing techniques. As introduced in Ref. [1], the partial-fixing technique, which increases the matching candidate twice, enables us to partially compute four steps in backward computation and one step in forward computation. The partial-matching technique enables us to skip three steps. Finally, eight steps can be skipped.

Avoid Swapping Function. In this attack, we assume that a_{16} and a'_{16}, b_{32} and b'_{32}, and c_{48} and c'_{48} are exchanged. d_{64} and d'_{64} are not exchanged since Step 63 is excluded from the attack target.

As you can see in Fig 8, b_{32} and b'_{32} are included in the second chunk and c_{48} and c'_{48} are included in the first chunk. Therefore, by computing both sides simultaneously, we can compute the values that follow the swapping function. a_{16} and a'_{16} are included in the skipped steps. When we check the matching of the results of both chunks, we do not use the values of a_{16} and a'_{16}. Therefore, swapping a_{16} and a'_{16} does not affect the attack complexity.

[4] If the previous attack is applied without improving the partial-fixing technique, this complexity would be 2^{241}.

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index L	(0)	1	2	3	4	5	6	7	8	9	(10)	11	12	13	14	15
index R	5	14	7	(0)	9	2	11	4	13	6	15	8	1	(10)	3	12
	first chunk													skip		

Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index L	7	4	13	1	(10)	6	15	3	12	(0)	9	5	2	14	11	8
											second chunk					
index R	6	11	3	7	(0)	13	5	(10)	14	15	8	12	4	9	1	2
	skip					second chunk										

Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index L	3	(10)	14	4	9	15	8	1	2	7	(0)	6	13	11	5	12
	second chunk							first chunk								
index R	15	5	1	3	7	14	6	9	11	8	12	2	(10)	(0)	4	13
	second chunk													first chunk		

Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
index L	1	9	11	(10)	(0)	8	12	4	13	3	7	15	14	5	6	2
	first chunk														excluded	
index R	8	6	4	1	3	11	15	(0)	5	12	2	13	9	7	(10)	14
	first chunk														excluded	

Fig. 8. Chunks for first 62 steps of RIPEMD-256

Outline of Attack Procedure. Fix messages $m_j, j \notin \{0, 10\}$, p_{39}, and p'_{45}, where p_j is a variable for the left side and p'_j is for the right side. In the first chunk, compute $R_j(p_j, m_{\pi(j)})$ and $R'_j(p'_j, m_{\pi(j)})$ to obtain p_{48} and p'_{48} and swap c_{48} and c'_{48} to follow the swapping function. Then, we compute $R'_j(p'_j, m_{\pi(j)})$ until we obtain p'_{13} and store the results in a table. In the second chunk, compute $R_j^{-1}(p_j, m_{\pi(j)})$ and $R_j'^{-1}(p'_j, m_{\pi(j)})$ to obtain p_{32} and p'_{32} and swap b_{32} and b'_{32} to follow the swapping function. Then, we compute $R_j'^{-1}(p'_j, m_{\pi(j)})$ until we obtain p'_{21} and check whether the result matches items in the table by using the partial-matching and partial-fixing techniques. Hence, a pseudo-preimage for the right side is found efficiently, and by repeating this attack 2^{128} times, one of the resulting pseudo-preimages will also be the pseudo-preimage for the left side.

Complexity Estimation. When we attack the right side, we use the splice-and-cut technique. Since the partial-fixing technique is used, the complexity to find a pseudo-preimage of the right side is 2^{112}. If a pseudo-preimage of the right side is found, we check whether the message is also a pseudo-preimage of the left side. This occurs with a probability of 2^{-128}. Therefore, with a complexity of 2^{240}, we obtain a pseudo-preimage, and this, with a complexity of 2^{249}, is converted to a preimage. The memory complexity is approximately $2^{16} \times 9$ words.

Preimage Attack on Intermediate 64 Steps of RIPEMD-320. With the same strategy as the attack on RIPEMD-256, the intermediate 64 steps of RIPEMD-320 can be attacked. From Steps 12–75 are our attack target, and we select m_{10} and m_{11} as neutral words. Since the attack strategy is the same as that of RIPEMD-256, we show details of the chunks in Appendix A, Fig. 9.

Since the partial-fixing technique is necessary, the complexity of the pseudo-preimage attack is 2^{304}, and this, with a complexity of 2^{313}, is converted to a preimage attack. The memory complexity is approximately $2^{16} \times 7$ words.

6 Conclusion

We first described preimage attacks on RIPEMD. The first attack focuses on differentials of two copies of the compression function and attacks the first 33 steps. The second attack uses local collision and attacks the intermediate 35 steps. We next analyzed the preimage resistance of double-length hash functions. Our attacks find preimages of full Extended MD4, the first 62 steps of RIPEMD-256, and the intermediate 64 steps of RIPEMD-320 faster than the brute force attack does. We believe that analyses presented in this paper will contribute to greater understanding of the security of double-branch hash functions.

References

1. Aoki, K., Sasaki, Y.: Preimage attacks on one-block MD4, 63-step MD5 and more. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 82–98 (2008)
2. Aumasson, J.-P., Meier, W., Mendel, F.: Preimage attacks on 3-pass HAVAL and step-reduced MD5. In: Workshop Records of SAC 2008, Sackville, Canada, pp. 99–114 (2008); ePrint version is available at IACR Cryptology ePrint Archive: Report 2008/183, http://eprint.iacr.org/2008/183.pdf
3. Canniére, C.D., Rechberger, C.: Preimages for reduced SHA-0 and SHA-1. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 179–202. Springer, Heidelberg (2008); (slides on preliminary results were appeared at ESC 2008 seminar http://wiki.uni.lu/esc/)
4. Debaert, C., Gilbert, H.: The RIPEMD^L and RIPEMD^R improved variants of MD4 are not collision free. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 65–74. Springer, Heidelberg (2002)
5. Dobbertin, H.: Cryptanalysis of MD4. Journal of Cryptology 11(4), 253–272 (1997); First result was announced at FSE 1996
6. Dobbertin, H.: RIPEMD with two-round compress function is not collision-free. Journal of Cryptology 10(1), 51–69 (1997)
7. Dobbertin, H., Bosselaers, A., Preneel, B.: RIPEMD-160: A strengthened version of RIPEMD. In: Gollmann, D. (ed.) FSE 1996. LNCS, vol. 1039, pp. 71–82. Springer, Heidelberg (1996)
8. Hong, D., Chang, D., Sung, J., Lee, S., Hong, S., Lee, J., Moon, D., Chee, S.: A new dedicated 256-bit hash function: FORK-256. In: Robshaw, M. (ed.) FSE 2006. LNCS, vol. 4047, pp. 195–209. Springer, Heidelberg (2006)
9. International Organization for Standardization. ISO/IEC 10118-3:2004, Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions (2004)

10. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
11. Leurent, G.: MD4 is not one-way. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 412–428. Springer, Heidelberg (2008)
12. Mendel, F., Pramstaller, N., Rechberger, C., Rijmen, V.: On the collision resistance of RIPEMD-160. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 101–116. Springer, Heidelberg (2006)
13. Mendel, F., Rijmen, V.: Weaknesses in the HAS-V compression function. In: Nam, K.-H., Rhee, G. (eds.) ICISC 2007. LNCS, vol. 4817, pp. 335–345. Springer, Heidelberg (2007)
14. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press, Boca Raton (1997)
15. Park, N.K., Hwang, J.H., Lee, P.J.: HAS-V: A New Hash Function with Variable Output Length. In: Stinson, D.R., Tavares, S. (eds.) SAC 2000. LNCS, vol. 2012, pp. 202–216. Springer, Heidelberg (2001)
16. RIPE Integrity Primitives, Berlin, Heidelberg, New York. Integrity Primitives for Secure Information Systems, Final RIPE Report of RACE Integrity Primitives Evaluation, RIPE-RACE 1040 (1995)
17. Rivest, R.L.: The MD4 message digest algorithm. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 303–311. Springer, Heidelberg (1991); Also appeared in RFC 1320, http://www.ietf.org/rfc/rfc1320.txt
18. Ronald, L.R.: Request for Comments 1321: The MD5 Message Digest Algorithm. The Internet Engineering Task Force (1992), http://www.ietf.org/rfc/rfc1321.txt
19. Saarinen, M.-J.O.: A meet-in-the-middle collision attack against the new FORK-256. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 10–17. Springer, Heidelberg (2007)
20. Sasaki, Y., Aoki, K.: Preimage attacks on 3, 4, and 5-pass HAVAL. In: Pieprzyk, J.P. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 253–271. Springer, Heidelberg (2008)
21. Sasaki, Y., Aoki, K.: Finding preimages in full MD5 faster than exhaustive search. In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 134–152. Springer, New York (2009)
22. U.S. Department of Commerce, National Institute of Standards and Technology. Federal Register 72(212) (November 2, 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
23. U.S. Department of Commerce, National Institute of Standards and Technology. Secure Hash Standard (SHS) (Federal Information Processing Standards Publication 180-3) (2008), http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf
24. Wang, G., Wang, S.: Preimage attack on hash function RIPEMD. In: Bao, F., Li, H., Wang, G. (eds.) ISPEC 2009. LNCS, vol. 5451, pp. 274–284. Springer, Heidelberg (2009)
25. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)

A Chunks for Intermediate 64-Step RIPEMD-320

Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
index L	0	1	2	3	4	5	6	7	8	9	**10**	**11**	12	13	14	15
						excluded								first chunk		
index R	5	14	7	0	9	2	**11**	4	13	6	15	8	1	**10**	3	12
						excluded								first chunk		

Step	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
index L	7	4	13	1	**10**	6	15	3	12	0	9	5	2	14	**11**	8
							first chunk								skip	
index R	6	**11**	3	7	0	13	5	**10**	14	15	8	12	4	9	1	2

Step	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
index L	3	**10**	14	4	9	15	8	1	2	7	0	6	13	**11**	5	12
	skip							second chunk								
index R	15	5	1	3	7	14	6	9	**11**	8	12	2	**10**	0	4	13
														second chunk		

Step	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
index L	1	9	**11**	**10**	0	8	12	4	13	3	7	15	14	5	6	2
	2nd chunk								first chunk							
index R	8	6	4	1	3	**11**	15	0	5	12	2	13	9	7	**10**	14
						second chunk									first chunk	

Step	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
index L	4	0	5	9	7	12	2	**10**	14	1	3	8	**11**	6	15	13
						first chunk								excluded		
index R	12	15	**10**	4	1	5	8	7	6	2	13	14	0	3	9	**11**
			first chunk											excluded		

Fig. 9. Chunks for intermediate 64 steps of RIPEMD-320

On the Weak Ideal Compression Functions

Akira Numayama and Keisuke Tanaka

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{numayam4,keisuke}@is.titech.ac.jp

Abstract. In SAC 2006, Liskov introduced the weak ideal compression functions. He proved that a hash construction based on these functions is indifferentiable from the random oracle. In ICALP 2008, Hoch and Shamir applied Liskov's idea and proved the indifferentiability of another hash construction. However, these proofs of indifferentiability can have gaps in certain situations. In this paper, we formalize these situations and propose the simulation method which covers these situations. In particular, we apply our simulation method to the latter proof of indifferentiability, and concretely analyze the security of the latter hash construction. We can derive a lower bound to find a collision in the concatenated hash construction, which covers the gaps of the original proof.

Keywords: random oracle, weak ideal compression function, indifferentiability, hash construction.

1 Introduction

In cryptography, hash functions have an important role. There have been numerous researches on the construction of hash functions. In order to handle messages of arbitrary length, it is common way to design hash functions based on the Merkle-Damgård iterated construction [4,12], which repeatedly applies a compression function to each successive block of the message. Let f be a compression function and let IV be a fixed initial value. Then, the Merkle-Damgård iterated construction can be described as follows.

$$H(M) = f(f(...(f(IV, m_1), m_2), \dots), m_k),$$

where $M = m_1 \| m_2 \| \cdots \| m_k$ is a k-block message. This design concept is in fact applied to the standard hash functions like SHA-1 [13] and MD5 [15].

There are many security properties for hash functions such as collision resistance and first-preimage resistance (one-wayness). Hash functions used in cryptography should have these properties. In particular, SHA-1 and MD5 were expected to be collision resistant. However, recent cryptanalyses of hash functions have found significant attacks [17,18] against various hash functions, including SHA-1 and MD5. Furthermore, there have been found many generic attacks [6,9,8] against the iterated hash functions.

1.1 Previous Works

The attacks mentioned above have brought our attention to the design of hash functions. A number of research have studied it and its security.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 232–248, 2009.

Indifferentiability framework: The notion of indifferentiability introduced by Maurer, Renner, and Holenstein [11] is useful in order to analyze the security of hash constructions. It is similar to the concept of indistinguishability, and it describes that two systems are indistinguishable despite having extra access to the internal structure of the systems. If we can prove a hash function is indifferentiable from the random oracle, then it has all the security properties of the random oracle model, including collision resistance and first-preimage resistance.

Coron, Dodis, Malinaud, and Puniya [2] applied the indifferentiability framework to the Merkle-Damgård iterated construction, and showed that some variants of the Merkle-Damgård iterated construction can be proved to be indifferentiable from the random oracles if the ideal primitives are used as the underlying compression functions.

Now, we review the formal definition of the indifferentiability. Let C^{Γ} be a hash construction with access to oracle Γ. Here, Γ represents the underlying ideal primitive, and let $\mathcal{RO}^h$ represent the random oracle which idealizes a hash function h. Then, the indifferentiability of C^{Γ} from $\mathcal{RO}^h$ is defined as follows.

Definition 1 (Indifferentiability [11,2]). *A construction C^{Γ} is (q, ϵ) indifferentiable in the presence of Γ from a random oracle $\mathcal{RO}^h$ if there exists a polynomial-time simulator S, such that for every distinguisher D which uses at most q oracle queries, the following holds:*

$$\left|\Pr[D^{C^{\Gamma},\Gamma} = 1] - \Pr[D^{\mathcal{RO}^h, S^{\mathcal{RO}^h}} = 1]\right| < \epsilon.$$

Notice that this definition is slightly different from that of indistinguishability in that the simulator has to simulate the behavior of Γ maintaining consistency not only with Γ itself but also with the random oracle $\mathcal{RO}^h$.

The following example illustrates the fact that a simple iterated hash construction is not indifferentiable from the random oracle. Let $C^{\mathcal{RO}^f}$ be a simple iterated hash construction built from a finite length input random oracle $\mathcal{RO}^f$ which idealize a compression function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. That is, for a k-block message $M = m_1 \| m_2 \| \cdots \| m_k$, the hash construction $C^{\mathcal{RO}^f}$ is described as follows.

$$C^{\mathcal{RO}^f}(M) = \mathcal{RO}^f(\mathcal{RO}^f(...(\mathcal{RO}^f(IV, m_1), m_2), \ldots), m_k).$$

Then, two pairs $(C^{\mathcal{RO}^f}, \mathcal{RO}^f)$ and $(\mathcal{RO}^h, S^{\mathcal{RO}^h})$ are differentiable for any simulator S. Let (α, β) be one of the pairs. We can construct the distinguisher D as follows. First, the distinguisher D queries the hash value of m_1 to α as $g_1 = \alpha(m_1)$. Then, D queries the hash value of (g_1, m_2) to β as $g_2 = \beta(g_1, m_2)$. Finally, D queries the hash value of $m_1 \| m_2$ to α as $g = \alpha(m_1 \| m_2)$. If $g = g_2$, the distinguisher returns 1, and otherwise 0. In the case of $(C^{\mathcal{RO}^f}, \mathcal{RO}^f)$, the equality $g = g_2$ always holds and D returns 1 with probability 1. On the other hand, in the case of $(\mathcal{RO}^h, S^{\mathcal{RO}^h})$, the simulator S does not know m_1 and hence does not know the value $g = \mathcal{RO}^h(m_1 \| m_2)$ which distributes uniformly at random. Therefore, on input (g_1, m_2), he cannot maintain the consistency with the random oracle $\mathcal{RO}^h$. That is, he cannot make the value $g_2 = S^{\mathcal{RO}^h}(g_1, m_2)$ such that the equality $g = g_2$ holds. In this case, D returns 1 with negligible probability.

The above example shows that for the indifferentiability the simulator S needs to simulate the behavior of Γ maintaining both consistency with (1) Γ itself and (2) the random oracle $\mathcal{RO}^h$.

Model of weak ideal compression functions: Liskov [10] introduced the model of weak ideal compression functions. This model idealizes the compression functions as the random oracle and captures the vulnerability of compression functions to the first-preimage finding attack by the additional attack oracles. Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be a compression function. There are three oracles in this model, i.e., the forward oracle, the backward oracle, and the bridging oracle. Given (x, y), the forward oracle provides the value $z = f(x, y)$. The backward oracle and the bridging oracle provide the first-preimages of f. Given (y, z), the backward oracle provides one of x such that $z = f(x, y)$ uniformly at random. Similarly, given (x, z), the bridging oracle provides one of y such that $z = f(x, y)$ uniformly at random. The latter two oracles make the compression function no longer first-preimage resistant.

In [10], he presented a new hash construction, the Zipper hash construction, and proved indifferentiability in this model. That is, the Zipper hash construction is indifferentiable from the random oracle if the underlying compression functions are weak ideal compression functions. Moreover, Hoch and Shamir [5] applied Liskov's idea and showed a similar result for the XORed hash construction $C(M) = F(M) \oplus G(M)$, where both the hash functions F and G are iterated.

1.2 Our Contributions

First, we point out that the simulators in [10,5] can have some problems in certain situations when the simulators simulate the behavior of Γ maintaining the consistency with Γ itself. Then, we formalize these situations and propose the simulation method which covers these gaps. In particular, we apply our simulation method to the proof of indifferentiability in [5], and concretely analyze the security of the XORed hash construction. We can derive a lower bound to find a collision in the concatenated hash construction, which covers the gaps of the original proof [5].

Problems in the previous works: Liskov [10] proposed the Zipper hash construction and proved it indifferentiable from the random oracle in the model of weak ideal compression functions. There are three oracles in this model. Let Γ represent these three oracles. In order to prove the indifferentiability, he constructed the simulator S which simulates the behavior of Γ maintaining both consistency with (1) Γ itself and (2) the random oracle $\mathcal{RO}^h$. While the latter consistency is satisfied by his simulator, the former consistency is not always satisfied.

The problem of Liskov's simulator is as follows: In the simulation of the backward oracle (resp. the bridging oracle), Liskov's simulator always provides new x (resp. new y). In the case of the bridging oracle, if m is sufficiently larger than n, then each pair of (x, z) can have an adequate number of possible answers. Therefore, his simulator can work in this case. However, in the case of the backward oracle, some of the pairs (y, z) have multiple possible answers, but some have no answers. The simulator has to take into account this fact in order to maintain the consistency.

Hoch and Shamir [5] and Numayama, Isshiki, and Tanaka [14] proposed answers to this problem. Hoch and Shamir [5] mentioned it in the same indifferentiability context and in the same model of weak ideal compression functions. In contrast, Numayama et al. mentioned it in the reduction context and in a bit relaxed model.

In particular, Hoch and Shamir [5] showed it in the case that m is sufficiently larger than n, and they mentioned that they could also do in the same way in the case that m is not sufficiently larger than n. However, if we carefully observe what will occur in such a case, we see that we cannot simply extend their simulator. We will describe the problem of their simulator in Section 4.

In this paper, we formalize this problem in order to cover the case that m is not sufficiently larger than n. We propose the simulation method of the forward oracle, the backward oracle, and the bridging oracle. Our simulation method is based on the simulation method in [14]. By using our simulation method, we can also prove the indifferentiability of the Zipper hash construction and the XORed hash construction in the case that m is not sufficiently larger than n.

A lower bound for finding collisions of the concatenated hash construction: Joux [6] found a multicollision attack to find collisions against the concatenated hash construction $H(M) = F(M) \| G(M)$ when at least one of the underlying hash functions F and G are iterated. If the outputs of F and G are n bits, then his attack succeeds in finding a collision in expected time $O(n2^{n/2})$. This is much faster than the birthday paradox based attack, which takes expected time $O(2^n)$ to find a collision for any function of $2n$-bit output. His result implies that the concatenated hash construction does not always improve the security of underlying hash functions.

In Joux's attack, he used the birthday attack to find collisions in the underlying compression functions of F or G. It costs $O(2^{n/2})$ time to find a collision. Joux then posed the question whether the ability to find collisions efficiently in both the underlying compression functions of F and G can help the attacker to improve the complexity of his attack.

Hoch and Shamir [5] showed that in the model of weak ideal compression functions, the XORed hash construction $C(M) = F(M) \oplus G(M)$ is indifferentiable from the random oracle when less than $O(2^{n/2})$ queries are made. They concluded that since finding collisions in $H(M) = F(M) \| G(M)$ implies finding collisions in $C(M)$ as well, the indifferentiability of $C(M)$ will give a lower bound on the number of queries required to find a collision in $H(M)$.

Their result proved that even in a powerful attack scenario where the attacker can find not only collisions but also first-preimages in all the compression functions in unit time, it is required $O(2^{n/2})$ time to find a collision in the concatenated hash construction. We note that they showed this result in the case that m is sufficiently larger than n, where m, n are the parameters for the underlying compression functions $f,\ g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$.

In contrast to their result, we formally prove the indifferentiability of $C(M)$ including the case that m is not sufficiently larger than n. We can derive a lower bound such that in the same attack scenario as in [5], finding a collision in the concatenated hash construction requires more than $O(2^{m/4})$ or $O(2^{n/4})$ time in the case that m is not sufficiently larger than n.

Organization: In Section 2, we review the model of weak ideal compression functions. First, we point out the problem in [10], and consider the simulation method in a simple restricted version in Section 3. Then, in Section 4, we point out the problem in [5], and consider the simulation method in the full version. Section 5 shows an application of our simulation method to the XORed hash construction. Finally, Section 6 concludes this paper.

2 Model of Weak Ideal Compression Functions

2.1 Notation

If $\mathcal{D}$ is a distribution, $x \leftarrow \mathcal{D}$ denote that x is sampled according to $\mathcal{D}$, and let $f_{\mathcal{D}}(x)$ be the probability mass function of distribution $\mathcal{D}$. Let $\mathrm{Bn}(N, p)$ be the binomial distribution with N trials and success probability p.

Let S be a finite set. Let $s \leftarrow S$ denote that s is sampled from the uniform distribution on S. $\#S$ denotes the number of elements in S. If $\mathcal{A}$ is a probabilistic machine and x is an input, let $\mathcal{A}(x)$ denote the output distribution of $\mathcal{A}$ on input x.

Finally, for a table $\mathbb{T} = \{(x, y)\}$, we define $\mathbb{T}(y) = \{(\tilde{x}, \tilde{y}) \in \mathbb{T} \mid y = \tilde{y}\}$. Similarly, for a table $\mathbb{T} = \{(x, y, z)\}$, we define $\mathbb{T}(y) = \{(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathbb{T} \mid y = \tilde{y}\}$ and $\mathbb{T}(y, z) = \{(\tilde{x}, \tilde{y}, \tilde{z}) \in \mathbb{T} \mid y = \tilde{y}, z = \tilde{z}\}$.

2.2 Model

Let X, Y, and Z be finite sets. The model of weak ideal compression functions first introduced by Liskov [10] has a compression function f chosen randomly from all the functions such that $f : X \times Y \to Z$. That is, f maps an element $(x, y) \in X \times Y$ to an element $f(x, y) = z \in Z$.

Let $\mathbb{T}_f = \{(x, y, f(x, y)) \mid (x, y) \in X \times Y\}$ be the table which defines the correspondence of all the elements in $X \times Y$ with the elements in Z. That is, there is an entry $(x, y, z) \in \mathbb{T}_f$ if and only if $z = f(x, y)$ holds. We note that we can identify the compression function f with the table $\mathbb{T}_f$.

In this model there are three oracles, i.e., the forward oracle, the backward oracle, and the bridging oracle. These oracles are defined as follows:

Forward Oracle $\mathcal{F}w\mathcal{O}^f$: Given (x, y), the forward oracle returns z such that $(x, y, z) \in \mathbb{T}_f$

Backward Oracle $\mathcal{B}w\mathcal{O}^f$: Given (y, z), if there is any entry $(x, y, z) \in \mathbb{T}_f$, then the backward oracle returns such x uniformly at random. Otherwise, it returns $\perp$.

Bridging Oracle $\mathcal{B}r\mathcal{O}^f$: Given (x, z), if there is any entry $(x, y, z) \in \mathbb{T}_f$, then the bridging oracle returns such y uniformly at random. Otherwise, it returns $\perp$.

Remark 1. In the previous works [10,5], they identified the forward oracle and the underlying compression function. However, in this paper as in [14] we explicitly distinguish them and make the forward oracle to be the interface to the underlying compression function. This setting helps us to make the model of weak compression functions well-defined.

2.3 Difference from the Random Oracle Model

We note an important difference between the random oracle model and the model of weak ideal compression functions. Let $h : X \times Y \to Z$ be the hash function in the random oracle model, and let $\mathbb{T}_h$ be the table $\mathbb{T}_h = \{(x, y, h(x, y)) \mid (x, y) \in X \times Y\}$. Furthermore, let $\mathcal{RO}^h$ be the random oracle, which provide $z = h(x, y)$ when we query the hash value of (x, y).

In the random oracle model, each output of the random oracle $\mathcal{RO}^h$ is independent of the other entry in the table $\mathbb{T}_h$ and uniformly distributed. This property of the random oracle model is called uniformity. In contrast to the situation in the random oracle model, when it comes to the model of weak ideal compression functions, this property is not easily attained.

This is because, when we query (y, z), the backward oracle *uniformly* returns one of the preimages of z under the function $f_y(x) = f(x, y)$, and hence if there are n_{yz} elements, then it outputs one of them with probability $\frac{1}{n_{yz}}$. Here, some information on n_{yz} may be revealed, and each entry in the table $\mathbb{T}_f$ has some relation depending on n_{yz}.

In order to verify this situation, let us consider the following extreme case. Let $z^* = f(x^*, y^*)$ for some $(x^*, y^*) \in X \times Y$, and let $n_{y^*z^*}$ be the number of the preimages of z^* under the function $f_{y^*}(x) = f(x, y^*)$. If $n_{y^*z^*} = 1$, then the backward oracle always returns the same x^* on input (y^*, z^*). In this case, we know the information that $n_{y^*z^*} = 1$ through the backward oracle. This implies that there is exactly one element that maps to z^* under the function $f_{y^*}(x) = f(x, y^*)$, and for any $x \neq x^*$ the value $\mathcal{FwO}^f(x, y^*)$ cannot be z^*, i.e., the value $\mathcal{FwO}^f(x, y^*)$ does not strictly follow the uniform distribution on Z. Therefore, we no longer have the uniformity in the model of weak ideal compression functions.

3 Our Simulation Methods (Restricted Version)

In this section, we begin by considering the simulation in the simple restricted version where we have only the forward oracle and the backward oracle available, but not the bridging oracle. The simulation in the full version is given in Section 4.

Before giving our methods, let us review the simulation in the random oracle model. In the random oracle model, each entry in the table is independent and uniformly distributed. That is, the random oracle model has the uniformity. Therefore, in a standard way, we simulate the random oracle by maintaining a table $\mathbb{T}$ that is initially empty as the following algorithm Std.RO (Algorithm 1).

In contrast to the simulation in the random oracle model, when it comes to the restricted model of weak ideal compression functions, there are two difficulties in the simulation. First, in the restricted model of weak ideal compression functions, we need to simulate the backward oracle in addition to the forward oracle, i.e., when the value (y, z) is queried to the backward oracle, we need to *uniformly* return one of the preimages of z under the function $f_y(x) = f(x, y)$. However, we do not know the number of the preimages of z under the function $f_y(x) = f(x, y)$, and hence we cannot *uniformly* return one of them. Second, each entry in the table is no longer independent because some information about the number of the preimages of z under the function $f_y(x) = f(x, y)$ are

Algorithm Std.RO($\hat{x}, \hat{y}$)

1. If $(\hat{x}, \hat{y}, z) \in \mathbb{T}$ for some z, then return z.
2. Otherwise,
   (a) pick uniformly $z \leftarrow Z$,
   (b) insert $(\hat{x}, \hat{y}, z)$ in the table $\mathbb{T}$, and
   (c) return z.

Algorithm 1. Standard simulation method of the random oracle

revealed through the backward oracle, i.e., the model of weak ideal compression functions does not have the uniformity. Therefore, in the simulation of the forward oracle, when the hash value of (x, y) is queried, we cannot do as in the algorithm Std.RO.

3.1 Problem of Liskov's Simulation Method

In the restricted model of weak ideal compression functions, the above two difficulties prevent us to simply simulate both the forward oracle and the backward oracle.

Let $X = Z = \{0,1\}^n$ and $Y = \{0,1\}^m$, i.e., $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be the compression function. The problem of Liskov's simulator is as follows: In the simulation of the backward oracle (resp. the bridging oracle), Liskov's simulator always provides new x (resp. new y). In the case of the bridging oracle, if m is sufficiently larger than n, then each pair of (x, z) has an adequate number of possible answers. Therefore, his simulator can work in this case. However, in the case of the backward oracle, some of the pairs (y, z) have multiple possible answers, but some have no answers. The simulator has to take into account this fact in order to maintain the consistency.

Hoch and Shamir [5] and Numayama et al. [14] proposed answers to this problem independently. Hoch and Shamir proposed the simulation method in the model of weak ideal compression functions, and considered to manage the number of preimages according to the Poisson distribution. Their simulation method seems reasonable in the restricted model of weak ideal compression functions. However, they did not prove it. Numayama et al. proposed the simulation method in the weakened random oracle models where the argument of the function is restricted to one, i.e., the underlying function is $z = f_y(x) = f(x, y)$ for fixed y. They considered to manage the number of preimages according to the binomial distribution and proved that their simulation method can simulate the random oracle and the first-preimage oracle in the weakened random oracle models, which correspond to the forward oracle and the backward oracle, in the restricted model of weak ideal compression functions, respectively.

In this paper, we apply the idea of Numayama et al. to the simulation of the forward oracle and the backward oracle in the restricted model of weak ideal compression functions. This is because the simulation methods of Numayama et al. was proved their validity.

3.2 Solution for the Problem of Liskov's Simulation Method

In the restricted case, it is easy to simulate both the forward oracle and the backward oracle by using the idea of Numayama et al. We show the reason why we can apply

Table 1.

	y_j							
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
x_i	*	z_{ij}	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*

the simulation method of the random oracle and the first-preimage oracle, where the underlying hash function is $z = f_y(x) = f(x, y)$ for fixed y. The correspondence of the hash value $z_{ij} = f(x_i, y_j)$ is described in Table 1. Given (y, z), in order to simulate the backward oracle, it only requires the number of preimages of z under the function $f_y(x) = f(x, y)$, and this number only has an effect on the corresponding y's column. This effect is described by the gray color in Table 1. Furthermore, there is nothing else except this number which has an effect on the other entries in Table 1 which are not yet determined.

Therefore, for each $y \in Y$ the function $f_y(x)$ (each y's column in Table 1) is mutually independent, and we can apply the simulation method proposed in [14] to each function $f_y(x)$ in order to simulate the forward oracle and the backward oracle in this restricted model of weak ideal compression functions where the bridging oracle is not available. In Appendix A, we give the concrete algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ and $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ which simulate the forward oracle and the backward oracle, respectively.

The main idea of our simulation methods is to manage the two tables $\mathbb{T}$ and $\mathbb{L}_{yz}$ which are initially empty. The table $\mathbb{T}$ has entries (x, y, z) in order to manage the correspondence of $z = f(x, y)$ as in the standard simulation method in the random oracle model. On the other hand, the table $\mathbb{L}_{yz}$ has entries (y, z, n_{yz}) in order to manage the number of the preimages of z under the function $f_y(x)$. These two tables are commonly used for the algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ and $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$.

As for the first difficulty described at the beginning of this section, the knowledge of the number n_{yz} in the table $\mathbb{L}_{yz}$ enables the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ to uniformly output one of the preimages of queried value z under the function $f_y(x)$, i.e., return one of them with probability $\frac{1}{n_{yz}}$.

As for the second difficulty, the knowledge of the number of n_{yz} also enables the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ to behave like the forward oracle considering the dependence of each entry stemmed from some leakage through n_{yz}. In order to verify how this idea works well, let us consider the extreme case again. Let $z = f(x, y)$ for some $(x, y) \in X \times Y$ and n_{yz} be the number of preimages of z under the function $f_y(x) = f(x, y)$. If $n_{yz} = 1$, then for all $x' \in X$ such that $x' \neq x$, the value $\mathcal{F}wO^f(x', y)$ does not match z. Therefore, when we query the hash value of (x', y), the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}(x', y)$ uniformly outputs $z' \in Z$ such that $z' \neq z$.

Then, by using the following proposition, we can analyze the algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ and $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$, and obtain the following theorem.

Proposition 1 ([7]). *There is a polynomial-time machine* $\mathsf{B_N}$ *such that the distribution* $\mathsf{B_N}(N, p)$ *output by the algorithm* $\mathsf{B_N}$ *is statistically close to the binomial distribution* $\mathrm{B_N}(N, p)$*, where* N *is a positive integer and* $0 \leq p \leq 1$.

Remark 2. In order to simplify the analysis of the algorithms, we ignore the above statistical distance. The statistical distances in Theorems 1 and 2 all arise from the above one.

Theorem 1. *We can simulate the forward oracle and the backward oracle in the restricted model of weak ideal compression functions where the bridging oracle is not available. Formally, the distribution on the outputs of the forward oracle and the backward oracle is statistically close to the distribution on the outputs of the algorithms* $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ *and* $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ *in the* restricted *model of weak ideal compression functions where the bridging oracle is not available.*

Proof. In this restricted case, for each $y \in Y$ the function $f_y(x) = f(x, y)$ is independent from each other, and therefore we can apply the similar simulation methods of the random oracle and the first-preimage oracle as in the first-preimage random oracle model [14]. □

We note that if we replace the role of the bridging oracle and the backward oracle, then in the same way we can obtain the algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Br}}$ and $\mathsf{BrO}_{\mathsf{R}}^{\mathsf{Br}}$ that simulate the forward oracle and the bridging oracle respectively in the restricted model of weak ideal compression functions where the backward oracle is not available.

Remark 3. There are three tables $\mathbb{T}, \mathbb{L}_{yz}$, and $\mathbb{L}_{xz}$ which are used in the four algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$, $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$, $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Br}}$, and $\mathsf{BrO}_{\mathsf{R}}^{\mathsf{Br}}$. Here, we mention which algorithm manages which table. It is helpful to keep this in mind in order to understand the simulation methods in the (full) model of weak ideal compression functions.

All the algorithms manage the table $\mathbb{T}$ on the fly. Furthermore, the former two algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ and $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ also manages the table $\mathbb{L}_{yz}$, whereas the latter two algorithms $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Br}}$ and $\mathsf{BrO}_{\mathsf{R}}^{\mathsf{Br}}$ also manage the table $\mathbb{L}_{xz}$.

4 Our Simulation Methods (Full Version)

Now, we consider the simulation method of three oracles in the (full) model of weak ideal compression functions. Given (y, z), in order to simulate the backward oracle, we need the number n_{yz} of preimages of z under the function $f_y(x) = f(x, y)$. Similarly, given (x, z), in order to simulate the bridging oracle, we need the number n_{xz} of preimages of z under the function $f_x(y) = f(x, y)$.

Therefore, in the (full) model of weak ideal compression functions, in order to simulate at once both the backward oracle and the bridging oracle, we have to manage the numbers n_{yz} and n_{xz} simultaneously. In this time, we manage the three tables $\mathbb{T}$, $\mathbb{L}_{xz}$, and $\mathbb{L}_{yz}$. The table $\mathbb{T}$ has entries (x, y, z) in order to manage the correspondence of $z = f(x, y)$. On the other hand, the table $\mathbb{L}_{xz}$ (and $\mathbb{L}_{yz}$, respectively) has entries (x, z, n_{xz}) (and (y, z, n_{yz}), respectively) in order to manage the number of the preimages of z under the function $f_x(y) = f(x, y)$ (and $f_y(x) = f(x, y)$, respectively).

4.1 Naive But Faulty Idea

We give a naive but faulty idea for the simulation of the forward oracle, and show that this idea does not work well. The naive idea is to determine the value $z = f(x, y)$, if and only if the value $f(x, y)$ is queried as the simulation method in the restricted model of weak ideal compression functions.

Let us examine the simulation of the forward oracle according to the naive idea in the case that the hash values of (x_i, y_j), (x_k, y_l), and (x_k, y_j) are queried in this order.

First, the hash value of (x_i, y_j) is queried. Then, we pick $z_{ij} \leftarrow Z$ uniformly, and insert (x_i, y_j, z_{ij}) in $\mathbb{T}$. Furthermore, we also determine the numbers n_{xz} and n_{yz} of preimages of z under the functions $f_{x_i}(y)$ and $f_{y_j}(x)$ respectively, and insert (x_i, z_{ij}, n_{xz}) in $\mathbb{L}_{xz}$ and insert (y_j, z_{ij}, n_{yz}) in $\mathbb{L}_{yz}$. Here, we note that the number n_{xz} has an effect on the elements of x_i row which are not yet determined, i.e., these elements are not uniformly distributed in Z. Similarly, the number of n_{yz} has an effect on the elements of y_j's column which are not yet determined. These effects are described by the gray color in Table 2.

Next, the hash value of (x_k, y_l) is queried. We have to determine the hash value $f(x_k, y_l)$ as z_{kl}. In this time, z_{kl} is not the gray colored element in Table 2, and hence z_{kl} is independent from the other entries (i.e., z_{kl} is distributed uniformly in Z). Then, we can pick $z_{kl} \leftarrow Z$ uniformly, and insert (x_k, y_l, z_{kl}) in $\mathbb{T}$. Furthermore, we also determine the numbers n'_{xz} and n'_{yz} of preimages of z_{kl} under the functions $f_{x_k}(y)$ and $f_{y_l}(x)$ respectively, and insert (x_k, z_{kl}, n'_{xz}) in $\mathbb{L}_{xz}$ and insert (y_l, z_{kl}, n'_{yz}) in $\mathbb{L}_{yz}$. Then, the numbers n'_{xz} and n'_{yz} have an effect on the elements of x_k's row and y_l's column, respectively (Table 3).

Finally, the value of $f(x_k, y_j)$ is queried. We have to determine the hash value $f(x_k, y_j)$ as z_{kj}. In this time, z_{kj} is the gray colored elements in Table 3, and hence z_{kl} is not independent from the other entries. More properly, for $(x_k, z_{kl}, n_{xz}) \in \mathbb{L}_{xz}$ and $(y_j, z_{ij}, n_{yz}) \in \mathbb{L}_{yz}$, z_{kj} is subject to the effects of n_{xz} and n_{yz}. Now, let us consider the extreme case where $z_{ij} = z_{kl} = z$, $n_{xz} = 1$, and $n_{yz} = \#Y$. Then, z_{kj} must *not* be equal to z by the effect of n_{xz}, whereas z_{kj} must be equal to z by the effect of n_{yz}. These two conditions cannot be satisfied at the same time, and we cannot determine z_{kj} according to this naive idea.

4.2 Problem of Hoch and Shamir's Simulation Method

Hoch and Shamir [5] proposed an answer to the problem of Liskov's simulation method, and proposed the simulation method in the model of weak ideal compression functions. In this section, we show some problem of their simulation method in a certain situation.

Table 2.

	y_j							
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
x_i	*	z_{ij}	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*

Table 3.

	y_j		y_l					
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
x_k	*	*	*	z_{kl}	*	*	*	*
	*	*	*	*	*	*	*	*
x_i	*	z_{ij}	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*

Table 4.

	y_j		y_l					
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
x_k	*	z_{kj}	*	z_{kl}	*	*	*	*
	*	*	*	*	*	*	*	*
x_i	*	z_{ij}	*	*	*	*	*	*
	*	*	*	*	*	*	*	*
	*	*	*	*	*	*	*	*

Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ be the underlying compression function. In the case that m is sufficiently larger than n, their simulation method is as follows. In order to overcome the problem of Liskov's simulation method, they managed the number of preimages according to the Poisson distribution. They managed the number n_{yz} of preimages of z under the function $f_y(x) = f(x, y)$ in order to simulate the backward oracle, and took n_{yz} into account in the simulation. On the other hand, they did not manage the number n_{xz} of preimages of z under the function $f_x(y) = f(x, y)$ in order to simulate the bridging oracle, and they always output new y when we query the preimage of (x, z).

This is because there are expected to be a few possible answers to the (y, z) query, and it is required to take care of it in order to simulate the backward oracle. In contrast, in the case that m is sufficiently larger than n, there are expected to be a large number of possible answers to the (x, z) query, and even if we always output new y, it does not seem to cause much deviation from the distribution on the outputs of the bridging oracle.

In the case that m is not sufficiently larger than n, we also need to take care of the number n_{xz} of preimages z under the function $f_x(y) = f(x, y)$, since there are expected to be a few possible answers to the (x, z) query. Hoch and Shamir [5] noted, in this case, the same special treatment as in simulation of the backward oracle can be given to the bridging oracle. However, in [5], they did not mention the above problem described in the naive idea caused by managing both the numbers of the preimages of z under the functions $f_x(y)$ and $f_y(x)$. Therefore, the simulation method of Hoch and Shamir can suffer from the same problem as in the naive idea.

4.3 Solution for the Problem of Hoch and Shamir's Simulation Method

In this section, we propose the solution to overcome the problem in the naive idea. We define the notion which plays an important role in our simulation method.

Definition 2. *For the tables $\mathbb{T}, \mathbb{L}_{xz}$, and $\mathbb{L}_{yz}$, we call as the* crossed elements *the elements in the table of the hash values $z_{ij} = f(x_i, y_j)$ such that they are in the gray colored row and the gray colored column at once. That is, the elements (x, y) are* crossed elements *if and only if there are $(x, z, n) \in \mathbb{L}_{xz}$ and $(y, z', n') \in \mathbb{L}_{yz}$ for some z, z', n, and n' .*

The problem in the naive idea appears when it comes to decide the value of *crossed elements*. That is, we have to take into account at once both the effects of n_{xz} under the function $f_x(y)$ (gray colored x's row) and the effects of n_{yz} under the function $f_y(x)$ (gray colored y's column). In order to simulate the three oracles, we have to overcome this problem, i.e., the dependence of two functions $f_x(y)$ and $f_y(x)$.

In the previous case, the problem is that there is a dependence of the functions $f_{x_k}(y)$ and $f_{y_j}(x)$ at the time when we need to determine the value z_{kj} of *crossed element*. Therefore, in order to overcome the dependence of the two functions it will be good idea to determine the value z_{kj} in advance of the appearance of this dependence.

Let us carefully observe when the dependence of two functions $f_{x_k}(y)$ and $f_{y_j}(x)$ appear in the same situation where the values of (x_i, y_j), (x_k, y_l), and (x_k, y_j) are queried in this order. Just after the first query (x_i, y_j), there is only $f_{y_j}(x)$ which affects the value

Table 5.

	Naive idea	Our idea
1.	z_{kl}	z_{kl}
2.		z_{kj}
3.	n'_{xz}	n'_{xz}
4.		z_{il}
5.	n'_{yz}	n'_{yz}

Procedure at the second query (x_k, y_l)

1. Uniformly pick $z_{kl} \leftarrow Z$.
2. Choose z_{kj} according to the function $f_{y_j}(x)$.
3. Determine the number of the preimage of z_{kj} and z_{kl} under the function $f_{x_k}(y)$.
4. Choose z_{il} according to the $f_{x_i}(y)$.
5. Determine the number of the preimage of z_{kl} and z_{il} under the function $f_{y_l}(x)$.

Procedure 1.

$z_{kj} = f(x_k, y_j)$ through n_{yz}. When the second query $f(x_k, y_l)$ is made, we determine the value $z_{kl} = f(x_k, y_l)$ and n'_{xz} and n'_{yz}. Here, n'_{xz} affects the function $f_{x_k}(y)$. Now, the dependence of two functions $f_{x_k}(y)$ and $f_{y_j}(x)$ appear at the second query (x_k, y_l).

In order to determine the value z_{kj} before this dependence caused through n_{yz} and n'_{xz} appears, we should decide z_{kj} in advance of the decision of n'_{xz} at the second query. Here, we note for the value z_{il}, the same idea should be applied to the case that the value $f(x_i, y_l)$ is queried. Therefore, the procedure at the second query is modified as follows. In contrast to the naive idea where we determine the values z_{kl}, n'_{xz}, and n'_{yz} in this order, we additionally decide the values z_{kj} and z_{il} in the following order, $z_{kl}, z_{kj}, n'_{xz}, z_{il}$, and then n'_{yz}. See Table 5 for comparison.

We can apply this idea and obtain the more precise procedure described as follows (Procedure 1).

In Step 1, z_{kl} is an independent entry, and we can determine the hash value of (x_k, y_l) uniformly at random. In Step 2, z_{kj} is only dependent on the y_j's column, and we can determine the hash value of (x_k, y_j) according to the function $f_{y_j}(x)$. In Step 3, z_{kj} and z_{kl} are only dependent on the x_k's row, and we can determine the number of preimages according to the function $f_{x_k}(y)$. In Step 4, z_{il} is only dependent on the x_i's row, and we can determine the hash value of (x_i, y_l) according to the function $f_{x_i}(y)$. In Step 5, z_{kl} and z_{il} are only dependent on the y_l's column, and we can determine the number of preimages according to the function $f_{y_l}(x)$.

We can check the dependence of the elements at each step in the procedure as described in Tables 6-11. Table 6 corresponds to the beginning of Step 1, and Table 10 describes the dependence after the procedure. In Tables 6-11, dark gray colored elements are subject to the effects of the number of preimages, whereas light gray colored elements are not, like the other elements. For example, in Table 6, the hash value of (x_k, y_j) (and (x_i, y_l), respectively) is decided according to the $f_{y_j}(x)$ (and $(f_{x_i}(y))$, respectively), and the hash value of (x_k, y_l) is uniformly distributed in Z.

Table 6.

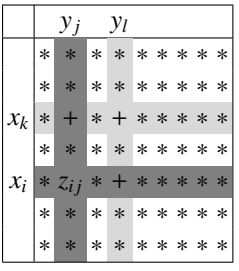


Table 7.

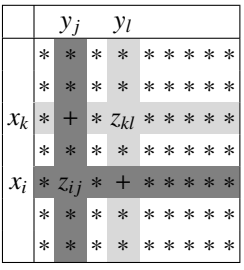


Table 8.

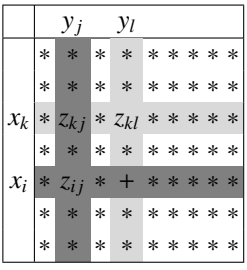


Table 9.

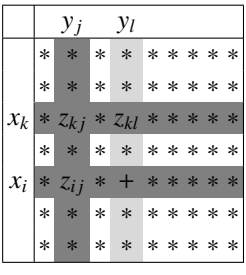


Table 10.

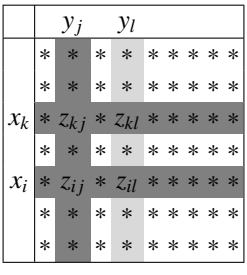
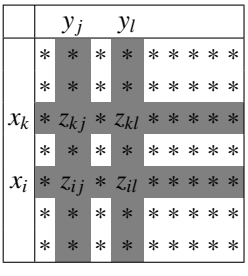


Table 11.



This procedure at the second query (x_k, y_l) is easily extended to the subsequent forward oracle queries. That is, in the simulation of the forward oracle, when queried the hash value of (x, y), we also decide the hash value of the elements which will become *crossed elements* in advance of deciding the number of preimages of $z = f(x, y)$.

Then, we can obtain the algorithm FwO which simulates the forward oracle by applying the above idea. By applying the same idea of keeping the every *crossed element* determined, we can also obtain the algorithms BwO and BrO which simulate the backward oracle and the bridging oracle, respectively. We give the concrete algorithms FwO, BwO, and BrO in the full version of this paper.

Remark 4. These three algorithms are designed according to the idea described above. Therefore, if every *crossed element* are determined at the beginning of each invocation of these algorithms, then every *crossed element* are determined at the end of the invocation. That is, these algorithms run maintaining all the *crossed elements* determined.

We can obtain the following theorem. The proof is given in the full version of this paper.

Theorem 2. *We can simulate the forward oracle, the backward oracle, and the bridging oracle in the (full) model of weak ideal compression functions. Formally, the distribution on the outputs of the forward oracle, the backward oracle, and the bridging oracle is statistically close to the distribution on the outputs of the algorithms* FwO*,* BwO*, and* BrO *in the (full) model of weak ideal compression functions.*

5 An Application of Our Simulation Method to the Proofs of the Indifferentiability

In this section, we apply the simulation method in Section 4 to the proofs of the indifferentiability in [10,5]. In particular, we modify the proof of [5] by applying our simulation method instead of theirs.

Let F and G are iterated hash functions based on the compression functions f and g, respectively. Hoch and Shamir [5] showed that the XORed hash construction $C(M) = F(M) \oplus G(M)$ where both the underlying compression functions of F and G are iterated, is indifferentiable from the random oracle in the model of weak ideal compression functions.

In order to overcome the problem of the simulator in [5] described in Section 4, we apply the simulation method in Section 4 and obtain the following theorem.

Theorem 3. *Let C be the XORed hash construction $C(M) = F(M) \oplus G(M)$, where F and G are iterated hash functions based on the compression functions f and g, respectively. Then, the XORed hash construction C is indifferentiable from the random oracle in the model of weak ideal compression functions.*

More formally, it is stated as follows. Let f and g are the compression functions such that $f,\ g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. Let Γ be an oracle encapsulating the forward oracles, the backward oracles, and the bridging oracles for the compression functions f and g. Then, there is a simulator S such that for every distinguisher D which uses at most q oracle queries, the following holds, where $\alpha = \min(m, n)$.

$$\left|\Pr[D^{C^{\Gamma},\Gamma} = 1] - \Pr[D^{\mathcal{RO}^h, S^{\mathcal{RO}^h}} = 1]\right| \lessapprox O(\frac{q^4}{2^{\alpha}}).$$

In order to prove Theorem 3, we construct a simulator S which simulates Γ in polynomial time. We note that the simulator S must keep consistency with both Γ itself and the random oracle. The simulator S uses the idea of our simulation method in Section 4 for the consistency with Γ, and uses the idea of Hoch and Shamir [5] for the consistency with the random oracle.

In the full version of this paper, we show our main lemma which is useful for the security proof of indifferentiability in the model of weak ideal compression functions. Then, by using this lemma, we formally prove Theorem 3.

6 Conclusion

First, we have pointed out that the previous proofs of indifferentiability [10,5] can have some problems in certain situations. Then, we have formalized these situations and proposed the simulation method for the forward oracle, the backward oracle, and the bridging oracle in the model of weak ideal compression functions. By applying our simulation method to the previous proofs, we can also prove the indifferentiability in the situations where the original proofs does not cover.

In particular, we have applied our simulation method to the proof of indifferentiability [5]. In [5], Hoch and Shamir proved that the XORed hash construction $C(M) =$

$F(M) \oplus G(M)$ is indifferentiable from the random oracle in the case that m is sufficiently larger than n, where the underlying compression functions are $f,\ g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$. Then, they derived a lower bound such that even in a powerful attack scenario where the attacker can find not only collisions but also first-preimages in all the compression functions in unit time, it required $O(2^{n/2})$ time to find a collision in the concatenated hash construction $H(M) = F(M) \,\|\, G(M)$.

In contrast, we have proved the indifferentiability even in the case that m is not sufficiently larger than n, and derived a lower bound such that in the same attack scenario it requires $O(2^{m/4})$ or $O(2^{n/4})$ time to find a collision in the concatenated hash construction in the case that m is not sufficiently larger than n.

References

1. Brassard, G. (ed.): CRYPTO 1989. LNCS, vol. 435. Springer, Heidelberg (1990)
2. Coron, J.-S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-damgård revisited: How to construct a hash function. In: Shoup [16], pp. 430–448
3. Cramer, R. (ed.): EUROCRYPT 2005. LNCS, vol. 3494. Springer, Heidelberg (2005)
4. Damgård, I.: A design principle for hash functions. In: Brassard [1], pp. 416–427
5. Hoch, J.J., Shamir, A.: On the strength of the concatenated hash combiner when all the hash functions are weak. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 616–630. Springer, Heidelberg (2008)
6. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
7. Kawachi, A., Numayama, A., Tanaka, K., Xagawa, K.: Approximation sampling and its application to security proofs in cryptography. In: Symposium on Cryptography and Information Security, pp. 3D1–1 (2009)
8. Kelsey, J., Kohno, T.: Herding hash functions and the nostradamus attack. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 183–200. Springer, Heidelberg (2006)
9. Kelsey, J., Schneier, B.: Second preimages on n-bit hash functions for much less than 2^n work. In: Cramer [3], pp. 474–490
10. Liskov, M.: Constructing an ideal hash function from weak ideal compression functions. In: Biham, E., Youssef, A.M. (eds.) SAC 2006. LNCS, vol. 4356, pp. 358–375. Springer, Heidelberg (2007)
11. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004)
12. Merkle, R.C.: One way hash functions and des. In: Brassard [1], pp. 428–446
13. National Institute of Standards and Technology. Secure hash standard. FIPS 180-2 (August 2002)
14. Numayama, A., Isshiki, T., Tanaka, K.: Security of digital signature schemes in weakened random oracle models. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 268–287. Springer, Heidelberg (2008)
15. Rivest, R.L.: The MD5 message-digest algorithm. Internet Request for Comments, RFC 1321 (April 1992)
16. Shoup, V. (ed.): CRYPTO 2005. LNCS, vol. 3621. Springer, Heidelberg (2005)
17. Wang, X., Yin, Y.L., Yu, H.: Finding collisions in the full SHA-1. In: Shoup [16], pp. 17–36
18. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer [3], pp. 19–35

A Our Simulation Algorithms (Restricted Version)

First, we consider how the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ runs on input $(\hat{x}, \hat{y})$ in the details. If the hash value of $(\hat{x}, \hat{y})$ is already determined, then the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns it. Otherwise, there are two situations depending on whether the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns *old* z which already appears in the table $\mathbb{T}$ or the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns *new* z which does not yet appear in the table $\mathbb{T}$. There are $(\#X - \#\mathbb{T}(\hat{y}))$ elements whose hash values are not yet determined, and among them there are $\sum_{(\hat{y},\tilde{z},\tilde{n}_{yz})\in\mathbb{L}_{yz}}(\tilde{n}_{yz} - \#\mathbb{T}(\hat{y},\tilde{z}))$ elements whose hash values are expected to be old z. Therefore, the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns old z or new z with this ratio. In case of old z, the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ picks old z according to the number of the preimages of each old z. In case of new z, the algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}$ picks new z uniformly at random, and defines the number of preimages of z. The whole algorithm is described in Algorithm 2.

Algorithm $\mathsf{FwO}_{\mathsf{R}}^{\mathsf{Bw}}(\hat{x}, \hat{y})$

1. If $(\hat{x}, \hat{y}, z) \in \mathbb{T}$ for some z, then return z.
2. Let $p(\hat{y})$ be the following probability,
$$p(\hat{y}) = \frac{\sum_{(\hat{y},\tilde{z},\tilde{n}_{yz})\in\mathbb{L}_{yz}}(\tilde{n}_{yz} - \#\mathbb{T}(\hat{y},\tilde{z}))}{\#X - \#\mathbb{T}(\hat{y})}.$$
3. With probability $p(\hat{y})$, return old z as Steps (a)-(b).
   (a) pick $z \leftarrow \mathcal{D}$ according to the following distribution.
$$f_{\mathcal{D}}(z) = \frac{n_{yz} - \#\mathbb{T}(\hat{y}, z)}{\sum_{(\hat{y},\tilde{z},\tilde{n}_{\tilde{z}})\in\mathbb{L}_{yz}}(\tilde{n}_{yz} - \#\mathbb{T}(\hat{y},\tilde{z}))} \text{ for } (\hat{y}, z, n_{yz}) \in \mathbb{L}_{yz}.$$
   (b) insert $(\hat{x}, \hat{y}, z)$ in $\mathbb{T}$ and return z.
4. With probability $1 - p(\hat{y})$, return new z as Steps (a)-(d).
   (a) pick $z \leftarrow Z \setminus \bigcup_{(\hat{y},\tilde{z},\tilde{n}_{yz})\in\mathbb{L}_{yz}}\{\tilde{z}\}$.
   (b) $n'_{yz} \leftarrow \mathsf{B}_{\mathsf{N}}(\#X - \sum_{(\hat{y},\tilde{z},\tilde{n}_{yz})\in\mathbb{L}_{yz}} \tilde{n}_{yz} - 1, \frac{1}{\#Z - \#\mathbb{L}_{yz}(\hat{y})})$.
   (c) $n_{yz} \leftarrow n'_{yz} + 1$.
   (d) insert $(\hat{y}, z, n_{yz})$ in $\mathbb{L}_{yz}$, insert $(\hat{x}, \hat{y}, z)$ in $\mathbb{T}$, and return z.

Algorithm 2. Simulation method of the forward oracle (Restricted Version)

Second, we review how the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ runs on input $(\hat{y}, \hat{z})$ in the details. If $\hat{z}$ is not yet determined (i.e., the number n_{yz} of preimages of $\hat{z}$ under the function $f_{\hat{y}}(x) = f(x, \hat{y})$ is not determined either), then the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ first defines the number n_{yz} of preimages of $\hat{z}$. If $n_{yz} = 0$, which implies that there is no preimage of $\hat{z}$, then the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns $\perp$. Otherwise, there are two situations depending on whether the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns *new* x which does not yet appear in the table $\mathbb{T}$ or the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns *old* x which already appears in the table $\mathbb{T}$. There

are n_{yz} elements whose hash values are expected to be $\hat{z}$ under the function $f_{\hat{y}}(x) = f(x, \hat{y})$, and among them there are $\#\mathbb{T}(\hat{y}, \hat{z})$ elements which already appear in the table $\mathbb{T}$. Therefore, the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ returns old x or new x with this ratio. In case of old x, the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ picks old x according to both the current table $\mathbb{T}$ and the number of the preimages of each old z defined in the table $\mathbb{L}_{yz}$. In case of new x, the algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}$ picks new x uniformly at random. The whole algorithm is described in Algorithm 3.

Algorithm $\mathsf{BwO}_{\mathsf{R}}^{\mathsf{Bw}}(\hat{y}, \hat{z})$

1. If $(\hat{y}, \hat{z}, n_{yz}) \notin \mathbb{L}_{yz}$ for any n_{yz}, then pick $n_{yz} \leftarrow \mathsf{B_N}(\#X - \sum_{(\hat{y},\tilde{z},\tilde{n}_{yz})\in\mathbb{L}_{yz}} \tilde{n}_{yz}, \frac{1}{\#Z - \#\mathbb{L}_{yz}(\hat{y})})$, and insert $(\hat{y}, \hat{z}, n_{yz})$ in $\mathbb{L}_{yz}$.
2. If $n_{yz} = 0$ for $(\hat{y}, \hat{z}, n_{yz}) \in \mathbb{L}_{yz}$, then return $\perp$.
3. If $n_{yz} \neq 0$ for $(\hat{y}, \hat{z}, n_{yz}) \in \mathbb{L}_{yz}$, then compute the probability $q = \frac{\#\mathbb{T}(\hat{y},\hat{z})}{n_{yz}}$.
4. With probability q return old x.
   (a) pick one entry uniformly from $\mathbb{T}$ such that $(\tilde{x}, \hat{y}, \hat{z}) \in \mathbb{T}$, and return $\tilde{x}$.
5. Otherwise return new x.
   (a) pick uniformly $x \leftarrow X$ such that $(x, \hat{y}, \tilde{z}) \notin \mathbb{T}$.
   (b) insert $(x, \hat{y}, \hat{z})$ in $\mathbb{T}$, and return x.

Algorithm 3. Simulation method of the backward oracle (Restricted Version)

Hardening the Network from the Friend Within

L. Jean Camp

School of Informatics, Indiana University
ljcamp@indiana.edu

Abstract. The insider threat in the networked world is distinct from the insider threat in the traditional physical business realm in that the most dangerous networked insider may be the least intentionally malicious. This inadvertent enemy within enables access by malicious outsiders through technologically nave or risk-seeking behavior. These behaviors include consistent choices (e.g., permission configurations, monotonically increasing access control rights) and specific behaviors (e.g., opening email attachments, clicking on video links). The risks of these actions are invisible to the individual, and the risks are borne at least in part by the organization. Any change in this insider behavior must include incentives for risk-avoidance, risk communication, and enable risk-mitigating choices. By developing incentive mechanisms and interactions that communicate these incentives, the risk behavior of the authorized insider can be aligned with the risk posture of the organization. We have combined game theory for incentive design, risk parameterization for pricing, and risk communication to create risk-based access control. The presentation will include the game formulation, presentation of the mechanism for pricing behaviors, and the remarkable results of initial human subjects experimentation.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, p. 249, 2009.

Reducing the Complexity in the Distributed Computation of Private RSA Keys

Peter Lory

University of Regensburg, D-93040 Regensburg, Germany
Peter.Lory@wiwi.uni-regensburg.de
http://www.wiwi.uni-regensburg.de/lory/

Abstract. Catalano, Gennaro and Halevi (2000) present a protocol for the distributed computation of inverses over a shared secret modulus. The most important application of their protocol is the distributed computation of the private RSA key from the public key. The protocol is attractive, because it requires only two rounds of communication in the case of honest but curious players. The present paper gives a modification of this protocol, which reduces its complexity from $O(n^3(\log n)^2 + n^2(\log n)(\log N) + (\log N)^2)$ to $O(n^3 \log n + n^2 \log N + (\log N)^2)$ bit-operations per player, where n is the number of players and N is the RSA modulus. The number of communication rounds is the same as in the original protocol.

1 Introduction

Secure distributed computation is attractive, because it avoids the problem of a *single point of failure*. Such an entity would have full control of the system. If it is dishonest, it could misuse this power. Additionally, its server would be an attractive target for malicious adversaries and has to be protected with high costs. The last two decades have seen an exciting development of techniques for secure multiparty computations. Classical theoretical results [1,4,9,17] show that any multiparty computation can be performed securely, if the number of corrupted participants does not exceed certain bounds. However already Gennaro, Rabin and Rabin [8] point out, that these generic secure circuit techniques are too inefficient in the area of practical feasibility, which might render them impractical. Thus, it is a high priority to optimize such techniques.

Catalano, Gennaro and Halevi [3] have presented a distributed inversion protocol, which requires only two rounds of communication. The most interesting application is the distributed computation of the shares of the private RSA exponent d from the public exponent e, if the shares of Euler's phi-funktion $\varphi(N)$ of an RSA modulus N are given (see Rivest, Shamir and Adleman [14]). This has several applications, among which the construction of threshold variants of signature schemes (for a survey see [3]). The protocol in [3] requires $O(n^3(\log n)^2 + n^2(\log n)(\log N) + (\log N)^2)$ bit-operations per player, where n is the number of players and N is the RSA modulus. The present paper reduces this complexity to $O(n^3 \log n + n^2 \log N + (\log N)^2)$. The suggested modified protocol needs also only two rounds of communication. This success is made possible by a loan from the field of numerical analysis, namely by the application of Newton's *Methodus Differentialis*.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 250–263, 2009.

A network of n players is assumed, that are connected by point-to-point private channels and by a broadcast channel. Failures are modeled by an adversary $\mathcal{A}$, who can corrupt at most t of the players in the "honest-but-curious" sense. This means, that the adversary $\mathcal{A}$ can just read the memories of the corrupted players but not modify their behaviour. The protocol is *secure*, i.e. *correct* and *private*. Correctness means that the output values constitute a $(t+1)$-out-of-n secret sharing of the inverse. Privacy is defined using the usual simulation approach (see Section 4).

For the investigation of the time complexities two basic assumptions are made:

a) The addition or subtraction of two k-bit-integers requires $(\rho_{add} \cdot k)$ bit-operations and consequently its bit-complexity is $O(k)$.
b) The multiplication of a k-bit-integer and an l-bit-integer requires $(\rho_{mult} \cdot k \cdot l)$ bit-operations and its bit-complexity is $O(kl)$. This is a reasonable estimate for realistic values.

The concrete values for ρ_{add} and ρ_{mult} are implementation dependent (see e.g. Knuth [11]).

The protocol of Catalano, Gennaro and Halevi refers to polynomial sharing over the integers on the basis of Shamir's seminal $(t+1)$-out-of-n threshold scheme [15]. The latter operates in the field $\mathbb{Z}_q$ with a prime q. Polynomial sharing over $\mathbb{Z}$ gives rise to some technical problems. These are discussed in detail in Catalano [2]. One of the tricks to overcome these problems is the multiplication of the arising equations by $L := n!$. This trick guarantees that the coefficients of the arising polynomials are integers instead of fractions.

The paper is organized as follows: Section 2 presents the protocol of Catalano, Gennaro and Halevi [3] for the reader's convenience and investigates its complexity. Section 3 gives basic material on Newton's scheme of divided differences for reference in Section 4. In this section the new protocol is presented. Section 5 investigates the complexity of the new protocol. Final remarks are given Section 6.

2 The Inversion Protocol of Catalano, Gennaro and Halevi

Catalano, Gennaro and Halevi [3] have presented an efficient protocol to accomplish the distributed computation of an inverse of a publicly known number e over a shared modulus (see also Catalano [2]). For the reader's convenience the protocol is repeated in Figure 1. Here, $L := n!$, where n is the number of players.

In Round 1 of this protocol player P_i chooses integer coefficients of the polynomials $g_i(z)$, $h_i(z)$ and $\rho_i(z)$. Then this player has to evaluate these polynomials at the abscissas $z = j = 1, 2, \ldots, n$. The polynomial $g_i(z)$ is of degree at most t. Its evaluation at $z = j$ by Horner's scheme

$$g_i(j) = (\ldots((b_{i,t}j + b_{i,t-1})j + b_{i,t-2})j + \ldots b_{i,1})j + L\lambda_i \tag{1}$$

requires t multiplications and t additions. Please note, that the protocol employs polynomial sharing over $\mathbb{Z}$. For the first multiplication the operands are integers of at most k_1 bits with $k_1 = \lceil \log(L^2N^3) \rceil \approx \log(L^2N^3)$ and of at most $\lceil \log n \rceil \approx \log n$ bits,

Public input: Prime number e with $e > n$, an appoximate bound N on φ.

Private inputs: Sharing of $L\varphi$ using a polynomial f of degree at most t over the integers. Player P_i has private input $f_i = f(i)$, where $f(z) = L\varphi + a_1 z + \ldots + a_t z^t$ and $a_j \in [-L^2N, L^2N]$ for $j = 1, \ldots, t$.

Round 1: Each player P_i does the following:

1. Choose $\lambda_i \in_R [0, N^2]$ and $b_{i,1}, \ldots, b_{i,t} \in_R [-L^2N^3, L^2N^3]$.
   Choose $r_i \in_R [0, N^3]$ and $c_{i,1}, \ldots, c_{i,t} \in_R [-L^2N^4, L^2N^4]$.
   Choose $\rho_{i,1}, \ldots, \rho_{i,2t} \in_R [-L^2N^5, L^2N^5]$.

2. Set
$$g_i(z) = L\lambda_i + b_{i,1}z + \ldots + b_{i,t}z^t\,,$$
$$h_i(z) = Lr_i + c_{i,1}z + \ldots + c_{i,t}z^t\,,$$
$$\rho_i(z) = 0 + \rho_{i,1}z + \ldots + \rho_{i,2t}z^{2t}\,.$$

3. Send to each player P_j the values $g_i(j), h_i(j), \rho_i(j)$, computed over the integers.

Round 2: Each player P_j does the following:

1. Set
$$g_j = \sum_{i=1}^{n} g_i(j)\,, \quad h_j = \sum_{i=1}^{n} h_i(j)\,, \quad \rho_j = \sum_{i=1}^{n} \rho_i(j)\,.$$
   (These are P_j's shares of the polynomials
$$g(z) = \sum_{i=1}^{n} g_i(z)\,, \quad h(z) = \sum_{i=1}^{n} h_i(z)\,, \quad \rho(z) = \sum_{i=1}^{n} \rho_i(z)\,.)$$

2. Broadcast the value $F_j = f_j g_j + e h_j + \rho_j$.

Output: Each player P_i does the following:

1. From the broadcast values interpolate the $2t$-degree polynomial $F(z) = f(z)g(z) + e \cdot h(z) + \rho(z)$.
2. Using the GCD algorithm, find integers a and b such that $aF(0) + be = 1$.
   If no such integers a and b exist, go to Round 1.
3. The inverse of e is $d = ah(0) + b$.
   Privately output the share of the inverse, $d_i = ah(i) + b$.

Fig. 1. The inversion protocol of Catalano, Gennaro and Halevi

respectively. Here, the symbol $\lceil a \rceil$ denotes the smallest integer b with $b \geq a$. In the following, this sign will be suppressed for readability reasons. The result of this multiplication has $k_1 + \log n$ bits. The following addition adds a k_1-bit-number to this result and delivers an integer of at most $k_1 + \log n + 1$ bits. The next multiplication multiplies this number with a $\log n$-bit-number. Its result has $k_1 + 2\log n + 1$ bits etc. Consequently, the n evaluations of the polynomial $g_i(z)$ require at most

$$n\rho_{mult}\left[tk_1\log n+\frac{t(t-1)}{2}(\log n)(\log n+1)\right]+$$
$$n\rho_{add}\left[tk_1+\frac{t(t+1)}{2}\log n+\frac{t(t-1)}{2}\right]$$

bit-operations, where

$$k_1=\log(L^2N^3)\,. \tag{2}$$

Similarly, the n evaluations of the polynomials $h_i(z)$ and $\rho_i(z)$ require at most

$$n\rho_{mult}\left[tk_2\log n+\frac{t(t-1)}{2}(\log n)(\log n+1)\right]+$$
$$n\rho_{add}\left[tk_2+\frac{t(t+1)}{2}\log n+\frac{t(t-1)}{2}\right]$$

and

$$n\rho_{mult}[2tk_3\log n+t(2t-1)(\log n)(\log n+1)]+$$
$$n\rho_{add}[(2t-1)k_3+t(2t-1)\log n+(2t-1)(t-1)]$$

bit-operations, respectively, where

$$k_2=\log(L^2N^4) \tag{3}$$

and

$$k_3=\log(L^2N^5)\,. \tag{4}$$

Please note that the multiplication $L\cdot\lambda_i$ requires $O(\log L\cdot\log N^2)$ bit-operations. This is small in comparison to $O(n^2\log(L^2N^3)$. A similar comment applies to the product Lr_i . So, the following theorem is proven:

Theorem 1. *Round 1 of the inversion protocol of Figure 1 requires*

$$O(n^2k_1\log n+n^2k_2\log n+n^2k_3\log n+n^3(\log n)^2)$$

bit-operations per player, where k_1, k_2 and k_3 are given in Equations (2) – (4).

3 Newton's Scheme of Divided Differences

Interesting historical remarks on Newton's interpolation formula and on the concept of divided differences can be found in the book of Hairer and Wanner [10]. For later reference, the basic facts following the notation in Stoer and Bulirsch [16] are presented here.

Let the support abscissas x_i and corresponding support ordinates f_i $(0\le i\le m)$ be given. The *divided differences* are defined recursively by

$$f_{i_0,i_1,\ldots,i_l}\overset{def}{=}\frac{f_{i_1,i_2,\ldots,i_l}-f_{i_0,i_1,\ldots,i_{l-1}}}{x_{i_l}-x_{i_0}} \tag{5}$$

and can be arranged in the *divided-difference scheme* (see Figure 2).

The most convenient way of computation is to start with the upper left corner and add successive ascending diagonal rows.

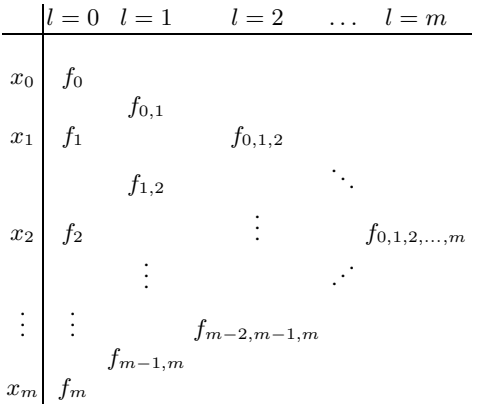


Fig. 2. Newton's divided-difference scheme

4 The New Inversion Protocol

In Round 1 of the protocol of Catalano, Gennaro and Halevi [3] (see Figure 1) each player P_i randomly chooses the coefficients of the polynomials $g_i(z), h_i(z)$ and $\rho_i(z)$ of degree at most t and $2t$, respectively, and then has to evaluate these polynomials at n different points. This can be done more efficiently by using Newton's scheme of divided differences.

Let us have a closer look at the polynomial

$$g_i(z) = L\lambda_i + b_{i,1}z + \ldots + b_{i,t-1}z^{t-1} + b_{i,t}z^t\,.$$

The present paper suggests that instead of choosing the coefficients $b_{i,1}, \ldots, b_{i,t}$, each of the players P_i randomly picks t support ordinates $g_i(1), g_i(2), \ldots g_i(t)$ for the t abscissas $z = 1, z = 2, \ldots, z = t$. This data and the condition

$$g_i(0) = L\lambda_i$$

implicitly define the unique interpolation poynomial $g_i(z)$ of degree at most t. Then player P_i has to evaluate this polynomial for $z = t+1, \ldots, z = n$. Using Newton's scheme of divided differences these computations can be accomplished very efficiently. The basic idea can be applied in other contexts of polynomial sharing, too (cf. the multiparty multiplication of two shared values over $\mathbb{Z}_q$ with a prime q in [12]).

The details are given in Figure 3 with $g_0 = g_i(0), g_1 = g_i(1), \ldots, g_n = g_i(n)$. For readability reasons the index i is omitted. A few remarks are in place:

1. The zeros in the columns $l = t+1, t+2, \ldots, n$ of Figure 3 are not computed. Instead, they are prescribed and force the interpolating polynomial to be of degree at most t.
2. The first $t+1$ ascending diagonal rows are computed from left to right starting at the prescribed support ordinate $g_0 = g_i(0) = L\lambda_i$ and the randomly chosen support ordinates $g_1 = g_i(1), \ldots, g_t = g_i(t)$. The following diagonal rows are computed from right to left starting at the prescribed zeros and ending with the function values

$$g_{t+1} = g_i(t+1), \ldots, g_n = g_i(n)\,.$$

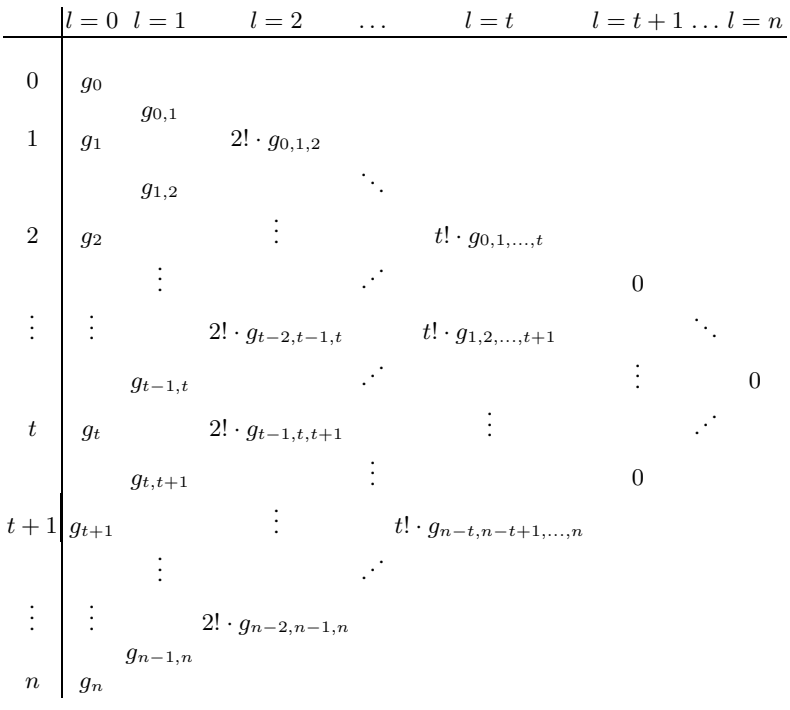


Fig. 3. Newton's divided-difference scheme for the new inversion protocol

3. Instead of calculating the divided differences as defined in Equation (5), the numbers

$$g_{i_0,i_1,\ldots,i_l} \cdot (x_{i_l} - x_{i_0}) = g_{i_1,i_2,\ldots,i_l} - g_{i_0,i_1,\ldots,i_{l-1}} \,. \tag{6}$$

   are computed. This modification is the reason for the factors $l!$ in column l of the scheme of Figure 3 and avoids superfluous arithmetic operations.

The polynomials $h_i(z)$ and $\rho_i(z)$ are treated in the same way. For reasons of consistency the definition of the polynomial $f(z)$ for the polynomial sharing of $L\varphi$ is also modified analogously: Whereas in the private inputs of the protocol of Figure 1 $f(z)$ is defined by its coefficients, it is now implicitly given by its free term $f(0) = L\varphi$ and by the function values $f(1) = f_1, \ldots, f(t) = f_t$. The new protocol is shown in Figure 4.

Step 3 of Round 1 of this protocol is presented in Figure 5. A few comments to this subprotocol are in place:

1. Step (a) calculates the upper left corners in the divided-difference schemes for the two polynomials $g_i(z)$ and $h_i(z)$. For $g_i(z)$ compare Figure 3.
2. Step (b) calculates the following t ascending diagonal rows from left to right in the divided-difference schemes for the two polynomials $g_i(z)$ and $h_i(z)$. For $g_i(z)$ compare again Figure 3.
3. Step (c) calculates the following $n-t$ ascending diagonal rows from right to left in the divided-difference schemes for the two polynomials $g_i(z)$ and $h_i(z)$. For $g_i(z)$ compare again Figure 3.

Public input: Prime number e with $e > n$, an appoximate bound N on φ.

Private inputs: Sharing of $L\varphi$ using a polynomial f of degree at most t. Player P_i has private input $f(i)$. The polynomial f is defined as the unique polynomial of degree at most t with free term $L\varphi$ and the function values $f(j) = f_j$ for $j = 1, \ldots, t$, where the integers $f_j \in [-L^2N, L^2N]$.

Round 1: Each player P_i does the following:

1. Choose an integer $\lambda_i \in_R [0, N^2]$
   and integer function values $g_{i,1}, \ldots, g_{i,t} \in_R [-L^2N^3, L^2N^3]$.
   Choose an integer $r_i \in_R [0, N^3]$
   and integer function values $h_{i,1}, \ldots, h_{i,t} \in_R [-L^2N^4, L^2N^4]$.
   Choose integer function values $\rho_{i,1}, \ldots, \rho_{i,2t} \in_R [-L^6N^5, L^6N^5]$.
2. The polynomial $g_i(z)$ of degree at most t is implicitly defined by its free term $L\lambda_i$ and $g_i(j) = g_{i,j}$ for $j = 1, \ldots, t$.
   The polynomial $h_i(z)$ of degree at most t is implicitly defined by its free term Lr_i and $h_i(j) = h_{i,j}$ for $j = 1, \ldots, t$.
   The polynomial $\rho_i(z)$ of degree at most $2t$ is implicitly defined by its free term 0 and $\rho_i(j) = \rho_{i,j}$ for $j = 1, \ldots, 2t$.
3. Send to each player P_j the values $g_i(j), h_i(j), \rho_i(j)$ for $j = 1, 2, \ldots, n$. The function values $g_i(1), \ldots, g_i(t), h_i(1), \ldots, h_i(t)$ and $\rho_i(1), \ldots, \rho_i(2t)$ are given; the remaining values are computed by the subprotocol of Figure 5.

Round 2 and **Output:** These steps are identical to the corresponding steps in the inversion protocol of Figure 1.

Fig. 4. The new inversion protocol

4. Step (d) calculates the upper left corner in the divided-difference scheme for the polynomial $\rho_i(z)$.
5. Step (e) calculates the following $2t$ ascending diagonal rows from left to right in the divided-difference scheme for the polynomial $\rho_i(z)$.
6. Step (f) calculates the following $n - 2t$ ascending diagonal rows from right to left in the divided-difference scheme for the polynomial $\rho_i(z)$.

It should be noted, that the polynomials $f(z)$, $g_i(z)$, $h_i(z)$ and $\rho_i(z)$ as implicitly defined above (and in the protocol) have integer function values for $z = 0, \ldots, z = n$. Consider, for example, the polynomial $g_i(z)$. The fact that $g_i(z)$ is an integer for $z = 0, \ldots, z = t$ is obvious. For $z = t + 1, \ldots, z = n$ it can be proven as follows: In the corresponding divided-difference scheme of Figure 3 the $g_0, g_1, \ldots, g_t$ in column $l = 0$ are the chosen integers $L\lambda_i, g_{i,1}, \ldots, g_{i,t}$. The entries in the tableau are computed by the recursion formula (6). All these computations involve only subtractions and additions of integers (cf. the subprotocol of Figure 5). Consequently, the $g_i(t+1), g_i(t+2), \ldots, g_i(n)$ are integers as well.

A modular inversion protocol is called *correct* if the output values $d_1, \ldots, d_n$ constitute a $(t+1)$-out-of-n secret sharing of $d = e^{-1} \mod \varphi$. For the definition of privacy, the view of the adversary $\mathcal{A}$ is considered to be the set of messages sent and received by

Player P_i executes the following steps:

(a) $\gamma_0 := L\lambda_i\,, \quad \eta_0 := Lr_i\,.$
(b) For $j = 1, 2, \ldots, t$:
$\gamma_j := g_{i,j}\,, \quad \eta_j := h_{i,j}\,,$
for $k = j-1, j-2, \ldots, 0$:
$\gamma_k := \gamma_{k+1} - \gamma_k\,, \quad \eta_k := \eta_{k+1} - \eta_k\,.$
(c) For $j = t+1, t+2, \ldots, n$:
for $k = 0, 1, \ldots, t-1$:
$\gamma_{k+1} := \gamma_{k+1} + \gamma_k\,, \quad \eta_{k+1} := \eta_{k+1} + \eta_k\,.$
$g_i(j) := \gamma_t\,, \quad h_i(j) := \eta_t\,.$
(d) $\sigma_0 := 0\,.$
(e) For $j = 1, 2, \ldots, 2t$:
$\sigma_j := \rho_{i,j}\,,$
for $k = j-1, j-2, \ldots, 0$:
$\sigma_k := \sigma_{k+1} - \sigma_k\,.$
(f) For $j = 2t+1, 2t+2, \ldots, n$:
for $k = 0, 1, \ldots, 2t-1$:
$\sigma_{k+1} := \sigma_{k+1} + \sigma_k\,.$
$\rho_i(j) := \sigma_{2t}\,.$

Fig. 5. Step 3 in Round 1 of the new inversion protocol

the bad players during a run of the protocol. The modular inversion protocol is called *private* if for any adversary $\mathcal{A}$ there exits a simulator $\mathcal{S}$ that runs an execution of the protocol together with $\mathcal{A}$ and produces for it a view that is indistinguishable from the real one.

Please note that the assumption $N - \varphi = O(\sqrt{N})$ in the following theorem is true for the case we are interested in, where N is an RSA modulus and $\varphi = \varphi(N)$.

Theorem 2. *Let $N - \varphi = O(\sqrt{N})$. If all the players carry out the prescribed protocol and $n > 2t$, then the protocol of Figure 4 is secure, i.e. correct and private according to the above definitions.*

Proof: The following proof sketch modifies the corresponding proof in [2] (Theorem 4) and [3] (Theorem 1).

Initial inputs. First it has to be proven that the players start from a $(t+1)$-out-of-n secret sharing of the value φ. It is shown that t players have no information about φ by demonstrating that the distribution of t shares of the secret $L\varphi$ with the polynomial f is statistically indistinguishable from the distribution of t shares that result from sharing the value LN via a polynomial $\hat{f}$.

For this purpose it is proven that with very high probability there is a sharing of LN using a polynomial $\hat{f}$ with integer function values in the same range as f such that $\hat{f}(j_k) = f(j_k)$ for $k = 1, 2, \ldots, t$, where $j_1, \ldots, j_t$ with

$$1 \leq j_1 < j_2 < \ldots < j_t \leq n$$

are the indices of t arbitrary shares. Thus, $\hat{f}$ is defined as the unique polynomial of degree at most t with the free term LN and the function values $\hat{f}(j_k) = f(j_k)$ for $k = 1, 2, \ldots, t$. In the following it is shown that $\hat{f}$ is in the legal range, i.e. the function values $\hat{f}(1), \ldots, \hat{f}(t)$ are integers and lie with very high probability in the range $[-L^2N, L^2N]$. For this purpose the polynomial χ is defined as $\chi(z) := f(z) - \hat{f}(z)$. This is a polynomial of degree at most t and obeys $\chi(0) = L(\varphi - N)$ and $\chi(j_1) = \ldots = \chi(j_t) = 0$. Consequently,

$$\chi(z) = L(\varphi - N) \prod_{k=1}^{t} \frac{z - j_k}{-j_k}$$

and

$$\chi(i) = L(\varphi - N) \prod_{k=1}^{t} \frac{j_k - i}{j_k} \quad \text{for } i = 1, \ldots, t\,. \tag{7}$$

Since $L = n!$, the values above are integers. As $\hat{f}(z) = f(z) - \chi(z)$, the polynomial $\hat{f}$ has free term LN and its function values $\hat{f}(i)$ are integers. An elementary calculation shows that

$$\left| \prod_{k=1}^{t} \frac{j_k - i}{j_k} \right| \leq t! \quad \text{for } i = 1, \ldots, t\,. \tag{8}$$

Because of Equations (7) and (8) the absolute values of $\chi(i)$ can be bounded by

$$|\chi(i)| \leq |\varphi - N| \cdot L \cdot t! \quad \text{for } i = 1, \ldots, t\,.$$

Consequently, the function values $\hat{f}(1), \ldots, \hat{f}(t)$ are in the range

$$\left[-L^2N - \kappa L^2\sqrt{N}, L^2N + \kappa L^2\sqrt{N}\right]$$

and the probability that these function values are outside the legal range is bounded by

$$t\frac{2\kappa L^2\sqrt{N}}{2(L^2N + \kappa L^2\sqrt{N})} \leq O\left(\frac{t}{\sqrt{N}}\right)$$

which is negligible.

Correctness. It is easy to see that the protocol computes the correct output. Since all players are honest, the interpolation at step 1 of the last round will give as the unique polynomial $F(z)$ a polynomial with integer function values. Thus $F(0) = L^2\lambda\varphi + LRe$ with $\lambda = \sum_i \lambda_i$ and $R = \sum_i r_i$ is an integer and its GCD with respect to e can be computed. If e does not divide φ, the probability that $GCD(e, F(0)) \neq 1$ is roughly $1/e$ (i. e. the probability that e divides λ). Thus, it is unlikely that the protocol has to be repeated more than once. When $aF(0) + be = 1$ is obtained, it can be re-written as

$$a(L^2\lambda\varphi + LRe) + be = 1\,.$$

Taken $\mod \varphi$ the last equation becomes $(aLR+b)e = 1 \mod \varphi$. This means that $d = aLR+b = e^{-1} \mod \varphi$. Thus, the t-degree polynomial $ah(z)+b$ interpolates to the correct value d and the shares d_i correctly lie on this polynomial.

Simulation of the inversion protocol. The simulator controls the players $P_{j_{t+1}}, \ldots, P_{j_n}$. For these players it holds initial values $\hat{f}_i$, which result from a sharing of LN (instead of $L\varphi$ as discussed above).

For Round 1 the simulator simply follows the same instructions as the protocol. This results in shared polynomials $\hat{g}(z)$, $\hat{h}(z)$ and $\hat{\rho}(z)$ and shared values $L\hat{\lambda} = \hat{g}(0)$ and $L\hat{R} = \hat{h}(0)$. Obviously $\hat{\lambda}$ and $\hat{R}$ follow the same distribution as λ and R. Moreover notice that an argument very similar to the one used for the sharing of the initial values shows that the adversary has no information about $\hat{\lambda}$ and $\hat{R}$.

During Round 2 the simulator publishes the values $\hat{F}(i) = \hat{f}(i)\hat{g}(i) + e\hat{h}(i) + \hat{\rho}(i)$ for $i = j_{t+1}, \ldots, j_n$. The function values f_i are in the range $[-L^2N, L^2N]$ for $i = 0, \ldots, t$. Because of the Lagrange interpolation formula

$$f(j) = \sum_{i=0}^{t} f_i \prod_{\substack{k \neq i \\ k=0\ldots t}} \frac{j-k}{i-k}.$$

An elementary calculation shows that

$$\left| \prod_{\substack{k \neq i \\ k=0\ldots t}} \frac{j-k}{i-k} \right| \leq L \quad \text{for } i = 0, \ldots, t \text{ and } j = t+1, \ldots, n.$$

Thus $|f(i)|$ and (using an analogous argument) also $|\hat{f}(i)|$ are bounded by nL^3N for $i = 1, \ldots, n$. Similarly it can be shown that $|g(i)|$ and $|\hat{g}(i)|$ are bounded by $n^2L^3N^3$. As a consequence the function values $\rho(i)$ and $\hat{\rho}(i)$ for $i = 1, \ldots, n$ are much larger than the corresponding function values for $f(z)g(z)$ and $\hat{f}(z)\hat{g}(z)$. Thus both polynomials $F(z)$ and $\hat{F}(z)$ follow a distribution which is statistically close to $\rho(z)$, except for the free term.

The $2t$-degree polynomial $\hat{F}(z)$ has the free term $L^2\hat{\lambda}N + L\hat{R}e$ (while in the real execution it interpolates to $L^2\lambda\varphi + LRe$). It is shown in Lemma 1 of [2] and [3] that the distributions of these two values are statistically close. □

5 Complexity of the New Inversion Protocol

In all the steps of the subprotocol of Figure 5 only additions and subtractions occur. Each of these operations may increase the bit-length by 1. Consequently, step (b) needs $t(t+1)/2$ subtractions of two numbers with at most $k_1 + t$ bits and of two numbers with at most $k_2 + t$ bits, where $k_1 = \log(L^2N^3)$ and $k_2 = \log(L^2N^4)$. Therefore, this step requires at most

$$\frac{t(t+1)}{2}\rho_{add}(k_1 + k_2 + 2t)$$

bit-operations. The maximum bit-sizes in step (c) are $k_1 + t + n - 1$ and $k_2 + t + n - 1$, respectively. Consequently, this step requires at most

$$(n-t)t\rho_{add}(k_1 + k_2 + 2t + 2n - 2)$$

bit-operations. As above, n is the number of players and t is the threshold. Step (e) needs at most

$$t(2t+1)\rho_{add}(k_3' + 2t)$$

bit-operations, where

$$k_3' = \log(L^6 N^5)\,. \tag{9}$$

Finally, step (f) requires at most

$$(n-2t)2t\rho_{add}(k_3' + 2t + n - 1)$$

bit-operations. Taking into account that $t < 2t + 1 \leq n$, the following theorem is proven:

Theorem 3. *Round 1 of the new inversion protocol of Figure 4 requires*

$$O(n^2 k_1 + n^2 k_2 + n^2 k_3' + n^3) \tag{10}$$

bit-operations per player, where k_1, k_2 and k_3' are given in Equations (2), (3) and (9).

A comparison of the above result with Theorem 1 shows the reduction of the complexity. Please note that Equation (10) covers the worst case. In reality the bit sizes of the numbers γ_k, η_k and σ_k will both increase and decrease during steps (b), (c), (e) and (f) of Figure 5 and in the average case the bit-sizes will approximately remain constant. So in the average case the last term in Equation (10) may be neglected.

In view of the complete protocol it remains to investigate the complexities of Round 2 and the Output, which are identical in the protocol of Figure 1 and in the new protocol of Figure 4. The complexity of Round 2 can be neglected. The computation of $F(0)$ in step 1 of the Output in the protocols of Figures 1 and 4 can again be done efficiently using a scheme of divided differences similar to that of Figure 3.

Now player P_i possesses the values $F_1, F_2, \ldots, F_{2t+1}$ and wants to compute $F_0 = F(0)$. These values define the first column of the scheme. First the triangle with the base $F_1, \ldots, F_{2t+1}$ is successively computed building the $2t + 1$ corresponding ascending diagonal rows from left to right starting at the prescribed support ordinates $F_1, \ldots, F_{2t+1}$ and ending with the values $F_1, F_{1,2}, \ldots, (2t)! \cdot F_{1,2,\ldots,2t+1}$. Then the uppermost descending diagonal row including $F_0 = F(0)$ is computed from right to left starting at the prescribed 0 in column $2t + 1$ and using the already computed values $(2t)! \cdot F_{1,2,\ldots,2t+1}, \ldots, F_{1,2}, F_1$ in this order. These ideas are the basis of the new algorithm given in Figure 6.

Analogously to the arguments at the beginning of this section, it follows that the protocol of Figure 6 requires at most

$$t(2t+1)\rho_{add}(k_4 + 2t) + 2t\rho_{add}(k_4 + 4t) \tag{11}$$

(a) For $j = 1, 2, \ldots, 2t+1$:
$\gamma_j := F_j$,
for $k = j-1, j-2, \ldots, 1$:
$\gamma_k := \gamma_{k+1} - \gamma_k$,
$\chi_j := \gamma_1$.
(b) For $k = 2t, 2t-1, \ldots, 1$:
$\chi_k := \chi_k - \chi_{k+1}$,
$F_0 := \chi_1$.

Fig. 6. Efficient computation of $F(0) = F_0$

i.e. $O(n^2 k_4 + n^3)$ bit-operations, where k_4 is the bit-size of the function values $F(1)$, $F(2)$, ..., $F(2t+1)$. A simple calculation shows that $k_4 \leq k_3' + 5n$ and consequently the operation count of Equation (11) is $O(n^2 k_3' + n^3)$. In the case of the original protocol the bit-sizes of $F(1)$, $F(2)$, ..., $F(2t+1)$ are slightly different and the corresponding esimate is $k_4 \leq k_3 + 3n \log n + 2n$. This results in a corresponding operation complexity of $O(n^2 k_3 + n^3 \log n)$.

The GCD algorithm in step 2 of the Output requires $O((\log U)^2)$ bit-operations if U is an upper bound of the nonnegative integers e and $F(0)$ (see e.g. Cohen [5] or Mao [13]). It is pointed out in the proof of Theorem 2 that if e does not divide φ, the probability that $\mathrm{GCD}(e, F(0)) \neq 1$ is roughly $1/e$. Thus it is very unlikely that the rounds of the protocol need to be repeated several times. Finally,

$$F(0) \leq nL^2N^3 + nLN^4 \leq 2nL^2N^4 .$$

These results together with Theorems 1 and 3 can be summarized in the following theorem:

Theorem 4. *The new inversion protocol of Figure 4 reduces the complexity from*

$$O(n^3(\log n)^2 + n^2(\log n)(\log N) + (\log N)^2) \tag{12}$$

to

$$O(n^3 \log n + n^2 \log N + (\log N)^2) \tag{13}$$

bit-operations per player, where n is the number of players and N is the RSA modulus.

In the following table the two functions $A(n, N) = n^3(\log n)^2 + n^2(\log n)(\log N) + (\log N)^2$ (cf. Equation (12)) and $B(n, N) = n^3 \log n + n^2 \log N + (\log N)^2$ (cf. Equation (13)) are compared for realistic values of N $(= 2^{1024})$ and the number n of players:

n	$A(n, 2^{1024})$	$B(n, 2^{1024})$	$A(n, 2^{1024})/B(n, 2^{1024})$
2^5	$7.11 \cdot 10^6$	$2.26 \cdot 10^6$	3.15
2^6	$3.57 \cdot 10^7$	$6.82 \cdot 10^6$	5.23
2^7	$2.21 \cdot 10^8$	$3.25 \cdot 10^7$	6.80
2^8	$1.61 \cdot 10^9$	$2.02 \cdot 10^8$	7.97

The table indicates that for an increasing albeit realistic number n of players the new protocol accelerates the original one by a factor of approximately $\log n$. For smaller

values of n the improvement is less remarkable because there is no reduction in the $(\log N)^2$-Term, which occurs both in (12) and in (13). Both protocols require (with very high probability) only two rounds of communication.

6 Conclusion

The present paper reduces the complexity in the protocol in [3] for the distributed computation of inverses over a shared modulus by a new technique, which applies Newton's divided-difference scheme. This efficiency improvement is particularly important for threshold variants (see [3]) of the signature schemes of Cramer and Shoup [6] and of Gennaro, Halevi and Rabin [7]. In these signature shemes the inversion operation is performed with a different RSA exponent e for each message signed.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th Annual Symposium on Theory of Computing (STOC 1988), pp. 1–10. ACM Press, New York (1988)
2. Catalano, D.: Efficient distributed computation modulo a shared secret. In: Catalano, D., Cramer, R., Damgård, I., Di Crescenco, G., Pointcheval, D., Takagi, T. (eds.) Contemporary Cryptology, CRM Barcelona. Advanced Courses in Mathematics, pp. 1–39. Birkhäuser, Basel (2005)
3. Catalano, D., Gennaro, R., Halevi, S.: Computing inverses over a shared secret modulus. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 190–206. Springer, Heidelberg (2000)
4. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols. In: Proceedings of the 20th Annual Symposium on Theory of Computing (STOC 1988), pp. 11–19. ACM Press, New York (1988)
5. Cohen, H.: A Course in Computational Algebraic Number Theory. Springer, Berlin (2000)
6. Cramer, R., Shoup, V.: Signature schemes based on the Strong RSA Assumption. ACM Transactions on Information and System Security (ACM TISSEC) 3(3), 161–185 (2000)
7. Gennaro, R., Halevi, S., Rabin, T.: Secure hash-and-sign signatures without the random oracle. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 123–139. Springer, Heidelberg (1999)
8. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified VSS and fast-track multiparty computations with applications to threshold cryptography. In: Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC 1998), pp. 101–111. ACM Press, New York (1998)
9. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the 19th Annual Symposium on Theory of Computing (STOC 1987), pp. 218–229. ACM Press, New York (1987)
10. Hairer, E., Wanner, G.: Analysis by Its History. Springer, New York (1995)
11. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1971)
12. Lory, P.: Reducing the complexity in the distributed multiplication protocol of two polynomially shared values. In: Proceedings of the 3rd IEEE International Symosium on Security in Networks and Distributed Systems (SSNDS 2007). AINA 2007, vol. 1, pp. 404–408. IEEE Computer Society Press, Los Alamitos (2007)

13. Mao, W.: Modern Cryptography: Theory and Practice. Prentice Hall, Upper Saddle River (2004)
14. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
15. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
16. Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis. Springer, Berlin (2002)
17. Yao, A.C.: How to generate and exchange secrets. In: Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS 1986), pp. 162–167. IEEE Computer Society, Los Alamitos (1986)

Efficiency Bounds for Adversary Constructions in Black-Box Reductions

Ahto Buldas[1,2,3,⋆], Aivo Jürgenson[2,4], and Margus Niitsoo[1,3]

[1] Cybernetica AS, Akadeemia tee 21, 12618 Tallinn, Estonia
[2] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia
[3] University of Tartu, J.Liivi 2, 50409 Tartu, Estonia
[4] Elion Enterprises Ltd, Endla 16, 15033 Tallinn, Estonia
Ahto.Buldas@ut.ee, Aivo.Jurgenson@elion.ee, Margus.Niitsoo@ut.ee

Abstract. We establish a framework for bounding the efficiency of cryptographic reductions in terms of their security transfer. While efficiency bounds for the reductions have been studied for about ten years, the main focus has been the efficiency of the *construction* mostly measured by the number of calls to the basic primitive by the constructed primitive. Our work focuses on the efficiency of the *wrapper* construction that builds an adversary for the basic primitive and has black-box access to an adversary for the constructed primitive. We present and prove a general upper bound theorem for the efficiency of black-box reductions. We also provide an example about upper bound for reductions between two security notions of cryptographic hash functions, which gives a negative answer to the open question about the existence of linear-preserving reductions from the so-called *hash-then-publish time-stamping schemes* to the collision resistance of the underlying hash function.

1 Introduction

The security of cryptographic schemes is usually proved based on certain assumptions. For instance, it is a well known fact that the ElGamal cryptosystem is secure only if the Diffie-Hellman assumption holds. Assumptions used can be very different, some being very specific computational problems and others being far more abstract in nature. The need to use assumptions has created a rather interesting subfield of cryptology that concerns itself with the theoretical bounds of their use. Much research has been put into studying what exactly can be constructed using, for instance, one-way permutations. More interestingly, what cannot be constructed from them has also been studied rather extensively.

Research in that direction started with the seminal paper of Impagliazzo and Rudich [7], who used the Oracle separation method from Complexity theory to show that a black-box construction from one-way permutations to secret agreement would imply a contradiction and thus concluded that such a construction

⋆ This research was supported by the European Regional Development Fund through the Estonian Center of Excellence in Computer Science (EXCS), by Estonian SF grant no. 6944, and by EU FP6-15964: "AEOLUS".

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 264–275, 2009.

cannot exist. Their result shows that any construction for a secret agreement would need more than just black-box access to a secure one-way permutation in order to be secure. While this approach does not completely rule out the existence of a construction for secret agreement from one-way permutations, it does show that the reduction would need some additional information about the permutations. As general constructions of that type are nearly unknown, their result is taken as strong evidence that a sensible construction cannot exist.

To date, the connections between the most basic assumptions and most important primitives have been mapped out. Sadly, a review article of these results has yet to be written, although most important results are covered by [7,5]. As a brief overview, there is one equivalence class for private-key cryptology that contains symmetric encryption, signature schemes and bit commitment and is equivalent to assuming the existence of a one-way function. For public key primitives things are a bit more complicated but the existence of trapdoor one-way permutations allows one to construct just about everything and key agreement seems to be constructible from every other public key primitive. Since it is known that key agreement cannot be based on one-way permutations, this implies that private-key cryptology is not enough to construct any public-key primitives. Many more non-reducibility results are known.

With the reductions more or less mapped out with existence or non-existence results, attention was directed towards the efficiency of the reductions. Kim, Simon and Tetali [8] showed that a construction for a universal one-way hash function based on a one-way permutation needs to call that permutation at least $\Omega(\sqrt{n/p(n)})$ times to construct a secure hash function $h : \{0,1\}^n \to \{0,1\}^{(1-\epsilon)n}$ with security $2^{-p(n)}$. Their bound was later strengthened by Gennaro and Trevisan [6] to $\Omega(n/p(n))$ that matches the best known construction. As much more efficient constructions are known for certain computational assumptions, these results can be interpreted as showing that although one-way permutations can be used to construct universal one-way hash functions in theory, it is not reasonable to do so in practice. Many other results in the same direction [3,4] seem to verify that intuition. Therefore, these are still infeasibility results, only of a weaker type.

Our approach, however, goes to a different direction by studying the efficiency of the security reduction instead of the construction used for the new primitive. While bounds on construction efficiency allow one to show just how inefficient the new primitive has to be, bounds on the security part allow one to show just how much assumed security has to be lost when converting from one primitive to the other. In effect, we show that whenever an adversary to the new construction exists, any construction that converts that adversary to one for the original primitive has to do so at the cost of its adversarial efficiency measured by the time/success ratio. As such, we use a definition that is a refinement of poly-preserving reductions defined by Luby [9].

However, to state those results, we first need to introduce some new notations. We begin by defining power c-secure reductions, in which we assume we know how the time-advantage ratio of an adversary to the newly constructed

primitive can be related to a time-success ratio of an attack against the underlying assumption. We note that this is the case for nearly all the positive reduction results known in modern cryptology. Furthermore, this definition allows us to use the oracle separation method to construct a framework for proving the lower bounds we desire. This article aims to provide a basis for further work on finding actual lower bounds on reduction efficiency and as such is mainly concerned with constructing the required tools for future work. However, to show how this approach can indeed be used, we also give two examples on how the theorem we provide can be used to obtain actual lower bounds.

2 Preliminaries and Notation

For bit-strings a and b we define $a\|b$ as their concatenation. By $x \leftarrow \mathcal{D}$ we mean that x is chosen randomly according to a distribution $\mathcal{D}$. If A is a probabilistic function or a Turing machine, then $x \leftarrow \mathsf{A}(y)$ means that x is chosen according to the output distribution of A on an input y. By $\mathcal{U}_n$ we denote the uniform distribution on $\{0,1\}^n$. If $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are distributions and $F(x_1, \ldots, x_m)$ is a predicate, then $\mathsf{Pr}\left[x_1 \leftarrow \mathcal{D}_1, \ldots, x_m \leftarrow \mathcal{D}_m : F(x_1, \ldots, x_m)\right]$ denotes the probability that $F(x_1, \ldots, x_m)$ is true after the ordered assignment of $x_1, \ldots, x_m$. For functions $f, g\colon \mathbb{N} \to \mathbb{R}$, we write $f(k) = O(g(k))$ $[f(k) = \Omega(g(k))]$ if there is $c \in \mathbb{R}$, so that $f(k) \leq cg(k)$ $[f(k) \geq cg(k)]$ for sufficiently large k. We write $f(k) = \Theta(g(k)$ if $f(k) = O(g(k))$ and $f(k) = \Omega(g(k))$. We write $f(k) = \omega(g(k))$ if $\lim\limits_{k\to\infty} \frac{g(k)}{f(k)} = 0$ and $f(k) = o(g(k))$ if $g(k) = \omega(f(k))$. If $f(k) = k^{-\omega(1)}$, then f is *negligible*. A Turing machine M is *polynomial-time* (*poly-time*) if it runs in time $k^{O(1)}$, where k denotes the input size. By an *oracle Turing machine* we mean an incompletely specified Turing machine S that comprises calls to *oracles*. The description can be completed by defining the oracle as a function $\mathcal{O}\colon \{0,1\}^* \to \{0,1\}^*$. In this case, the machine is denoted by $S^{\mathcal{O}}$. The function $y \leftarrow \mathcal{O}(x)$ does not have to be computable but has a conditional running time $t(x)$, which does not necessarily reflect the actual amount of computations needed to produce y from x. The running time of $S^{\mathcal{O}}$ comprises the conditional running time of oracle calls where we assume that each call $\mathcal{O}(x)$ takes $t(x)$ steps. An oracle $\mathcal{O}$ is *poly-time* if $t(x) = |x|^{O(1)}$, where $|x|$ denotes the bit-length of x. We say that S is a *poly-time oracle machine* if $S^{\mathcal{O}}$ runs in poly-time, whenever $\mathcal{O}$ is poly-time. By a *non-uniform* poly-time oracle machine we mean an ordinary poly-time oracle machine S together with a family $\mathsf{A} = \{a_k\}_{k\in\mathbb{N}}$ of (advice) bit-strings a_k with length $k^{O(1)}$. For any oracle $\mathcal{O}$ and any input x it is assumed that $S^{\mathcal{O}}(x)$ has access to the advice string $a_{|x|}$. Usually, the advice strings are omitted for simplicity, but their presence must always be assumed when S is non-uniform. In the following we do not use the non-uniformity assumption anywhere and as such can rather safely assume that the machines can be non-uniform. We also define $\textsc{Time}_k(\mathsf{A}, f)$ as the expected running time of A on breaking $f_k \in f$ where the expectation is taken over all the possible randomness of A. Analogously, we define $\textsc{Adv}_k(\mathsf{A}, f)$ as the average success probability of A in breaking $f_k \in f$.

3 Cryptographic Reductions

Reductions have been one of the main tools of cryptologists since the beginning of the modern era in the field. At first, reductions were just used to show that breaking the security is as hard as some given computational problem and thus, if we assume that the computational problem is intractable, we have a secure system. However, as the field of cryptology expanded, reductions found use in many other applications as well. Most importantly, they were the main tool used in mapping out the limits of what can be constructed from the most general assumptions such as the existence of a one-way function.

As reductions turned out to be a very effective tool, people soon took interest in exploring the limits of their use. Inspired by the results in complexity theory obtained by Baker, Gill and Solovay [1], Impagliazzo and Rudich [7] were the first to formally define a so-called black-box cryptographic reduction, with the only aim of proving that such a thing cannot exist between secret agreement and one-way permutations.

In essence, a black-box construction of one primitive from another means that the constructed primitive is allowed to call the underlying primitive but is unable to gather any extra knowledge about it by any other means. This notion might seem a little restrictive at first, but in fact most of the reductions used in cryptology even to date still fit that model. However, even more general reduction hierarchy was introduced by Reingold et al. [10].

We use a somewhat informal approach in defining a primitive by saying that a primitive is a security criterion on some functions along with all the functions on which that criterion makes sense. For example, the security criterion for a *one-way function* $f\colon \{0,1\}^* \to \{0,1\}^*$ states that for every probabilistic poly-time adversary A:

$$\mathrm{Adv}_k(\mathsf{A}) = \mathsf{Pr}\left[x \leftarrow \{0,1\}^k, x' \leftarrow \mathsf{A}(f(x))\colon f(x') = f(x)\right] = k^{-\omega(1)} \ .$$

The primitive of one-way functions is this criterion together with the class of all functions of type $f : \{0,1\}^* \to \{0,1\}^*$ some of which might not be secure instances. For a fully formal approach, see Reingold et al. [10].

A black-box reduction of a primitive $\mathcal{P}$ to a primitive $\mathcal{Q}$ can be formalized as two oracle Turing machines S and P such that if $f \in \mathcal{Q}$ then $\mathsf{P}^f \in \mathcal{P}$ and if A breaks the security criterion of $\mathcal{P}$ for P^f then then $\mathsf{S}^{\mathsf{A},f}$ breaks the security criterion of $\mathcal{Q}$ for f. The definition seems somewhat complex but the idea behind it is actually rather simple – essentially it says is that a reduction has to provide an algorithmic means of constructing an instance of the new primitive with the help of the old (P) and that if that new construction is insecure then the original underlying primitive instance also has to be so as we can construct an adversary against it with S.

4 Power *c* Secure Reductions

Thus far, the construction P has been the main object of study in terms of efficiency. We, however, study the efficiency of S instead. While the efficiency

of P tells us just how effectively the new construction can be implemented, the efficiency of S actually tells us how much security we can guarantee for the new primitive on the assumption that the old primitive is secure in some sense.

Suppose, for instance, that the most efficient adversary for the original primitive always takes at least 1000 steps to break it. Now, suppose we know that S construction calls the adversary for the new construction 10 times to break the old construction. This means that any adversary for the new construction has to make at least 100 steps on each call as otherwise we could use the good adversary in the construction S to create an adversary for the original primitive that works in less than 1000 steps. This informal[1] argument should convince the reader that the efficiency of the construction S is indeed important in determining how secure we can prove the new construction.

However, the number of steps the adversary takes to break a primitive is usually not the best measure for its efficiency. Practical adversaries are usually probabilistic and are not guaranteed to always break the primitive. This implies that it usually makes more sense to define the efficiency of the adversary to be its time/success ratio (the expected number of steps it takes on a run divided by the probability of success), which can be interpreted as the expected time to break the primitive if you just kept on running the adversary until it finally succeeds. This motivates the following definition:

Definition 1. *Let $c > 0$ be a positive real number. A* Power c-secure fully black-box reduction *from primitive $\mathcal{P}$ to primitive $\mathcal{Q}$ is a pair (P, S) of poly-time oracle machines, satisfying the following two conditions:*

1. *For any function f that implements $\mathcal{Q}$, the function P^f implements $\mathcal{P}$.*
2. *For any pair (A, f) of functions we have*

$$\frac{\textsc{Time}_k(\mathsf{S}^{\mathsf{A},f}, f)}{\textsc{Adv}_k(\mathsf{S}^{\mathsf{A},f}, f)} \le k^{O(1)} \cdot \left[\frac{\textsc{Time}_k(\mathsf{A}, \mathsf{P}^f)}{\textsc{Adv}_k(\mathsf{A}, \mathsf{P}^f)}\right]^c .$$

The reduction is uniform*, if both P and S are uniform, and* non-uniform *if both of them are allowed to be non-uniform.*

As said before, it is generally assumed that the adversary construction knows nothing about the oracles that are given to it save for their basic syntactic functionality and the previous definition captures that formally as well. In some practical reductions, however, it is assumed that the construction S can actually be dependent on the success probability $\delta = \textsc{Adv}_k(\mathsf{A}, \mathsf{P}^f)$ of A. For example, choosing the number of iterations of A when constructing strong one-way functions from weak ones [9] uses that knowledge in an essential way. Many other reductions also use this extra information and as such cannot be considered fully black-box anymore. As it turns out, however, allowing for such knowledge causes no significant theoretical problems and so we alter our original definition slightly to obtain:

[1] Informal to the point of being incorrect unless we assume the adversaries always make the same number of steps independent of the input. This numerical example is here just to provide intuition to the formalization.

Definition 2. *We say that there exists a power c-secure* success-specific *black-box reduction from primitive $\mathcal{P}$ to primitive $\mathcal{Q}$ iff there is a poly-time oracle machine P and a polynomial p such that for every $\delta > 0$ there is a poly-time oracle machine S_δ so that for all A such that $\mathrm{Adv}_k(\mathsf{A}, \mathsf{P}^f) \geq \delta$ the following conditions hold:*

1. *For any function f that implements $\mathcal{Q}$, the function P^f implements $\mathcal{P}$.*
2. *For any pair (A, f) of functions we have*

$$\frac{\mathrm{Time}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f)}{\mathrm{Adv}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f)} \leq p(k) \cdot \left[\frac{\mathrm{Time}_k(\mathsf{A}, \mathsf{P}^f)}{\mathrm{Adv}_k(\mathsf{A}, \mathsf{P}^f)}\right]^c \quad .$$

for sufficiently large values of k.

Note that black-box reductions that satisfy Def. 2 for any certain (unspecified) c were first defined by Luby [9] and were called *polynomial-preserving reductions* and the reductions with $c = 1$ were called *linear-preserving*.

5 The Lower Bound Theorem

We are now ready to prove the theorem that gives the means of proving the lower bounds described in the introduction.

Theorem 1. *If for every pair (S, P) of poly-time oracle machines and for every $\delta > 0$ there is a probability distribution $(\mathsf{A}, f) \leftarrow \Omega_{\mathsf{S},\mathsf{P},\delta}$ so that:*

- *f implements $\mathcal{Q}$ and P^f implements $\mathcal{P}$ for every (A, f) in the range of $\Omega_{\mathsf{S},\mathsf{P},\delta}$;*
- *$\mathrm{Adv}_k(\mathsf{A}, \mathsf{P}^f) = \delta$ and $\mathrm{Time}_k(\mathsf{A}, \mathsf{P}^f) = O(k^{c_0})$ for some c_0 for all (A, f) in the range of $\Omega_{\mathsf{S},\mathsf{P},\delta}$;*
- *for every polynomial $q(k)$ there exists $\delta(k)$ such that $\lim_{k\to\infty} \delta(k) = 0$ and:*

$$\lim_{k\to\infty} \frac{\delta(k)^c}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A},f)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta(k)}} \left[\mathrm{Time}_k(\mathsf{S}^{\mathsf{A},f}, f)\right]}{\mathop{\mathsf{E}}_{(\mathsf{A},f)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta(k)}} \left[\mathrm{Adv}_k(\mathsf{S}^{\mathsf{A},f}, f)\right]} > 1 \quad ,$$

then there are no power c-secure success-specific black-box reductions of $\mathcal{P}$ to $\mathcal{Q}$.

Proof. Assume to the contrary that there exists such a reduction $(\mathsf{S}_\delta, \mathsf{P})$. Then by the assumptions there exists a polynomial $p(k)$ so that for every $\delta > 0$ and for every pair (A, f) of functions such that $\mathrm{Adv}_k(\mathsf{A}, \mathsf{P}^f) \geq \delta$ we have

$$\frac{\mathrm{Time}_k(\mathsf{S}_\delta^{A,f}, f)}{\mathrm{Adv}_k(\mathsf{S}_\delta^{A,f}, f)} \leq p(k) \cdot \left(\frac{\mathrm{Time}_k(\mathsf{A}, \mathsf{P}^f)}{\delta}\right)^c \tag{1}$$

for large k. This implies $\delta^c \,\mathrm{Time}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f) \leq p(k)t^c(k)\,\mathrm{Adv}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f)$, where $t(k) = \mathrm{Time}_k(\mathsf{A}, \mathsf{P}^f)$. Let $\Omega_{\mathsf{S},\mathsf{P},\delta}$ be the distribution guaranteed by the assumption of the theorem. Then,

$$\delta^c \mathop{\mathsf{E}}_{\mathsf{A},f}\left[\mathrm{Time}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f)\right] \leq p(k)t^c \mathop{\mathsf{E}}_{\mathsf{A},f}\left[\mathrm{Adv}_k(\mathsf{S}_\delta^{\mathsf{A},f}, f)\right] \quad ,$$

Where $(\mathsf{A}, f) \leftarrow \Omega_{\mathsf{S},\mathsf{P},\delta}$. Let $q(k)$ denote a polynomial upper bound for $p(k)t^c(k)$ (that exists because $t(k) = O(k^{c_0})$ for all A). Since the preceding inequality holds for all δ, we also have

$$\lim_{k\to\infty} \frac{\delta(k)^c}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{\mathsf{A},f}\left[\textsc{Time}_k(\mathsf{S}^{\mathsf{A},f}_{\delta(k)}, f)\right]}{\mathop{\mathsf{E}}_{\mathsf{A},f}\left[\textsc{Adv}_k(\mathsf{S}^{\mathsf{A},f}_{\delta(k)}, f)\right]} \le 1$$

for every possible $\delta(k)$ such that $\lim_{k\to\infty} \delta(k) = 0$, a contradiction. □

6 A Simple Example

As an example of the use of this theorem we give a proof that any reduction from collision-resistant functions to one-way functions has to be at least linear[2]. Although the fact itself is rather straightforward, this proof serves as a toy example to introduce the technical complexities of using Theorem 1. However, we first need to introduce some definitions:

Definition 3. *By a* cryptographic (2-1) hash function *we mean a function family* $h_k\colon \{0,1\}^{p(k)} \times \{0,1\}^{2k} \to \{0,1\}^k$ *where* $p(k) > k$ *is a polynomial. The first argument is the so-called function index which is mostly chosen uniformly at random.*

Definition 4. *A cryptographic 2-1 hash function* $h = \{h_k\}$ *is said to be* collision resistant *if for every poly-time adversary* A*:*

$$\begin{aligned}\textsc{Adv}_k(\mathsf{A}) &= \mathsf{Pr}\left[r \leftarrow \{0,1\}^{p(k)}, (x_1, x_2) \leftarrow \mathsf{A}(r)\colon x_1 \ne x_2,\ h_k(r, x_1) = h_k(r, x_2)\right] \\ &= k^{-\omega(1)}\ .\end{aligned}$$

Definition 5. *A cryptographic 2-1 hash function* $h = \{h_k\}$ *is said to be* one-way *if for every poly-time adversary* A*:*

$$\textsc{Adv}_k(\mathsf{A}) = \mathsf{Pr}\left[x \leftarrow \{0,1\}^k, x' \leftarrow \mathsf{A}(h_k(x))\colon h_k(x') = h_k(x)\right] = k^{-\omega(1)}\ .$$

It actually turns out we need one more technical but rather natural assumption. Essentially, we want the running time to be relatively independent of the output distribution of the adversary used as the black box oracle assuming that its mean value of success is fixed and known.

Definition 6. *We say that the reduction is* oracle-independent *if the running time* $\textsc{Time}_k(\mathsf{S}^{\mathsf{A},f}_{\delta}, f)$ *of* S *is between* $m(k,\delta)$ *and* $u(k)m(k,\delta)$ *for all oracles* A_k *that achieve an advantage of* δ *against* P^f *where* m *is polynomial and* $u(k) = 2^{o(k)}$ *and does not depend on* δ*.*

[2] Classical properties of hash functions and the relations (implications and separations) between them are well studied by Rogaway and Shrimpton [11]. However, it is still not that obvious whether the known constructions are optimal in terms of efficiency.

After these definitions we can now state and prove the result:

Theorem 2. *Let $h = \{h_k\}$ be a cryptographic 2-1 hash function that is collision-resistant. Then for $c < 1$ there exist no power c-secure success-specific oracle-independent (possibly non-uniform) black-box reductions $\mathsf{S}^{\mathsf{A},f}$ showing that h is also a one-way function.*

Proof. Fix a $c < 1$ and a success-specific adversary construction family $\mathsf{S} = \{\mathsf{S}_\delta\}$. According to the theorem, we want to be able to exhibit oracle distributions $\Omega_{\mathsf{S},\mathsf{P},\delta}$ $(0 < \delta \leq 1)$ and a series of δ_k for any given polynomial $q(k)$ such that

$$\lim_{k\to\infty} \frac{\delta_k^c}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}} \left[\textsc{Time}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}} \left[\textsc{Adv}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]} > 1$$

so that δ_k would also converge to 0 as k goes to infinity. Therefore assume we have a fixed polynomial $q(k)$. Let $m(k,\delta)$ and $u(k)$ be the functions guaranteed by oracle-independence for S_δ. Then define the sequence $\delta_k = (ku(k)q(k))^{-1/\epsilon}$ where $\epsilon = 1 - c$. It is clear that $\lim_{k\to\infty} \delta_k = 0$ as desired, since $kq(k)$ is a polynomial with degree at least 1.

We choose f to be a random oracle and choose A_δ from all the oracles that break the given f on exactly δ fraction of inputs[3]. We also assume that A calls take unit time. The best way to find a collision for a random oracle with the help of the inversion adversary is to choose a random input x to f and then to call $\mathsf{A}(f(x))$ to try to get a second pre-image for $f(x)$. Since A succeeds with probability δ, the chance of not having a collision after using this method m times is $(1-\delta)^m$. If the one-wayness adversary is not used, the best possible attack against a random oracle is the birthday attack, that has a probability of success $p(m,k) = O(m^2 2^{-k})$ if k is the security parameter and m is the number of calls to f. This gives us an upper bound for the success probability for the construction: $f(\delta_k, m) = 1-(1-\delta_k)^m + p(m,k) < m\delta_k + p(m,k)$ for large enough k (as δ_k goes to 0 as k increases). This gives us:

$$\begin{aligned}
\frac{\delta_k^{1-\epsilon}}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}} \left[\textsc{Time}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}} \left[\textsc{Adv}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]} &\geq \frac{\delta_k^{1-\epsilon} m(k)}{q(k) f(\delta_k, u(k)m(k))} \\
&= \frac{\delta_k^{1-\epsilon} m(k)}{q(k)(u(k)m(k)\delta_k + p(u(k)m(k),k))} \\
&= \frac{\delta_k^{-\epsilon}}{q(k)u(k)(1 + O(u(k)m(k)\delta^{-1}2^{-k}))} \\
&\geq \frac{k}{2} > 1
\end{aligned}$$

for large enough k since $u(k)m(k)\delta^{-1} = 2^{o(k)}$ according to the assumptions. □

[3] We mean that the oracle distribution is a uniform distribution over all possible $f\colon \{0,1\}^{2k} \to \{0,1\}^k$ and all A_δ that fit the given description for the given f.

7 A Practical Example

We now give another example of using Theorem 1. We stick to properties of hash functions and introduce the property of *division-resistance* important in some time-stamping applications [2] and in the security against the so-called Nostradamus attacks [12]. As the length of the paper is limited, we will not provide an introduction into time-stamping and to Nostradamus attacks. A reader more interested in them should refer to the two papers [2,12].

Definition 7. *A cryptographic 2-1 hash function $h = \{h_k\}$ is said to be* division-resistant *if for every poly-time adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$:*

$$\mathsf{Pr}\left[r \leftarrow \{0,1\}^{p(k)}, y \leftarrow \mathsf{A}_1(r), x_1 \leftarrow \{0,1\}^k, x_2 \leftarrow \mathsf{A}_2(y, x_1) : h_k(r, x_1 \| x_2) = y\right] = k^{-\omega(1)} \ .$$

We are now ready to state the result. The proof is, yet again, rather technical and the reader is advised to try to understand the previous proof before trying this one.

Theorem 3. *Let $h = \{h_k\}$ be a cryptographic 2-1 hash function that is collision-resistant. Then for $c < 1.5$ there exist no power c-secure success-specific oracle-independent (possibly non-uniform) black-box reductions $\mathsf{S}^{\mathsf{A},f}$ showing that h is also division-resistant.*

Proof. We base the proof on theorem 1. As such, fix a $c < 1.5$ and a success-specific adversary construction $\mathsf{S} = \{\mathsf{S}_\delta\}$. According to the theorem, we want to be able to exhibit oracle distributions $\Omega_{\mathsf{S},\mathsf{P},\delta}$ (where $0 < \delta \leq 1$) and a series of δ_k for any given polynomial $q(k)$ such that

$$\lim_{k\to\infty} \frac{\delta_k^c}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{TIME}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{ADV}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]} > 1$$

so that δ_k would also converge to 0 as k goes to infinity. Let $q(k)$ be a polynomial and let $m(k,\delta)$ and $u(k)$ be the functions guaranteed by oracle-independence for S_δ. Define the sequence $\delta_k = (ku(k)^2q(k))^{-1/\epsilon}$ where $\epsilon = 1.5 - c$. It is clear that $\lim_{k\to\infty} \delta_k = 0$ as desired, since $kq(k)$ is a polynomial with degree at least 1.

We now begin constructing the oracle distributions $\Omega_{\mathsf{S},\mathsf{P},\delta}$. We choose the hash functions $h = \{h_k\}$ to be taken from the set of all possible 2-1 cryptographic hash functions where $\pi_{r,x}(y) = h(r, x\|y)$ is a uniformly chosen random permutation that is chosen independently for each choice of r and $x \in \{0,1\}^k$. This ensures that we can always break division-resistance with respect to any output and fixed first half of the input. This choice of h also makes finding collisions hard. Without an adversary oracle A, the best possible tactic (given a fixed r) is clearly to just choose a sequence of random inputs $x_1||y_1$, $x_2||y_2$, …, $x_m||y_m$ so that all y_i are different and then check if any pair of them gives a collision. Denote the probability of such an attack succeeding after at most m different h-queries

by $p(m,k)$. Clearly, the birthday bound applies and $p(m,k) = O(m^2 2^{-k})$. It is worth noting that $p(u(k)m(k,\delta_k),k) = o(1)$.

We now describe the behavior of $\mathsf{A}_\delta = (\mathsf{A}^1_\delta, \mathsf{A}^2_\delta)$. The part A^1_δ is always the same – given k, it just returns 0^k. The behavior of A^2_δ, however, may vary for different values of k. Let $m(k) = m(k,\delta_k)$. There are two possible choices for a given k depending on $m(k)$. If $m(k) > \delta_k^{-0.5}$ then take as $\mathsf{A}^2_\delta(k,\cdot)$ an oracle that always succeeds on a δ-fraction of randomness strings for h while always failing on the others. Otherwise, if $m(k) \le \delta_k^{-0.5}$, choose $\mathsf{A}_{\delta_k}(k,\cdot)$ to be an oracle that breaks h_k for every randomness r but for each of them on only a δ_k fraction of possible partial inputs. In both cases the oracle is not one fixed choice, but rather chosen uniformly from amongst all the oracles matching that description. It should be clear that the success probability of A_δ is δ. Note, however, that the two oracle types we use are very different in terms of breaking collision-resistance. The first one is such that if it works on the given r, we are guaranteed a collision with a query to A^1 and two queries to A^2 while if it does not, the oracle is completely useless. The second one is always somewhat useful, but we normally need a lot more queries to find the collision.

We will now analyze the time-success ratio of S_{δ_k}. We look at three cases for a given k, again depending on $m(k)$.

We first examine the case where $m(k) \le \delta_k^{-0.5}$. For an adversary to break collision resistance, it either has to be able to find two partial inputs for which A_{δ_k} gives an answer or to find a collision with just h queries. The case where we find just one input with A_{δ_k} gives us no more information than a random h query that just happens to give an output of 0^k and as such is equivalent with the case where we just used h queries. It is clear then, that after at most m queries to the oracle the success function is bounded from above by

$$f(m,\delta) = 1 - m\delta(1-\delta)^{m-1} - (1-\delta)^m + p(m,k) = O\left(m^2(\delta^2 + 2^{-k})\right) \quad .$$

Thus

$$\begin{aligned}
\frac{\delta_k^{1.5-\epsilon}}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\textsc{Time}_k(\mathsf{S}^{\mathsf{A}_{\delta_k},h}_{\delta_k})\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\textsc{Adv}_k(\mathsf{S}^{\mathsf{A}_{\delta_k},h}_{\delta_k})\right]} &\ge \frac{m(k)\delta_k^{1.5-\epsilon}}{q(k)f(u(k)m(k),\delta_k)} \\
&= \frac{m(k)\delta_k^{1.5-\epsilon}}{q(k)O\left(u(k)^2 m(k)^2(\delta_k^2 + 2^{-k})\right)} \\
&\ge \frac{\delta_k^{1.5-\epsilon}}{c_0 u(k)^2 q(k)\delta_k^{1.5}(1+o(1))} = \frac{k}{2c_0} \quad ,
\end{aligned}$$

where c_0 is the constant derived from O-notation and we assume k to be large enough so that the $o(1)$ value is less than 1.

For the other two cases the analysis is somewhat simpler. If $\delta_k^{-0.5} < m(k) \le \delta_k^{-1.5}$ then the analysis is a little bit more complicated:

$$\frac{\delta_k^{1.5-\epsilon}}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{TIME}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_\delta,h})\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{ADV}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h})\right]} \geq \frac{m(k)\delta_k^{1.5-\epsilon}}{q(k)(\delta_k+O(u(k)^2m(k)^2 2^{-k}))}$$

$$\geq \frac{\delta_k^{-\epsilon}}{q(k)(1+O(\delta_k^{-4}2^{-k}))} \geq \frac{k}{2c_0} \ .$$

Finally, if $m(k) > \delta_k^{-1.5}$ then

$$\frac{\delta_k^{1.5-\epsilon}}{q(k)} \cdot \frac{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{TIME}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h},h)\right]}{\mathop{\mathsf{E}}_{(\mathsf{A}_{\delta_k},h)\leftarrow\Omega_{\mathsf{S},\mathsf{P},\delta_k}}\left[\mathrm{ADV}_k(\mathsf{S}_{\delta_k}^{\mathsf{A}_{\delta_k},h},h)\right]} \geq \frac{\delta(k)^{-\epsilon}}{q(k)} \geq ku(k)^2 \geq \frac{k}{2c_0} \ ,$$

where the first inequality holds simply because the success probability of $\mathsf{S}^{A,h}$ is bounded by 1. In all three cases the value is larger than $\frac{k}{2c_0}$ for large enough k and thus goes to infinity as k does so we can apply Theorem 1. □

As we see, the proof is quite technical and may seem too much for just an example. However, this theorem has rather interesting implications for Merkle-tree based hash-then-publish time-stamping schemes whose security relies on the collision-resistance of the underlying hash function. Such schemes have been used in practice for many years. The division-resistance condition corresponds to the security requirement for the hash-then publish time-stamping for just two documents and this result thus gives a bound on how much we could tighten the security proofs of the time-stamping scheme based solely on the collision-resistance assumption. This example result answers to the open question in [2] about the existence of *linear-preserving* (i.e. with $c = 1$) reductions from secure hash-and-publish time-stamping schemes to the collision-resistance of the underlying hash function. Our result implies that any such reduction should have $c \geq 1.5$. All known reductions of secure hash-then-publish time-stamping systems to the collision-resistance are quadratic ($c = 2$) and hence, the existence of a reduction with $1.5 \leq c < 2.0$ is still an open question.

8 Conclusions

We have presented a framework for proving lower bounds on the efficiency of cryptographic reductions in terms of time-success ratio of black-box adversary constructions. We have also shown that this framework can actually be used to produce meaningful and interesting lower bounds for reductions actually used in practice that are not too far from best constructions already known.

Our work leaves some related open questions for further study. First of all, it would please us very much to see this framework being used to show some other lower bounds or to even show that some construction used in practice is actually optimal in the sense of time-advantage ratio.

References

1. Baker, T., Gill, J., Solovay, R.: Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question. SIAM Journal on Computing 4, 431–442 (1975)
2. Buldas, A., Saarepera, M.: On Provably Secure Time-Stamping Schemes. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 500–514. Springer, Heidelberg (2004)
3. Gennaro, R., Gertner, Y., Katz, J.: Lower bounds on the efficiency of encryption and digital signature schemes. In: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing, pp. 417–425 (2003)
4. Gennaro, R., Gertner, Y., Katz, J., Trevisan, L.: Bounds on the efficiency of generic cryptographic constructions. SIAM Journal on Computing 35, 217–246 (2006)
5. Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: 41st Annual Symposium on Foundations of Computer Science, Redondo Beach, California, November 2000, pp. 325–335 (2000)
6. Gennaro, R., Trevisan, L.: Lower Bounds on the Efficiency of Generic Cryptographic Constructions. In: FOCS 2000, pp. 305–313 (2000)
7. Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: Proc. of the Twenty First Annual ACM Symposium on Theory of Computing, pp. 44–61 (1989)
8. Kim, J.H., Simon, D.R., Tetali, P.: Limits on the efficiency of one-way permutation-based hash functions. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 535–542 (1999)
9. Luby, M.: Pseudorandomness and cryptographic applications. Princeton University Press, Princeton (1996)
10. Reingold, O., Trevisan, L., Vadhan, S.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
11. Rogaway, P., Shrimpton, T.: Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance. In: Roy, B., Meier, W. (eds.) FSE 2004. LNCS, vol. 3017, pp. 371–388. Springer, Heidelberg (2004)
12. Stevens, M., Lenstra, A., de Weger, B.: Chosen-prefix collisions for md5 and colliding x.509 certificates for different identities. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 1–22. Springer, Heidelberg (2007)

Building Key-Private Public-Key Encryption Schemes

Kenneth G. Paterson and Sriramkrishnan Srinivasan

Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, U.K.
{kenny.paterson,s.srinivasan}@rhul.ac.uk

Abstract. In the setting of identity-based encryption with multiple trusted authorities, TA anonymity formally models the inability of an adversary to distinguish two ciphertexts corresponding to the same message and identity, but generated using different TA master public-keys. This security property has applications in the prevention of traffic analysis in coalition networking environments. In this paper, we examine the implications of TA anonymity for key-privacy for normal public-key encryption (PKE) schemes. Key-privacy for PKE captures the requirement that ciphertexts should not leak any information about the public-keys used to perform encryptions. Thus key-privacy guarantees recipient anonymity for a PKE scheme. Canetti, Halevi and Katz (CHK) gave a generic transform which constructs an IND-CCA secure PKE scheme using an identity-based encryption (IBE) scheme that is selective-id IND-CPA secure and a strongly secure one-time signature scheme. Their transform works in the standard model (i.e. does not require the use of random oracles). Here, we prove that if the underlying IBE scheme in the CHK transform is TA anonymous, then the resulting PKE scheme enjoys key-privacy. Whilst IND-CCA secure, key-private PKE schemes are already known in the standard-model, our result gives the first generic method of constructing a key-private PKE scheme in the standard model. We then go on to investigate the TA anonymity of multi-TA versions of well-known standard model secure IBE schemes. In particular, we prove the TA anonymity and selective-id IND-CPA security of a multi-TA version of Gentry's IBE scheme. Applying the CHK transform, we obtain a new, efficient key-private, IND-CCA secure PKE scheme in the standard model.

Keywords: public-key encryption, key-privacy, identity-based encryption, multiple trusted authorities, TA anonymity, standard model.

1 Introduction

Building public-key encryption (PKE) schemes that are secure in a very strong sense, satisfying indistinguishability against chosen ciphertext attacks or IND-CCA secure, remains a very active area of research. Only a handful of approaches [20,13,11] are known for constructing IND-CCA secure PKE schemes

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 276–292, 2009.

without resorting to the Random Oracle Model [3]. In the usual public-key setting, the security notion termed key-privacy has also gained increasing importance in recent years, in the context of anonymous communications [2]. While specific schemes such as ElGamal, Cramer-Shoup and RSA-based schemes are known to be key-private [2], no generic method is known for constructing a key-private PKE scheme.

Following the results of Cocks [12], and the pairing-based solutions of Sakai, Ohigishi and Kasahara [22] and Boneh and Franklin [7], identity-based cryptography (IBC) [23] has become one of the most active areas of cryptographic research. Canetti *et al.* [11] give a generic construction, now called the CHK transform, to obtain an IND-CCA secure PKE scheme from an IBE scheme that is selective-id IND-CPA secure, and a strong one time signature scheme. No mention is made in [11] of the key-privacy of the PKE schemes arising from the CHK transform.

In the world of identity-based encryption (IBE), the setting of multiple trusted authorities has recently been treated rigorously [21]. In this setting, the relatively new security property termed trusted authority (TA) anonymity captures the inability of an adversary to distinguish two ciphertexts corresponding to the same message and identity, but generated using different TA master public-keys. At a high level, in this paper, we prove that a key-private PKE scheme is obtained from the CHK transform if the underlying IBE scheme has a weak form of TA anonymity. Our result gives the first generic method for constructing a PKE scheme in the standard model that is both key-private and IND-CCA secure.

Based on our current results, we argue that the relatively new notion of TA anonymity is not only of interest in the area of anonymous communications, for example in thwarting traffic analysis [21], but also has rather subtle cryptographic implications for schemes that use IBE as a building block. It therefore merits further study. We investigate the TA anonymity of multi-TA versions of well-known IBE schemes in the standard model and we are easily able to show that they do not satisfy the notion of TA anonymity. By contrast, we prove that a multi-TA version of Gentry's IBE scheme [15] is TA anonymous. Instantiating the CHK transform with this multi-TA Gentry scheme gives us a concrete and reasonably efficient key-private, IND-CCA secure PKE scheme in the standard model.

2 Background

Anonymous encryption was historically motivated in the context of mobile communication. In the standard public-key setting, entities $\mathcal{A}$ and $\mathcal{B}$ exchange encrypted messages using each others' public-keys, over a broadcast medium, where eavesdroppers can see all ciphertexts on the network. It is reasonable to assume that $\mathcal{A}$ and $\mathcal{B}$ will want to keep their identities hidden from such eavesdroppers and this is possible only when ciphertexts do not leak information about the public-keys used to create them, a notion formalized as key-privacy in [2].

In almost all the existing literature on IBE, with a small number of exceptions, there is a single TA that issues keys to all the users in the system, and all ciphertexts are created using the public parameters of that single TA. This TA is also known as the private key generator (PKG) in the literature. In this traditional single-TA

identity-based setting, the notions of security roughly equivalent to the IND-CPA and IND-CCA security notions for PKE were first formalized in [7]. In the IND-CPA and IND-CCA games for IBE, the adversary is also given access to a private key extraction oracle with suitable restrictions on its use.

In IBE, the security notion equivalent to key-privacy in PKE is termed recipient anonymity. The systematic study of recipient anonymity was initiated in [1], motivated both by its intrinsic interest in IBE and for its application in constructing public-key encryption with keyword search (PEKS) schemes. Recipient anonymity models the requirement that ciphertexts should not leak the identity of their intended recipients.

It is possible to envisage scenarios with multiple, independent TAs perhaps sharing some common system parameters. The systematic study of security of IBE in this multi-TA setting was initiated in [21]. In such a setting, in addition to the usual IBE security notions of indistinguishability and recipient anonymity, the notion of TA anonymity arises naturally. TA anonymity captures the requirement that an adversary should find it difficult to distinguish ciphertexts produced using different TA master public-keys, even if the ciphertext is for the same message and identity string. TA anonymity has practical significance, again in the context of anonymous communications. For example, if a coalition of TAs operate in a wireless setting where all ciphertexts can be captured from the network by an adversary, and if the ciphertext were to somehow leak the identity of the TA, this would open up avenues for traffic analysis [21]. However, the cryptographic implications of TA anonymity for schemes that use IBE as a building block have as yet not been studied.

3 Our Contributions

We consider the CHK transform in the setting of multiple public keys that is needed when studying key-privacy. This quite naturally gives rise to a multi-TA IBE setting of the type considered in [21]. We show how to modify the CHK construction to reflect this setting. We then prove that the key-privacy of the PKE scheme resulting from our modified CHK transform follows from a weak form of TA anonymity for the underlying multi-TA IBE scheme. Our result gives us the first generic method of constructing a PKE scheme in the standard model that is key-private, as well as being IND-CCA secure.

We note that the transform of Boneh and Katz [8] builds on the ideas of [11] to give a more efficient construction of PKE from IBE. We can prove similar results for the Boneh-Katz transform. (The results from both [11,8] appear in [6].) Due to constraints of space and bearing in mind that the proof of security in [8] is more involved and that the our aim in this paper has been to highlight the significance of TA anonymity, especially with relation to building key-private PKE schemes, we have limited our discussions to the original CHK transform.

To obtain concrete PKE schemes that are key-private and IND-CCA secure, we study the TA anonymity properties of multi-TA versions of the known standard-model IND-CPA secure IBE schemes. We are able to prove that a multi-TA version of the scheme of Gentry [15] is TA anonymous. We are also

able to show that multi-TA versions of the two popular standard model schemes of Boneh and Boyen in [5], termed BB1 and BB2 in the literature, and multi-TA versions of the schemes related to the BB1 scheme, such as those of Waters [24] and Naccache [19], trivially do not meet the notion of TA anonymity.

4 Definitions

In this section, we provide basic definitions needed for the remainder of the paper. Here, we omit standard definitions for PKE, IBE and strongly secure one-time signatures which can be found in the full version of this paper or for example in [6].

Definition 1. *A pairing-friendly group generator* `PairingGen` *is a polynomial time algorithm with input* 1^k *and output a tuple* (G, G_T, e, p, g)*. Here* G, G_T *are groups of prime order* p*,* g *generates* G*, and* $e : G \times G \rightarrow G_T$ *is a bilinear, non-degenerate and efficiently computable map.*

For ease of presentation, we work exclusively in the setting where e is symmetric; our definitions and results can be generalised to the asymmetric setting where $e : G_1 \times G_2 \rightarrow G_T$, with G_1 and G_2 being different groups. Further details concerning the basic choices that are available when using pairings in cryptography can be found in [14].

Definition 2. *We define the advantage of an algorithm* $\mathcal{A}$ *in solving the Truncated Decisional* ℓ*-Augmented Bilinear Diffie-Hellman Exponent (*ℓ*-TDABDHE) problem in* (G, G_T) *to be:*

$$\boldsymbol{Adv}_{\mathcal{A}}^{\ell\text{-}TDABDHE}(k) = |\Pr(\mathcal{A}(g', g'_{(l+2)}, g_1, g_2, \ldots, g_l, e(g_{(l+1)}, g')) = 1)$$

$$- \Pr(\mathcal{A}(g', g'_{(l+2)}, g_1, g_2, \ldots, g_l, Z) = 1)|$$

where $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$*,* $Z \xleftarrow{\$} \mathbb{G}_T$*,* $g' \xleftarrow{\$} \mathbb{G}$*,* $g_i = g^{(\alpha^i)}$ *and* $g'_i = g'^{(\alpha^i)}$*. Here, we implicitly assume that parameters* (G, G_T, e, p, g) *are given to* $\mathcal{A}$ *as additional inputs. The distribution on the left is referred to as* P_{ABDHE} *and that on the right is referred to as* R_{ABDHE}*.*

We note that this is the same assumption that is used to prove the security of the IBE scheme presented in [15].

Definition 3. *We say that the* (t, ϵ, ℓ)*-TDABDHE problem is* hard *in* (G, G_T) *if no* t*-time algorithm has advantage at least* ϵ *in solving the* ℓ*-TDABDHE problem in* (G, G_T)*.*

Definition 4. *A function* $\epsilon(k)$ *is said to be negligible if, for every* c*, there exists* k_c *such that* $\epsilon(k) \leq k^{-c}$ *for every* $k \geq k_c$*.*

5 Multi-TA IBE

A multi-TA IBE scheme is defined in [21] in terms of five algorithms:

- `CommonSetup`: On input 1^k, outputs *params*, a set of system parameters shared by all TAs; $\mathcal{T} = \{ta_i : 1 \leq i \leq n\}$ will represent the set of (labels of) TAs, where $n = n(k) \in \mathbb{N}$.
- `TASetup`: On input *params*, outputs a master public-key *mpk* (which includes *params*), and a master secret key *msk*. This algorithm is randomized and executed independently for each TA in $\mathcal{T}$.
- `KeyDer`, `Enc`, `Dec`: These are all as per a normal IBE scheme.

Following the reasoning in [21] we note that for the concrete schemes and transforms considered in this paper, common parameters are needed in order to achieve the notion of TA anonymity; doing so without having some (non-trivial) common parameters is an interesting open problem. We note that it is not unreasonable to assume that the different TAs may share some common system parameters (e.g. the output of a pairing parameter generator) [21]. In fact the possibility of TAs sharing parameters becomes much more likely when we consider the greater complexity of setting up an IBE scheme (where consideration has to be given, among other things, to the choice of elliptic curves, groups used in Pairings, the representation of elements etc.) compared to the "relative simplicity" of setting up, say an RSA scheme. The IEEE P1363.3 working group aims to produce a set of standards specific to identity-based cryptography to address these difficulties. Indeed, sharing of common parameters is inevitable if any kind of interoperability is desired between TAs and such scenarios are becoming more and more desirable [4].

5.1 Security Models for Multi-TA IBE

In all the security games that follow, we associate to an adversary $\mathcal{A}$ and a bit $b \in \{0, 1\}$, the advantage of the adversary for a "notion-attack" combination, which is defined to be:

$$\mathbf{Adv}_{\mathcal{A}}^{\text{notion-atk-b}}(k) = \left| \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{notion-atk-1}}(k) = 1] - \Pr[\mathbf{Exp}_{\mathcal{A}}^{\text{notion-atk-0}}(k) = 1] \right|.$$

A scheme is said to be "notion-atk"-secure if the advantage of all PPT adversaries is negligible as a function of the security parameter k.

We detail the m-IND-RA-TAA-CCA experiment as defined in [21] that simultaneously captures message indistinguishability, recipient anonymity and TA anonymity in the multi-TA IBE setting, against chosen ciphertext adversaries. This model also gives the adversary access to a `Corrupt` oracle that returns the master secret key for a TA of the adversary's choice.

In the security game defined below, $TASet$ represents the set of TAs that have been corrupted, i.e. queried for their master secret keys, $IDSet_{ta}$ represents the set of identities queried for private keys for each $ta \in \mathcal{T}$, while $CSet_{ta}$ represents the set of identity/ciphertext pairs on which decryption queries have been performed for each $ta \in \mathcal{T}$. In these games, $MPK = \{mpk_{ta} : ta \in \mathcal{T}\}$ and $MSK = \{msk_{ta} : ta \in \mathcal{T}\}$ represent the set of all master public-keys and all master secret keys, respectively.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{m-IND-RA-TAA-CCA-b}}(k)$
$params \leftarrow \texttt{CommonSetup}(1^k)$
$TASet \leftarrow \emptyset$
$\forall ta \in \mathcal{T}$, $(mpk_{ta}, msk_{ta}) \leftarrow \texttt{TASetup}(params)$,
$IDSet_{ta} \leftarrow \emptyset$ and $CSet_{ta} \leftarrow \emptyset$
$(ta_0, ta_1, id_0, id_1, m_0, m_1, state) \leftarrow$
$\mathcal{A}^{\texttt{Corrupt},\texttt{KeyDer},\texttt{Dec}}(\texttt{find}, MPK)$
$c^* \leftarrow \texttt{Enc}(mpk_{ta_b}, id_b, m_b)$
$b' \leftarrow \mathcal{A}^{\texttt{Corrupt},\texttt{KeyDer},\texttt{Dec}}(\texttt{guess}, c^*, state)$
If $\{m_0, m_1\} \not\subseteq \mathsf{MsgSp}$ or $|m_0| \neq |m_1|$
then Return 0
If $(ta_0 = ta_1$ and $id_0 = id_1$ and $m_0 = m_1)$
then Return 0
If $ta_0 \notin TASet$, $ta_1 \notin TASet$, $id_0 \notin IDSet_{ta_0}$, $id_1 \notin IDSet_{ta_1}$, $(id_0, c^*) \notin CSet_{ta_0}$ and $(id_1, c^*) \notin CSet_{ta_1}$ then Return b' else Return 0

Oracle $\texttt{Corrupt}(ta)$
$TASet \leftarrow TASet \cup \{ta\}$
Return msk_{ta}

Oracle $\texttt{KeyDer}(ta, id)$
$IDSet_{ta} \leftarrow IDSet_{ta} \cup \{id\}$
$usk_{id,ta} \leftarrow \texttt{KeyDer}(msk_{ta}, id)$
Return $usk_{id,ta}$

Oracle $\texttt{Dec}(ta, id, c)$
$CSet_{ta} \leftarrow CSet_{ta} \cup \{(id, c)\}$
$usk_{id,ta} \leftarrow \texttt{KeyDer}(msk_{ta}, id)$
$m \leftarrow \texttt{Dec}(mpk_{ta}, usk_{id,ta}, c)$
Return m

By placing suitable restrictions on the m-IND-RA-TAA-CCA security notion, we can define other, weaker security notions appropriate to the multi-TA IBE setting. For example, CPA secure versions can be defined by removing the adversary's access to the decryption oracle. By removing the adversary's access to the `Corrupt` oracle we define "restricted" versions, and appropriate selective-id versions can be defined by having the adversary commit ahead of time to the identities used in the challenge query. Furthermore, setting $m_0 = m_1$, $id_0 = id_1$ or $ta_0 = ta_1$ gives security notions appropriate to specific circumstances. We will elaborate on some of these security models as we encounter them in this paper.

6 Key-Privacy of the CHK Transform

Canetti *et al.* [11] give a construction that builds an IND-CCA secure PKE scheme from a selective-id IND-CPA secure IBE scheme and a strongly secure one-time signature scheme.

We first describe a security notion for PKE which we term IND-IK-CCA and which simultaneously captures message indistinguishability and key-privacy. We then define the selective-id r-m-IND-TAA-CPA security notion for IBE, this being a weakened version of the m-IND-RA-TAA-CCA notion defined above. We then modify the CHK construction from [11] to reflect the setting of multiple users. Finally we show that the IND-IK-CCA security of the public-key encryption scheme built using the (modified) CHK transform follows from the selective-id r-m-IND-TAA-CPA security of the underlying IBE scheme.

We note that we do not require recipient anonymity of the IBE scheme to obtain our result. Rather, the security property needed from the IBE scheme is the form TA anonymity which is captured in our selective-id r-m-IND-TAA-CPA security notion. In section 7.1 we will prove that a multi-TA version of Gentry's IBE scheme meets the stronger m-IND-RA-TAA-CPA security notion. This is sufficient

for the application of our result. Instantiating the CHK transform with the multi-TA Gentry scheme and any strongly secure one-time signature scheme will give us a concrete construction of a key-private and IND-CCA secure PKE scheme.

6.1 IND-IK-CCA Security for PKE

Bellare *et al.* [2] define two notions, IK-CPA and IK-CCA security, that capture the notions of key-privacy under chosen plaintext attacks and chosen ciphertext attacks, respectively. For our purposes, we define a combined security notion which simultaneously captures both message indistinguishability and key-privacy. We term this IND-IK-CCA security.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{IND-IK-CCA-b}}(k)$
$I \xleftarrow{\$} \texttt{CommonSetup}(1^k)$
$(PK_0, SK_0) \xleftarrow{\$} \texttt{KeyGen}(I)$
$(PK_1, SK_1) \xleftarrow{\$} \texttt{KeyGen}(I)$
$CSet_{SK_0} \leftarrow \emptyset$, $CSet_{SK_1} \leftarrow \emptyset$
$(m_0, m_1, state) \leftarrow \mathcal{A}^{\texttt{Dec}}(\texttt{find}, PK_0, PK_1)$
$c^* \leftarrow \texttt{Enc}(PK_b, m_b)$
$b' \leftarrow \mathcal{A}^{\texttt{Dec}}(\texttt{guess}, c^*, state)$
If $|m_0| \neq |m_1|$ or $m_0 = m_1$ then Return 0
If $c^* \notin CSet_{SK_0}$ and $c^* \notin CSet_{SK_1}$
then Return b' else Return 0

Oracle $\texttt{Dec}(PK_b, c)$
$CSet_{SK_b} \leftarrow CSet_{SK_b} \cup \{c\}$
$m \leftarrow \texttt{Dec}(SK_b, c)$
Return m

6.2 Security for Multi-TA IBE

We define the selective-id r-m-IND-TAA-CPA security notion for multi-TA IBE. A single identity is used in the challenge phase in this model, i.e. $id_0 = id_1$. Furthermore, the adversary commits to this identity at the start of the game. The adversary is not allowed to make decryption or `Corrupt` queries.

Experiment $\mathbf{Exp}_{\mathcal{A}}^{\text{s-id r-m-IND-TAA-CPA-b}}(k)$
$id^* \leftarrow \mathcal{A}(1^k)$
$params \leftarrow \texttt{CommonSetup}(1^k)$
$TASet \leftarrow \emptyset$
$\forall ta \in \mathcal{T}$, $(mpk_{ta}, msk_{ta}) \leftarrow \texttt{TASetup}(params)$,
$IDSet_{ta} \leftarrow \emptyset$
$(ta_0, ta_1, m_0, m_1, state) \leftarrow$
$\quad \mathcal{A}^{\texttt{KeyDer}}(\texttt{find}, MPK)$
$c^* \leftarrow \texttt{Enc}(mpk_{ta_b}, id^*, m_b)$
$b' \leftarrow \mathcal{A}^{\texttt{KeyDer}}(\texttt{guess}, c^*, state)$
If $\{m_0, m_1\} \not\subseteq \mathsf{MsgSp}$ or $|m_0| \neq |m_1|$ or $m_0 = m_1$
then Return 0
If $ta_0 = ta_1$ then Return 0
If $ta_0 \notin TASet$, $ta_1 \notin TASet$, $id^* \notin IDSet_{ta_0}$,
$id^* \notin IDSet_{ta_1}$ then Return b' else Return 0

Oracle $\texttt{KeyDer}(ta, id)$
$IDSet_{ta} \leftarrow IDSet_{ta} \cup \{id\}$
$usk_{id,ta} \leftarrow \texttt{KeyDer}(msk_{ta}, id)$
Return $usk_{id,ta}$

6.3 The Modified CHK Transform

Let $\Pi' = \{\texttt{CommonSetup}', \texttt{TASetup}, \texttt{KeyDer}, \texttt{Enc}', \texttt{Dec}'\}$ be a multi-TA IBE scheme for identities of length n.

Let $\texttt{Sig} = \{\texttt{Gen}, \texttt{Sgn}, \texttt{Vrfy}\}$ be a one-time signature scheme in which the verification keys output by $\texttt{Gen}$ have length n.

Define $\Pi = \{\texttt{CommonSetup}, \texttt{KeyGen}, \texttt{Enc}, \texttt{Dec}\}$ as follows

- $\texttt{CommonSetup}$: Runs $\texttt{CommonSetup}'$ to obtain *params*.
- $\texttt{KeyGen}$: Runs $\texttt{TASetup}$ to obtain mpk, msk. The public-key is $PK = mpk$ (PK includes *params*, as mpk by definition includes *params*) and the secret key is $SK = msk$.
- $\texttt{Enc}$: To encrypt a message m using public-key PK, the sender first runs $\texttt{Gen}$ to obtain a verification key vk and the corresponding signing key sk (with $|vk| = n$). Then, the sender computes $c = \texttt{Enc}(PK, m) = \texttt{Enc}'(mpk, vk, m)$ (i.e. the sender encrypts the message m with respect to identity vk for recipient with public-key $PK = mpk$) and $\sigma = \texttt{Sgn}(sk, c)$. The final ciphertext is (vk, c, σ).
- $\texttt{Dec}$: To decrypt (vk, c, σ) using the secret key msk, the recipient first checks whether $\texttt{Vrfy}(vk, c, \sigma) \stackrel{?}{=} 1$. If not, the receiver outputs $\perp$. Otherwise, the receiver computes $usk_{vk} = \texttt{KeyDer}(msk, vk)$ and outputs $m = \texttt{Dec}(SK, c) = \texttt{Dec}'(mpk, usk_{vk}, c)$.

Theorem 1. *If Π' is an IBE scheme which is selective-id r-m-IND-TAA-CPA secure and Sig is a strongly secure one-time signature scheme, then Π is an IND-IK-CCA secure PKE scheme.*

Proof. Our proof follows closely the proof of [11] with suitable modifications to reflect the setting of multiple users. In the following, expressions of the form $\Pr_{A,S}[\texttt{Event}]$ denote the probability that an $\texttt{Event}$ occurs when an adversary $\mathcal{A}$ interacts with a scheme S in a specified security game.

Let $\mathcal{A}$ be an IND-IK-CCA adversary against Π. We say a ciphertext (vk, c, σ) is valid if $\texttt{Vrfy}(vk, c, \sigma) = 1$. Let (vk^*, c^*, σ^*) denote the challenge ciphertext received by $\mathcal{A}$ during a particular run of the experiment and let $\texttt{Forge}$ denote the event that $\mathcal{A}$ submits a valid ciphertext (vk^*, c, σ) to its decryption oracle.

Claim 1: $\Pr_{A,\Pi}[\texttt{Forge}]$ is negligible.

Proof of Claim 1: $\mathcal{A}$ is an IND-IK-CCA adversary against the PKE scheme Π. We use $\mathcal{A}$ to construct an adversary $\mathcal{F}$ that forges a signature with respect to the one-time signature scheme $\texttt{Sig}$, with probability $\Pr_{A,\Pi}[\texttt{Forge}]$.

$\mathcal{F}$ is given a verification key vk. $\mathcal{F}$ first runs $\texttt{KeyGen}$ to obtain (PK_0, SK_0) and (PK_1, SK_1). It gives $\mathcal{A}$ the two public-keys PK_0 and PK_1. Note that $\mathcal{F}$ can answer any decryption queries of $\mathcal{A}$.

If $\mathcal{A}$ happens to submit a valid ciphertext (vk^*, c, σ) to its decryption oracle before requesting the challenge ciphertext then $\mathcal{F}$ simply outputs the forgery (c, σ) and stops.

Otherwise, when $\mathcal{A}$ outputs messages m_0 and m_1, it chooses a random bit b and computes $c^* = \texttt{Enc}'(mpk_b, vk^*, m_b)$ and obtains from its signing oracle a signature σ^* on the message c^*, i.e. $\sigma^* = \texttt{Sgn}(sk, c^*)$ where sk is the signing key corresponding to vk. $\mathcal{F}$ gives $\mathcal{A}$ the challenge ciphertext (vk^*, c^*, σ^*)

Subsequently, if $\mathcal{A}$ submits a valid ciphertext (vk^*, c, σ) to its decryption oracle, (note that we must have $(c, \sigma) \neq (c^*, \sigma^*)$) $\mathcal{F}$ simply outputs (c, σ) as its forgery.

It is easy to see that $\mathcal{F}$'s success probability is exactly $\Pr_{\mathcal{A},\Pi}[\texttt{Forge}]$.

Claim 2: $|\Pr_{\mathcal{A},\Pi}[\texttt{Succ} \wedge \overline{\texttt{Forge}}] + \frac{1}{2}\Pr_{\mathcal{A},\Pi}[\texttt{Forge}] - \frac{1}{2}|$ is negligible.

Proof of Claim 2: We now use $\mathcal{A}$ to construct a selective-id r-m-IND-TAA-CPA attacker $\mathcal{B}$ against the IBE scheme Π'.

Adversary $\mathcal{B}$ acts as a Challenger for $\mathcal{A}$ as follows.

$\mathcal{B}$ runs $\texttt{Gen}(1^k)$ to obtain (sk^*, vk^*) and outputs a target identity $id^* = vk^*$ to its Challenger $\mathcal{C}$.

$\mathcal{C}$ gives $\mathcal{B}$ MPK, the set of all master public-keys in the multi-TA IBE scheme. Adversary $\mathcal{B}$ gives $\mathcal{A}$ the two public-keys $PK_0 = mpk_{ta_0}$ and $PK_1 = mpk_{ta_1}$.

$\mathcal{A}$ is a IND-IK-CCA attacker against the public-key scheme. When $\mathcal{A}$ makes decryption queries on ciphertexts of the form (vk, c, σ), it specifies whether it wants the decryption corresponding to PK_0 or PK_1. $\mathcal{B}$ answers decryption queries as follows.

- If $vk = vk^*$ then $\mathcal{B}$ checks whether $\texttt{Vrfy}(vk^*, c, \sigma) = 1$. In this case, $\mathcal{B}$ does not know the corresponding IBE secret key corresponding to the identity vk^* and it is not allowed to make this query to its Challenger $\mathcal{C}$. Consequently, $\mathcal{B}$ aborts and outputs a random bit. If $\texttt{Vrfy}(vk^*, c, \sigma) \neq 1$ then $\mathcal{B}$ responds with $\perp$.
- If $vk \neq vk^*$ and $\texttt{Vrfy}(vk, c, \sigma) \neq 1$ then $\mathcal{B}$ responds with $\perp$.
- If $vk \neq vk^*$ and $\texttt{Vrfy}(vk, c, \sigma) = 1$ then $\mathcal{B}$,
  - Makes the oracle query $\texttt{KeyDer}(ta_i, vk)$ where PK_i is specified in the challenge query and obtains usk_{vk,ta_i}.
  - Computes $m = \texttt{Dec}'(mpk_{ta_i}, usk_{vk,ta_i}, c)$ and responds with m.

At some point during the simulation $\mathcal{A}$ outputs two equal length messages m_0 and m_1. $\mathcal{B}$ forwards (ta_0, m_0) and (ta_1, m_1) to its Challenger. $\mathcal{B}$ is given the challenge ciphertext $c^* = \texttt{Enc}'(mpk_{ta_b}, id^*, m_b)$. $\mathcal{B}$ computes $\sigma^* = \texttt{Sgn}(sk^*, c^*)$ and gives $\mathcal{A}$ (vk^*, c^*, σ^*).

$\mathcal{A}$ continues to make decryption oracle queries which are answered by $\mathcal{B}$ as before.

Finally $\mathcal{A}$ outputs a guess b' and this same guess is output by $\mathcal{B}$ and $\mathcal{B}$ wins if $b' = b$. We note that $\mathcal{B}$ provides a perfect simulation for $\mathcal{A}$ as well as a legal strategy for attacking the IBE scheme. In particular it never requests the secret key corresponding to the target identity vk^* for either of the target TAs.

Therefore we have

$$|\Pr_{\mathcal{B},\Pi'}[\texttt{Succ}] - \tfrac{1}{2}| = |\Pr_{\mathcal{A},\Pi}[\texttt{Succ} \wedge \overline{\texttt{Forge}}] + \tfrac{1}{2} \cdot \Pr_{\mathcal{A},\Pi}[\texttt{Forge}] - \tfrac{1}{2}|.$$

Claim 2 then follows as we know the left hand side of the above equation is negligible by the assumed security of the IBE scheme.

Finally, we have

$$\begin{aligned} & |\Pr_{\mathcal{A},\Pi}[\mathtt{Succ}] - \tfrac{1}{2}| \\ \leq\ & |\Pr_{\mathcal{A},\Pi}[\mathtt{Succ} \wedge \mathtt{Forge}] - \tfrac{1}{2} \cdot \Pr_{A,\Pi}[\mathtt{Forge}]| \\ & + |\Pr_{\mathcal{A},\Pi}[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}] + \tfrac{1}{2}\Pr_{\mathcal{A},\Pi}[\mathtt{Forge}] - \tfrac{1}{2}| \\ \leq\ & \tfrac{1}{2} \cdot \Pr_{\mathcal{A},\Pi}[\mathtt{Forge}] + |\Pr_{\mathcal{A},\Pi}[\mathtt{Succ} \wedge \overline{\mathtt{Forge}}] + \tfrac{1}{2}\Pr_{\mathcal{A},\Pi}[\mathtt{Forge}] - \tfrac{1}{2}|. \end{aligned}$$

The proof of the result follows from the proofs of claims 1 and 2.

7 Anonymity of Standard Model IBE Schemes

To obtain a standard-model-secure key-private PKE scheme by applying the CHK transform, we need a multi-TA IBE scheme that is suitably TA anonymous under chosen plaintext attacks, in the standard model. While the TA anonymity of multi-TA versions of some popular IBE schemes in the Random Oracle Model has been previously studied in [21], the TA anonymity of multi-TA versions of standard model IBE schemes has not as yet been investigated.

A multi-TA version of the BB1 IBE scheme from [5] can be sketched similar to the multi-TA version of Gentry's IBE scheme in section 7.1.

We can easily show that such a multi-TA BB1 scheme is not TA anonymous. Let us consider an adversary that requests the encryption of a message m to identity id in either ta_0 or ta_1, in its challenge. The challenge ciphertext it receives is of the form:

$$c^* = (\hat{e}(g_1, g_2)^s \cdot m, g^s, F(id)^s) = (A, B, C).$$

Since

$$params_{ta_0} = (params, g_1, g_2, \hat{e}(g_1, g_2), h, F)$$

and

$$params_{ta_1} = (params, g_1', g_2', \hat{e}(g_1', g_2'), h', F')$$

the adversary simply checks if

$$\hat{e}(B, {g_1}^{id} \cdot h) = \hat{e}(C, g) \quad \text{or} \quad \hat{e}(B, {g_1'}^{id} \cdot h') = \hat{e}(C, g)$$

to find, with overwhelming probability, which TA's parameters were used.

Multi-TA analogues of schemes related to the BB1 scheme, such as those of Waters [24] and Naccache [19], as well as the multi-TA analogue of the BB2 scheme [5], are also not TA anonymous for similar reasons.

The original scheme by Gentry [15] is recipient anonymous and we now show that a multi-TA version of this scheme is TA anonymous.

7.1 Multi-TA Gentry

We first sketch a multi-TA version of Gentry's IBE scheme. We assume identities are elements in $\mathbb{Z}_p^*$ and messages are elements in $\mathbb{G}_T$. Later, we will need identities that are bit-strings of a fixed length; such identities can easily and securely be converted into elements of $\mathbb{Z}_p^*$ by applying a suitable collision-resistant hash function.

$\texttt{CommonSetup}(1^k)$:

- $(G, G_T, e, p, g) \leftarrow \texttt{PairingGen}(1^k)$.
- Output $params = (G, G_T, e, p, g)$.

$\texttt{TASetup}(params)$:

- Pick $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$. Set $g_1 = g^\alpha$.
- Pick $h \xleftarrow{\$} \mathbb{G}$.
- Define function $F : \mathbb{Z}_p^* \rightarrow \mathbb{G}$ st $F(x) = g_1 \cdot g^{-x}$.
- Set $mpk = (params, g_1, h, e(g,g), e(g,h), F)$.
- Set $msk = \alpha$.
- Output (mpk, msk).

$\texttt{KeyDer}(ta, id)$:

- Pick $r_{id} \xleftarrow{\$} \mathbb{Z}_p^*$.
- Output $usk_{ta,id} = (r_{id}, h_{id})$ where $h_{id} = (h \cdot g^{-r_{id}})^{\frac{1}{\alpha - id}}$.

$\texttt{Enc}(ta, id, m)$:

- Pick $s \xleftarrow{\$} \mathbb{Z}_p^*$.
- Output $c = (F(id)^s, e(g,g)^s, e(g,h)^{-s} \cdot m)$.

$\texttt{Dec}(ta, id, c)$:

- Parse c as (u, v, w).
- Parse $usk_{ta,id}$ as (d_0, d_1).
- Output $m = w \cdot e(u, h_{id}) v^{r_{id}}$.

The `Multi-TA Gentry` scheme.

Anonymity of Multi-TA Gentry: We will first show that the multi-TA version of Gentry's IBE scheme meets the r-m-IND-RA-TAA-CPA security notion under the q-TDABDHE assumption. This gives us a reduction that has tightness similar to the original single-TA scheme. Our proof follows closely the proof of [15] with suitable modifications to reflect the multi-TA setting.

Theorem 2. *Let $q = q_{id} + 1$ where q_{id} is the maximum number of private key extraction queries allowed by the adversary per TA. Assume the (t, ϵ, q)-TDABDHE assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Then, the above multi-TA IBE scheme is (t', ϵ', q_{id}) r-m-IND-RA-TAA-CPA secure for $t' = t - O(t_{exp} \cdot n \cdot q^2)$ and $\epsilon' = \epsilon + (4/p)$ where t_{exp} is the time required to exponentiate in $\mathbb{G}$.*

Proof. The proof is given in the appendix.

The above proof can be modified slightly to enable $\mathcal{B}$ to respond to `Corrupt` queries as well, thereby giving us a proof of security for the m-IND-RA-TAA-CPA security for the multi-TA version of Gentry's IBE scheme, under the same assumptions:

Theorem 3. *Let $q = q_{id} + 1$ where q_{id} is the maximum number of private key extraction queries allowed by the adversary per TA. Assume the (t, ϵ, q)-TDABDHE assumption holds in $(\mathbb{G}, \mathbb{G}_T)$. Then, the above multi-TA IBE scheme is (t', ϵ', q_{id}) m-IND-RA-TAA-CPA secure for $t' = t - O(t_{exp} \cdot n \cdot q^2)$ and $\epsilon' = (\epsilon + (4/p)) \cdot \binom{n}{2}$ where t_{exp} is the time required to exponentiate in $\mathbb{G}$.*

Proof. $\mathcal{B}$ simply generates two related q-TDABDHE challenges from the original input challenge and uses these to respond to private key extraction queries for two specific TAs indexed by $ta_x, ta_y \in \mathcal{T}$. It cannot respond to `Corrupt` queries on these two TAs and the success of the proof relies on $\mathcal{A}$ choosing these two TAs in its Challenge query (thereby reducing the tightness of the reduction).

For all other TAs $\{ta_i \in T : i \neq x, i \neq y\}$, $\mathcal{B}$ simply generates the master public-keys and master secret keys itself and can therefore respond to private key extraction and `Corrupt` queries on these TAs. Further details are similar to the proof of Theorem 2.

Final Observations: The multi-TA version of Gentry's IBE scheme that we have given meets stronger notions of security than those required for the application of Theorem 1. We can therefore instantiate the modified CHK transform with the multi-TA version of Gentry's IBE scheme (and any strongly secure one-time signature scheme) to obtain an IND-IK-CCA PKE scheme. This scheme is quite efficient. Ciphertexts consist of 3 group elements plus a signature and a verification key from the one-time signature scheme. Encryption and decryption cost roughly the same as encryption and decryption in Gentry's scheme, with the additional requirement of generating or verifying one-time signatures.

8 Conclusion and Future Work

We have shown that the key-privacy of the PKE scheme resulting from the application of the CHK transform follows from the TA anonymity of the underlying IBE scheme, giving us the first generic method to construct a key-private PKE scheme. We have investigated various IBE schemes in the standard model and shown that a multi-TA version of Gentry's IBE scheme meets the notion of TA anonymity. We have also constructed a key-private PKE scheme by instantiating the CHK transform with the multi-TA version of Gentry's IBE scheme.

We believe that the relatively new notion of TA anonymity in the setting of multiple TAs has rather subtle cryptographic implications on schemes that use IBE as a building block, but these have not been studied rigorously. For example, Holt [16] also considered security of IBE in the multi-TA setting, motivated by earlier work on anonymous credential systems [17,9]. However, the TA anonymity requirements for these applications are yet to be formally investigated.

Acknowledgements

This research was sponsored in part by the US Army Research Laboratory and the UK Ministry of Defence and was accomplished under Agreement Number

W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the US Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

The second author is supported by a Dorothy Hodgkin Postgraduate Award, funded by EPSRC and Vodafone and administered by Royal Holloway, University of London.

We are grateful to the anonymous referees for valuable comments and suggestions and Gaven Watson for proofreading parts of this work.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 205–222. Springer, Heidelberg (2005)
2. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
3. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Boklan, K.D., Klagsbrun, Z., Paterson, K.G., Srinivasan, S.: Flexible and Secure Communications in an Identity-Based Coalition Environment. In: IEEE Military Communications Conference, 2008. MILCOM 2008, pp. 1–6 (2008)
5. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin and Camenisch [10], pp. 223–238
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM J. Comput. 36(5), 1301–1328 (2007)
7. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
8. Boneh, D., Katz, J.: Improved efficiency for CCA-secure cryptosystems built using identity-based encryption. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 87–103. Springer, Heidelberg (2005)
9. Bradshaw, R.W., Holt, J.E., Seamons, K.E.: Concealing complex policies with hidden credentials. In: Atluri, V., Pfitzmann, B., McDaniel, P.D. (eds.) ACM Conference on Computer and Communications Security, pp. 146–157. ACM, New York (2004)
10. Cachin, C., Camenisch, J.L. (eds.): EUROCRYPT 2004. LNCS, vol. 3027. Springer, Heidelberg (2004)
11. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: Cachin and Camenisch [10], pp. 207–222
12. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (2001)

13. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk [18], pp. 13–25
14. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), http://eprint.iacr.org/
15. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
16. Holt, J.E.: Key privacy for identity based encryption. Cryptology ePrint Archive, Report 2006/120 (2006), http://eprint.iacr.org/
17. Holt, J.E., Bradshaw, R.W., Seamons, K.E., Orman, H.K.: Hidden credentials. In: Jajodia, S., Samarati, P., Syverson, P.F. (eds.) WPES, pp. 1–8. ACM Press, New York (2003)
18. Krawczyk, H. (ed.): CRYPTO 1998. LNCS, vol. 1462. Springer, Heidelberg (1998)
19. Naccache, D.: Secure and practical identity-based encryption. Information Security, IET 1(2), 59–64 (2007)
20. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: STOC, pp. 427–437. ACM Press, New York (1990)
21. Paterson, K.G., Srinivasan, S.: Security and anonymity of identity-based encryption with multiple trusted authorities. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 354–375. Springer, Heidelberg (2008)
22. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: The 2000 Sympoium on Cryptography and Information Security, Okinawa, Japan, January 2000, pp. 26–28 (2000)
23. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
24. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

Appendix

Proof of Theorem 2

Proof. Let $\mathcal{A}$ be an adversary that (t', ϵ', q_{id}) breaks the r-m-IND-RA-TAA-CPA security of the multi-TA Gentry IBE scheme described. Here q_{id} is the maximum number of private key extraction queries allowed by the adversary per TA. We construct an algorithm $\mathcal{B}$ that solves the q-TDABDHE problem, as follows.

$\mathcal{B}$ takes as input a random q-TDABDHE challenge $(g', g'_{q+2}, g_1, g_2, \ldots, g_q, Z)$ where Z is either $e(g_{q+1}, g')$ or a random element of $\mathbb{G}_T$ and the expected additional inputs $(\mathbb{G}, \mathbb{G}_T, e, p, g)$. (Recall that $g_i = g^{(\alpha^i)}$.)

Algorithm $\mathcal{B}$ proceeds as follows.

For an n TA system, $\mathcal{T} = \{ta_i : 1 \leq i \leq n\}$ represents the set of (labels of) TAs, where $n = n(k) \in \mathbb{N}$. $\mathcal{B}$ uses the input challenge to generate n related q-TDABDHE challenges, one for each TA in $\mathcal{T}$. Let CHAL_i denote the q-TDABDHE corresponding to $ta_i \in \mathcal{T}$.

For $ta_i \in \mathcal{T}$, $\mathcal{B}$ first draws $\beta_i \xleftarrow{\$} \mathbb{Z}_p^*$ and sets CHAL_i equal to:

$$(g', {g'_{q+2}}^{(\beta_i{}^{(q+2)})}, {g_1}^{(\beta_i)}, {g_2}^{(\beta_i{}^2)}, \ldots, {g_q}^{(\beta_i{}^q)}, Z^{(\beta_i{}^{(q+1)})})$$

or

$$(g', g'^{((\alpha\beta_i)^{q+2})}, g^{((\alpha\beta_i))}, g^{((\alpha\beta_i)^2)}, \ldots, g^{((\alpha\beta_i)^q)}, Z^{(\beta_i{}^{(q+1)})}).$$

We make a few important observations. Firstly, note that if $Z = e(g_{q+1}, g')$ then $Z^{(\beta_i{}^{(q+1)})}$ is the correct response for the corresponding input challenge CHAL_i. That is, if the original input q-TDABDHE challenge is drawn from P_{ABDHE}, then so are all the CHAL_is. Similarly, if Z is random in $\mathbb{G}_T$ then so is $Z^{(\beta_i{}^{(q+1)})}$, i.e. if the original challenge is drawn from R_{ABDHE} then so are all the CHAL_is.

Secondly, we note that the g' value is the same in all the n "related" challenges. This does not present a problem as g' is used only once to construct the challenge ciphertext.

- **CommonSetup:** $\mathcal{B}$ sets *params* equal to $(\mathbb{G}, \mathbb{G}_T, e, p, g)$.
- **Setup:** For each $ta_i \in \mathcal{T}$, $\mathcal{B}$ generates a random polynomial $f_i(x) \in \mathbb{Z}_p[x]$ of degree q. It sets $h_i = g^{f_i(\alpha\beta_i)}$, computing h_i from $g, g_1{}^{(\beta_i)}, g_2{}^{(\beta_i{}^2)}, \ldots, g_q{}^{(\beta_i{}^q)}$. It sets the public-key for ta_i to $mpk_i = (params, g_1{}^{(\beta_i)}, h_i, e(g,g), e(g,h_i), F_i)$ where $F_i : \mathbb{Z}_p^* \rightarrow \mathbb{G}$ is such that $F_i(x) = g_1{}^{\beta_i} \cdot g^{-x}$ and sends all the n master public-keys to $\mathcal{A}$. Since g, α are uniformly random, the β_i values and the polynomials $f_i(x)$ are chosen uniformly at random, the $g_1{}^{(\beta_i)}$ and h_i values are also uniformly random. Therefore the master public-keys have a distribution identical to that in an actual construction. (At this stage, we have essentially succeeded in using the single q-TDABDHE challenge to set up n independent TAs.)
- **Phase 1:** A makes key generation queries on (ta, id). $\mathcal{B}$ responds to a query on $ta = ta_i \in \mathcal{T}$ and $id \in \mathbb{Z}_p^*$ as follows.

  $\mathcal{B}$ checks if $g^{id} = g^{\alpha\beta_j}$ in each CHAL_j. If the equality holds, this implies that $id = \alpha\beta_j$ and $\mathcal{B}$ uses $\alpha\beta_j$ to solve the q-TDABDHE challenge immediately by computing the target response to the challenge itself.
  Else, let $F_{id,i}(x)$ denote the $q-1$ degree polynomial

  $$F_{id,i}(x) = (f_i(x) - f_i(id))/(x - id).$$

  $\mathcal{B}$ sets the private key for id in ta_i to

  $$(r_{id,i}, h_{id,i}) = (f_i(id), g^{F_{id,i}(\alpha\beta_i)}).$$

  This is a valid private key since $g^{F_{id,i}(\alpha\beta_i)} = (h_i \cdot g^{-f_i(id)})^{1/(\alpha\beta_i - id)}$.
- **Challenge:** $\mathcal{A}$ outputs TAs ta_0, ta_1 (which correspond to ta_x and $ta_y \in \mathcal{T}$ respectively), identities id_0, id_1 and messages m_0, m_1.

  Again, as in phase 1, $\mathcal{B}$ checks if either of g^{id_0} or g^{id_1} is equal to $g^{\alpha\beta_j}$ in each CHAL_j. If the equality holds, this implies that one of id_0 or id_1 is equal to $\alpha\beta_j$ and $\mathcal{B}$ uses $\alpha\beta_j$ to solve the q-TDABDHE challenge immediately by computing the target response to the challenge itself.
  Else, $\mathcal{B}$ generates a bit $b \in \{0,1\}$ and computes a private key $d_{id_b,x} = (r_{id_b,x}, h_{id_b,x})$ for id_b in ta_x if $b = 0$ or $d_{id_b,y} = (r_{id_b,y}, h_{id_b,y})$ for id_b in ta_y if $b = 1$ as in Phase 1.

Now, let $g_2(x) = x^{(q+2)}$ and $F_{2,id_b}(x) = (g_2(x) - g_2(id_b))/(x - id_b)$ which is a $(q+1)$ degree polynomial. Then $F_{2,id_b}(x)$ can be written as

$$F_{2,id_b}(x) = \sum_{i=0}^{q+1} F_{2,id_b,i} \cdot x^i = x^{q+1} + \sum_{i=0}^{q} F_{2,id_b,i} \cdot x^i$$

where $F_{2,id_b,i}$ is the coefficient of x^i in $F_{2,id_b}(x)$.
$\mathcal{B}$ sets the ciphertext $c = (u, v, w)$ as follows. In the following $\beta = \beta_x$ corresponding to ta_x if $b = 0$ or $\beta = \beta_y$ corresponding to ta_y if $b = 1$.
$\mathcal{B}$ sets $u = g'^{(g_2(\alpha\beta) - g_2(id_b))}$, $v = Z^{(\beta^{(q+1)})} \cdot e(g', \prod_{i=0}^{q} g^{F_{2,id_b,i} \cdot (\alpha\beta)^i})$ and $w = m_0/(\hat{e}(u, h_{id_0,x}) \cdot v^{r_{id_0,x}})$ if $b = 0$ and $w = m_1/(\hat{e}(u, h_{id_1,y}) \cdot v^{r_{id_1,y}})$ if $b = 1$.
To see that $c = (u, v, w)$ is a valid and appropriately distributed ciphertext when $Z = e(g_{q+1}, g')$, first let $s = \log_g(g') \cdot F_{2,id_b}(\alpha\beta)$.
Note that s is uniformly random as $\log_g(g')$ is uniformly random and the β_i values are chosen uniformly at random. We will show that c is constructed using "implicit" randomness s. Now

$$g' = g^{s/(F_{2,id_b}(\alpha\beta))} = g^{(s(\alpha\beta - id_b))/(g_2(\alpha\beta) - g_2(id_b))}.$$

Therefore, $u = g^{s(\alpha\beta - id_b)}$. If Z is a random element in $\mathbb{G}_T$ then v is random in $\mathbb{G}_T$. On the other hand, if $Z = e(g_{q+1}, g')$, then $v = e(g, g)^s$ since it can be shown that

$$e(g', \prod_{i=0}^{q} g^{F_{2,id_b,i} \cdot (\alpha\beta)^i}) = e(g', g^{F_{2,id_b}(\alpha\beta) - (\alpha\beta)^{q+1}})$$

and therefore, it can be shown that

$$v = Z^{(\beta^{(q+1)})} \cdot e(g', \prod_{i=0}^{q} g^{F_{2,id_b,i} (\alpha\beta)^i}) = e(g, g)^s.$$

Finally, note that for any private key $d_{id_b,i} = (r_{id_b,i}, h_{id_b,i})$ corresponding to $ta_i \in \mathcal{T}$, it can be shown that

$$e(u, h_{id_b,i}) \cdot v^{r_{id_b,i}} = e(g, h_i)^s.$$

Therefore, $w = m_b \cdot e(g, h_x)^{-s}$ if $b = 0$ and $w = m_b \cdot e(g, h_y)^{-s}$ if $b = 1$.

- **Phase 2:** $\mathcal{A}$ continues to make key extraction queries and $\mathcal{B}$ responds as in Phase 1.
- **Guess:** Finally, the adversary outputs a guess $b' \in \{0, 1\}$. If $b = b'$ then $\mathcal{B}$ outputs 0 indicating that $Z = e(g_{q+1}, g')$; otherwise, it outputs 1.

We have already shown that the public-keys and ciphertexts are appropriately distributed. We now show that the private keys issued by $\mathcal{B}$ are appropriately distributed as well. If I_i denotes the set consisting of $\alpha\beta_i$, id_b and all the identities queried by $\mathcal{A}$ for ta_i then $|I_i| \leq (q+1)$. Then, from $\mathcal{A}$'s view the values $\{f_i(a) : a \in I\}$ are uniformly random and independent and this follows from the fact that $f_i(x)$ is a uniformly random polynomial of degree q.

Probability Analysis: As we have already seen, if $Z = e(g_{(q+1)}, g')$, then the simulation is perfect and $\mathcal{A}$ will guess the bit b with probability $(1/2) + \epsilon'$.

On the other hand, if Z is a random element in $\mathbb{G}_T$ then, u, v are uniformly random and independent elements in $\mathbb{G}, \mathbb{G}_T$ respectively. It remains to reason about w.

Now, $w = m_b/(\hat{e}(u, h_{id_b,i}) \cdot v^{r_{id_b,i}})$ where i is x or y corresponding to ta_x or ta_y. The value in the denominator can be expressed as follows:

$$e(u, h_{id_b,i}) \cdot v^{r_{id_b,i}} = e(u, h_i)^{1/(\alpha\beta_i - id_b)} \cdot (v/e(u, g)^{1/(\alpha\beta_i - id_b)})^{f_i(id_b)}.$$

Now $f_i(id_b)$ is independent of $\mathcal{A}$'s view. Therefore as long as the inequalities $v \neq e(u, g)^{1/(\alpha\beta_x - id_0)}$, $v \neq e(u, g)^{1/(\alpha\beta_x - id_1)}$ and $v \neq e(u, g)^{1/(\alpha\beta_y - id_0)}$, $v \neq e(u, g)^{1/(\alpha\beta_y - id_1)}$ hold (and they hold with probability $(1 - 4/p)$), the value $e(u, h_{id_b,i}) \cdot v^{r_{id_b,i}}$ is random and independent of $\mathcal{A}$'s view. Consequently the value w is random and independent of $\mathcal{A}$'s view. This implies that if Z is a random element then (u, v, w) can impart no information regarding the bit b.

Assuming that no queried identity equals $\alpha\beta_j$ such that $g^{\alpha\beta_j}$ is in one of the challenges CHAL_j, (which would only increase $\mathcal{B}$'s success probability), we can see that:

$$\left|\Pr(\mathcal{B}(g', g'_{(q+2)}, g_1, g_2, \ldots, g_q, e(g_{(q+1)}, g'), Z) = 1) - 1/2\right| \leq (4/p)$$

when $(g', g'_{(q+2)}, g_1, g_2, \ldots, g_q, e(g_{(q+1)}, g'), Z)$ is sampled from R_{ABDHE}.

However,

$$\left|\Pr(\mathcal{B}(g', g'_{(q+2)}, g_1, g_2, \ldots, g_q, e(g_{(q+1)}, g'), Z) = 1) - 1/2\right| \geq \epsilon'$$

when $(g', g'_{(q+2)}, g_1, g_2, \ldots, g_q, e(g_{(q+1)}, g'), Z)$ is sampled from P_{ABDHE}.

Thus, for uniformly random g, g', α, Z we have:

$$|\Pr(\mathcal{B}(g', g'_{(l+2)}, g_1, g_2, \ldots, g_l, e(g_{(l+1)}, g'))$$

$$-\Pr(\mathcal{A}(g', g'_{(l+2)}, g_1, g_2, \ldots, g_l, Z)| \geq \epsilon' - (4/p).$$

Time-Complexity: In the simulation, $\mathcal{B}$'s overhead is dominated by computing $g^{F_{id,i}(\alpha\beta_i)}$ in response to $\mathcal{A}$'s key generation query on identity id for $ta_i \in \mathcal{T}$, where $F_{id,i}(x)$ is a polynomial of degree $(q-1)$. Each such computation requires $O(q)$ exponentiations in $\mathbb{G}$. Since $\mathcal{A}$ makes at most $(q-1)$ such queries for each TA, $t = t' + O(t_{exp} \cdot n \cdot q^2)$.

Multi-recipient Public-Key Encryption from Simulators in Security Proofs

Harunaga Hiwatari[1], Keisuke Tanaka[2],
Tomoyuki Asano[1], and Koichi Sakumoto[1]

[1] Sony Corporation
1-7-1 Konan, Minato-ku, Tokyo, 108-0075, Japan
{Harunaga.Hiwatari,Tomoyuki.Asano,Koichi.Sakumoto}@jp.sony.com
[2] Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
keisuke@is.titech.ac.jp

Abstract. In PKC 2003, Bellare, Boldyreva, and Staddon proposed the reproducibility test. The test determines whether a single-recipient public-key encryption scheme is adapted to transform into an efficient multi-recipient public-key encryption scheme. In this paper, we propose a new approach to design an efficient multi-recipient single-message public-key encryption scheme. We focus on a certain simulator which appears in the security proof of an ordinary (single-recipient) public-key encryption scheme. By considering the behavior of the simulator, we construct two efficient multi-recipient single-message public-key encryption schemes. These schemes show that there exist schemes which can be transformed into efficient multi-recipient schemes, even they do not pass the reproducibility test.

Keywords: public-key encryption, multi-recipient, broadcast encryption, simulator, IND-CCA.

1 Introduction

1.1 Background

Multi-recipient public-key encryption (PKE) is a technology to encrypt messages for several recipients. Suppose that there are n recipients. In a trivial multi-recipient PKE scheme, a ciphertext consists of n independently encrypted messages by an ordinary (single-recipient) PKE scheme. However, this trivial scheme requires n times of the encryption cost and the ciphertext length of the underlying single-recipient PKE scheme. In [13], Kurosawa constructed more efficient schemes from specific single-recipient PKE schemes by reusing the randomness across the single-recipient PKE algorithms with the distinct public keys of the recipients. This randomness-reuse technique produces efficient multi-recipient PKE schemes, however, may weaken the security.

In order to provide a generic construction of secure and efficient multi-recipient PKE schemes from single-recipient ones, Bellare, Boldyreva, and Staddon [2] introduced the notion of reproducibility. This notion is used to determine whether

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 293–308, 2009.

the randomness-reuse technique for the single-recipient PKE scheme maintains the security of the multi-recipient PKE scheme.

In [2,13], they also proposed multi-recipient single-message PKE schemes which allow a sender to broadcast a single message for all recipients. They constructed these schemes as hybrid versions.

In [15], Smart proposed the MKEM/DEM framework as a multiple-recipient version of the KEM/DEM framework. He divided a multi-recipient single-message hybrid encryption scheme into a multi-recipient key encapsulation mechanism (MKEM) and a data encapsulation mechanism (DEM). An MKEM is used to encrypt a single session key for multiple recipients.

In [2,13,15], they also proposed the MKEMs by using the same structure such that:

1. Generate a single session key at random.
2. Encrypt the session key by using a multi-recipient single-message PKE scheme.
3. Output the session key and the ciphertext of it.

In the above structure, they require a multi-recipient single-message PKE scheme.

It is the difference between a PKE and a KEM whether the encrypted data is a message chosen by sender or a random string. In general, a KEM is more efficient than a PKE scheme. If an MKEM can be constructed without using a multi-recipient single-message PKE scheme, one might design a more efficient MKEM. Two different techniques which realize such MKEMs were proposed in [1,10,16].

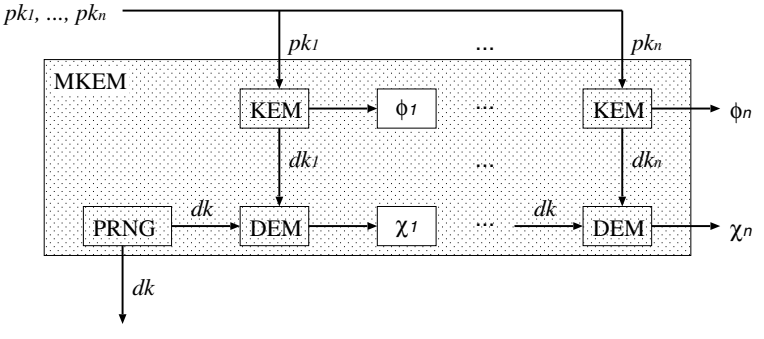


Fig. 1. The structure of the MKEM proposed in [16]

In [16], Yasuda, Kobayashi, Aoki, Fujisaki, and Fujioka proposed a generic transformation from KEMs and DEMs into an MKEM. We describe the structure of the MKEM in Fig. 1 and the procedure as follows:

1. Generate a single session key dk at random.
2. Generate multiple intermediate keys $(dk_1, dk_2, \ldots, dk_n)$ and their ciphertexts $(\phi_1, \phi_2, \ldots, \phi_n)$ by invoking a KEM multiple times with distinct user's public keys $(pk_1, pk_2 \ldots, pk_n)$.

3. Encrypt the session key dk by using a DEM with intermediate key dk_i for each $i \in \{1, \ldots, n\}$.
4. Output the session key dk, the multiple ciphertexts of the session key $(\chi_1, \chi_2, \ldots, \chi_n)$, and the ciphertexts of the intermediate keys $(\phi_1, \phi_2, \ldots, \phi_n)$.

In general, a KEM does not guarantee to generate a same session key with distinct public keys. Their idea is to encrypt a single session key with distinct intermediate keys of the users by using a DEM. This technique can be applied to any KEM. If we apply the conversion to the Kurosawa-Desmedt scheme [14], we can obtain an efficient MKEM. However, a DEM is invoked multiple times in the resulting scheme. They proved its security by using the hybrid argument, and the reduction efficiency of the security proof becomes lower in proportion to the number of the recipients.

In [1], Barbosa and Farshim proposed a KEM. The proposed KEM can guarantee to generate the same session key when using the same random coin with distinct public keys. This property allows us to convert the KEMs into the MKEMs without using a DEM. We describe the structure of MKEM in Fig. 2 and the procedure as follows:

1. Using the randomness-reuse technique, generate a session key dk and ciphertexts $(\phi_1, \phi_2, \ldots, \phi_n)$ by invoking a KEM multiple times with distinct user's public keys $(pk_1, pk_2 \ldots, pk_n)$. Note that although we do not explicitly state for illustrative purposes, one can batch the same computation and ciphertext by using the randomness-reuse technique.
2. Output the session key dk and the multiple ciphertexts of the session key $(\phi_1, \phi_2, \ldots, \phi_n)$.

Since the MKEM consists of a KEM without a DEM, one may prove the security of the MKEM without using the hybrid argument. Therefore, the reduction efficiency of the security proof may be tight. However, proving the security of the proposed KEM in [1] remains as an open problem.

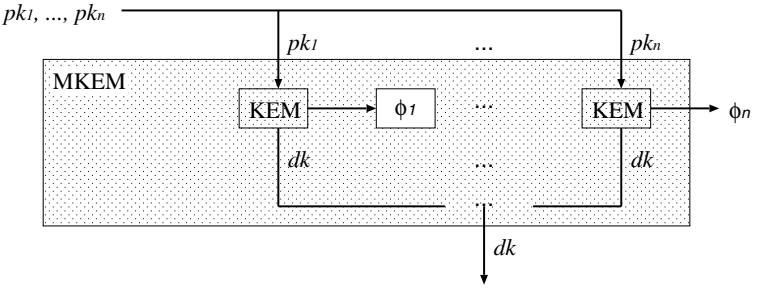


Fig. 2. The structure of the MKEM proposed in [1]

In [10], Hofheinz and Kiltz proposed a provable secure and more efficient KEM which has the above property. Both of the proposed schemes in [1] and [10] were designed by modifying specific single-recipient KEMs.

1.2 Our Contribution

We propose a new approach to design efficient MKEMs which have the structure described in Fig. 2. We focus on a certain simulator which appears in the security proof of a single-recipient PKE scheme. By observing the behavior of the simulator, we construct two efficient MKEMs based on the schemes proposed in [5] and [9], respectively.

The proposed scheme based on [5] is the most efficient scheme among the MKEMs which are secure against chosen-ciphertext attacks under the hashed decisional Diffie-Hellman (HDDH) assumption without MAC. The other proposed scheme based on [9] is one of the most efficient schemes among the MKEMs whose reduction efficiency of the security proof is tight, and which is secure against constrained chosen-ciphertext attacks introduced in [11] under the HDDH assumption.

Our constructions show that there exist single-recipient PKE schemes which can be transformed into efficient multi-recipient schemes, even they do not pass the reproducibility test.

1.3 Relation to Broadcast Encryption

MKEM is related to a previously proposed primitive called public-key broadcast encryption [3]. This primitive requires the key generation center to generate public/secret key pairs of all users at the setup phase. In MKEM, each user generates its own public/secret key pair by themselves. Therefore, users can join the system any time without help of the third party. On the other hand, in many broadcast encryption schemes, the number of users must be fixed at the setup phase. There are only few schemes with the dynamic join property, e.g., recently proposed scheme in [8]. Even in these schemes, users must require the key generation center to generate their own key pairs when they join the system.

1.4 Organization

The organization of this paper is as follows. In Section 2, we review some definitions. In Section 3, we propose a new approach to design efficient MKEMs. In Section 4 and 5, we construct new schemes. We conclude in Section 6.

2 Preliminaries

2.1 Hashed Decisional Diffie-Hellman Assumption

In this section, we review the definition of the hashed decisional Diffie-Hellman (HDDH) assumption.

Let $\mathbb{G}$ be an abelian group of order p, where p is a large prime, and let $h : \mathbb{G} \to \mathcal{D}$ be a function.

Definition 1. *For a probabilistic polynomial-time algorithm $\mathcal{A}$ that outputs a single bit, we define $Adv_{hddh,\mathcal{A}}$ to be*

$$|\Pr_{g\in_R\mathbb{G},x,y\in_R\mathbb{Z}_p}[\mathcal{A}(g,g^x,g^y,h(g^{xy}))=1] - \Pr_{g\in_R\mathbb{G},x,y\in_R\mathbb{Z}_p,Z\in_R\mathcal{D}}[\mathcal{A}(g,g^x,g^y,Z)=1]|.$$

The hashed decisional Diffie-Hellman (HDDH) *assumption is that, for any $\mathcal{A}$, $Adv_{hddh,\mathcal{A}}$ is at most ϵ_{hddh} as a negligible function of the security parameter.*

If h is the identity function from $\mathbb{G}$ to $\mathbb{G}$, the above assumption is identical to the HDDH assumption. It is known that the HDDH assumption is not stronger than the DDH assumption.

2.2 Target Collision Resistant Hash Function

In this section, we review the definition of the target collision resistant hash function.

Definition 2. *For a probabilistic polynomial-time algorithm $\mathcal{A}$, we define $Adv_{tcr,\mathcal{A}}$ to be*

$$\Pr_{x\in_R\mathbb{G},h\in_R\mathcal{H}}[\mathcal{A}(x,h)=y|h(x)=h(y)]$$

where $h \in \mathcal{H}$ is a hash function and $\mathbb{G}$ is a domain of this function. $\mathcal{H}$ is target collision resistant family of hash functions if for any $\mathcal{A}$, $Adv_{tcr,\mathcal{A}}$ is at most ϵ_{tcr} as a negligible function of the security parameter.

2.3 Key Encapsulation Mechanisms

In this section, we review the KEM/DEM framework which is used for construction of secure hybrid encryption schemes [7].

Hybrid encryption schemes are a kind of PKE scheme which use PKE techniques for key distribution and secret-key encryption techniques for message encryption. If we encrypt a message using a hybrid encryption scheme, we take the two steps as follows.

1. Generate a random session key and a ciphertext of it using public key encryption techniques with recipient's public key.
2. Encrypt the message using symmetric key encryption techniques with the session key which is generated in step 1.

Cramer and Shoup showed that these two steps can be separated and security criteria can be defined for each section individually [7]. The first and second steps are known as key encapsulation mechanisms (KEM) and data encapsulation mechanisms (DEM), respectively.

The KEMs consist of the following algorithms. Through this paper, we use λ and $\mathcal{K}_D$ to describe a security parameter and a key space of DEM, respectively.

- A probabilistic polynomial-time common-key generation algorithm $\mathcal{G}_{kem}$, which takes as input a security parameter 1^λ, outputs a common key I. For given I, a randomness space $\mathcal{R}(I)$ is uniquely determined.
- A probabilistic polynomial-time key generation algorithm $\mathcal{K}_{kem}$, which takes as input a common key I, outputs a public/secret key pair (pk, sk).
- A probabilistic polynomial-time encryption algorithm $\mathcal{E}_{kem}$, which takes as input a common key I and a public key pk, outputs a pair (dk, ϕ) where dk is a key for DEM and ϕ is a ciphertext of dk, using a random coin $r \in_R \mathcal{R}(I)$.
- A deterministic polynomial-time decryption algorithm $\mathcal{D}_{kem}$, which takes as input a common key I, a secret key sk and a ciphertext ϕ, outputs a key dk or the special symbol **reject**.

A common-key generation algorithm is usually omitted or included in a key generation algorithm. In this paper, for convenience of explanation, we explicitly describe the common-key generation algorithm. A common key I may include the description of group and hash function.

We require that, for any $\lambda \in \mathbb{N}$, if $I \leftarrow \mathcal{G}_{kem}(1^\lambda)$ and $(pk, sk) \leftarrow \mathcal{K}_{kem}(I)$,

$$\Pr_{r \in_R \mathcal{R}(I)}[dk \neq \mathcal{D}_{kem}(sk, \phi, I) | (dk, \phi) \leftarrow \mathcal{E}_{kem}(pk, I; r)] \leq \epsilon,$$

where ϵ denotes negligible.

We review the indistinguishability against chosen ciphertext attacks (IND-CCA) for KEM.

Definition 3. *Let $\Pi_{kem} = (\mathcal{G}_{kem}, \mathcal{K}_{kem}, \mathcal{E}_{kem}, \mathcal{D}_{kem})$ be a* KEM *and let $\mathcal{A}$ be an adversary. For $\lambda \in \mathbb{N}$, we define the advantage of $\mathcal{A}$ as*

$$\boldsymbol{Adv}^{\text{ind-cca}}_{\Pi_{kem},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\text{ind-cca-0}}_{\Pi_{kem},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ind-cca-1}}_{\Pi_{kem},\mathcal{A}}(\lambda) = 1]|$$

where, for $b \in \{0, 1\}$,

$$\begin{aligned}
&\text{Experiment } \mathsf{Exp}^{\text{ind-cca-b}}_{\Pi_{kem},\mathcal{A}}(\lambda)\\
&\quad I \leftarrow \mathcal{G}_{kem}(1^\lambda);\ (pk, sk) \leftarrow \mathcal{K}_{kem}(I);\\
&\quad (dk_0, \phi^*) \leftarrow \mathcal{E}_{kem}(pk, I);\ dk_1 \in_R \mathcal{K}_D;\ b \in_R \{0, 1\};\\
&\quad d \leftarrow \mathcal{A}^{\mathcal{O}}(dk_b, \phi^*, pk, I);\\
&\quad \textbf{\textit{return}}\ d
\end{aligned}$$

where $\mathcal{O}$ answers $\mathcal{D}_{kem}(sk, \phi, I)$, when $\mathcal{A}$ queries ϕ $(\neq \phi^*)$. We say that Π_{kem} is secure in the sense of IND-CCA*, if for any $\mathcal{A}$, $\boldsymbol{Adv}^{\text{ind-cca}}_{\Pi_{kem},\mathcal{A}}(\lambda)$ is negligible.*

2.4 Multi-recipient KEMs

In this section, we review the definition of MKEMs which are introduced by Smart [15]. A multi-recipient hybrid encryption scheme consists of MKEM and DEM. The MKEMs are defined similarly to KEMs, however, the encryption algorithm takes multiple public keys $\boldsymbol{pk}$ $(:= (pk_1, pk_2, \ldots, pk_n))$ as input.

- A probabilistic polynomial-time common-key generation algorithm $\mathcal{G}_{mkem}$, which takes as input a security parameter 1^λ, outputs a common key I.
- A probabilistic polynomial-time key generation algorithm $\mathcal{K}_{mkem}$, which takes as input a common key I, outputs a public/secret key pair (pk, sk).
- A probabilistic polynomial-time encryption algorithm $\mathcal{E}_{mkem}$, which takes as input a common key I and a vector of public keys $\boldsymbol{pk} = (pk_1, pk_2, \ldots, pk_n)$, outputs a pair (dk, ϕ) where dk is a key for DEM and ϕ is a ciphertext of dk.
- A deterministic polynomial-time decryption algorithm $\mathcal{D}_{mkem}$, which takes as input a common key I, a ciphertext ϕ, a secret key sk_i, and an index i which corresponds to the index of pk_i used in the $\mathcal{E}_{mkem}$ algorithm, outputs a key dk or the special symbol **reject**.

We review IND-CCA for MKEM.

Definition 4. *Let $\Pi_{mkem} = (\mathcal{G}_{mkem}, \mathcal{K}_{mkem}, \mathcal{E}_{mkem}, \mathcal{D}_{mkem})$ be a* MKEM *and let $\mathcal{A}$ be an adversary. For $\lambda \in \mathbb{N}$, we define the advantage of $\mathcal{A}$ as*

$$\boldsymbol{Adv}^{\text{ind-cca}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\text{ind-cca-0}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ind-cca-1}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = 1]|$$

where, for $b \in \{0, 1\}$,

$$\begin{aligned}
&\text{Experiment } \mathsf{Exp}^{\text{ind-cca-b}}_{\Pi_{mkem},\mathcal{A}}(\lambda)\\
&\quad I \leftarrow \mathcal{G}_{mkem}(1^\lambda);\\
&\quad \textbf{for } i = 1, \ldots, n\\
&\qquad (pk_i, sk_i) \leftarrow \mathcal{K}_{mkem}(I);\\
&\quad (dk_0, \phi^*) \leftarrow \mathcal{E}_{mkem}(\boldsymbol{pk}, I);\ dk_1 \in_R \mathcal{K}_D;\ b \in_R \{0,1\};\\
&\quad d \leftarrow \mathcal{A}^{\mathcal{O}}(dk_b, \phi^*, \boldsymbol{pk}, I);\\
&\quad \textbf{\textit{return}}\ d
\end{aligned}$$

where $\mathcal{O}$ is the decryption oracle. It answers $\mathcal{D}_{mkem}(i, sk_i, \phi, I)$, when $\mathcal{A}$ queries (i, ϕ) such that the ciphertext ϕ does not contain a target ciphertext for user i. We say that Π_{mkem} is secure in the sense of IND-CCA*, if for any $\mathcal{A}$, $\boldsymbol{Adv}^{\text{ind-cca}}_{\Pi_{mkem},\mathcal{A}}(\lambda)$ is negligible.*

2.5 Constrained Chosen-Ciphertext Security for MKEMs

Hofheinz and Kiltz [11,10] proposed a relaxed security notion of IND-CCA, which is called indistinguishable against constrained chosen ciphertext attacks (IND-CCCA). Intuitively, an adversary $\mathcal{A}$ is only allowed to make a decryption query when $\mathcal{A}$ already has some a priori knowledge about the session key. They denoted the priori knowledge by an efficiently computable boolean predicate $pred : \mathcal{K}_D \to \{0, 1\}$. If $pred(dk) = 1$, then the session key is returned, and $\perp$ otherwise. They showed that though IND-CCCA was weaker than IND-CCA, IND-CCCA KEM was sufficient for constructing a secure hybrid encryption scheme if authenticated symmetric encryption scheme was used as a DEM.

We can easily extend the above composition theorem for MKEM/DEM framework. The IND-CCCA security for MKEM is defined as follows.

Definition 5. *Let* $\Pi_{mkem} = (\mathcal{G}_{mkem}, \mathcal{K}_{mkem}, \mathcal{E}_{mkem}, \mathcal{D}_{mkem})$ *be a* MKEM *and let* $\mathcal{A}$ *be an adversary. For* $\lambda \in \mathbb{N}$*, we define the advantage of* $\mathcal{A}$ *as*

$$\boldsymbol{Adv}^{\text{ind-ccca}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{\text{ind-ccca-0}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ind-ccca-1}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = 1]|$$

where, for $b \in \{0, 1\}$*,*

$$\begin{aligned}
&\text{Experiment } \mathsf{Exp}^{\text{ind-ccca-b}}_{\Pi_{mkem},\mathcal{A}}(\lambda)\\
&\quad I \leftarrow \mathcal{G}_{mkem}(1^{\lambda});\\
&\quad \textbf{for } i = 1, \ldots, n\\
&\qquad (pk_i, sk_i) \leftarrow \mathcal{K}_{mkem}(I);\\
&\quad (dk_0, \phi^*) \leftarrow \mathcal{E}_{mkem}(\boldsymbol{pk}, I);\ dk_1 \in_R \mathcal{K}_D;\ b \in_R \{0,1\};\\
&\quad d \leftarrow \mathcal{A}^{\mathcal{CO}}(dk_b, \phi^*, \boldsymbol{pk}, I);\\
&\quad \textbf{\textit{return}}\ d
\end{aligned}$$

where $\mathcal{CO}$ *is the constrained decryption oracle. When* $\mathcal{A}$ *issues a query* $(i, \phi, pred)$ *to* $\mathcal{CO}$*, if* $pred(\mathcal{D}_{mkem}(i, sk_i, \phi, I)) = 1$*, then the session key is returned, and* $\perp$ *otherwise. We define* $uncert_{\mathcal{A}} = (1/Q_d)\sum_{1 \le j \le Q_D} \Pr_{dk \in_R \mathcal{K}_D}[pred_j(dk) = 1]$*. The adversary* $\mathcal{A}$ *must control* $uncert_{\mathcal{A}}$ *to be negligible. We say that* Π_{mkem} *is secure in the sense of* IND-CCCA*, if for any* $\mathcal{A}$*,* $\boldsymbol{Adv}^{\text{ind-ccca}}_{\Pi_{mkem},\mathcal{A}}(\lambda)$ *is negligible.*

3 Our Approach for Construction of MKEMs

Bellare, Boldyreva, and Staddon [2] introduced the notion of reproducibility. Intuitively, if a PKE scheme is reproducible, one can convert a ciphertext which is an encryption of message m_1 for user u_a into another ciphertext which is an encryption of message m_2 for user u_b without changing a randomness used in these ciphertexts.

This notion is used to determine whether the randomness-reuse technique for the single-recipient PKE scheme maintains the security of the multi-recipient PKE scheme. The reproducibility test is useful, however the reduction efficiency of the security proof of the multi-recipient PKE scheme becomes lower in proportion to the number of recipients. They also proposed concrete schemes whose reduction efficiency is tight by using the random self-reducibility of the discrete logarithm problem, however these multi-recipient PKE schemes are not obtained through the reproducibility test.

Barbosa and Farshim [1] proposed weakly reproducibility for multi-recipient single-message PKE schemes, by restricting the reproducibility test to be used in the single message setting. They also proposed the direct reproducibility test to construct schemes whose reduction efficiency is tight. However, all of the above

reproducibility tests are not used to construct efficient MKEMs, since a session key of each user becomes different, even using the randomness-reuse technique.

We observe that the above proposed reproducibility tests are designed based on the behavior of a simulator which appears in the security proof of PKE schemes. The simulator runs as follows:

1. Obtain an instance of the problem that the simulator tries to solve (e.g. the DDH instance).
2. Generate public key from the instance as an input to an adversary against the PKE.
3. Construct a target ciphertext from the instance and the public key.

In fact, if a scheme passes the test, we can construct another simulator which given a public key, can simulate the environment of an adversary against the multi-recipient scheme.

Although the previous results capture the above behavior of the simulator, we consider that they do not capture another type of simulator which uses an algebraic trick from selective-ID secure identity-based encryption. We call such a simulator the sID simulator. It has a feature that it can simulate the decryption oracle without a real secret key. It runs as follows:

1. Obtain an instance.
2. Generate a part of a target ciphertext from the instance.
3. Generate a public key with a witness from the instance and the part of the target ciphertext.
4. Construct an entire target ciphertext.

At first, given an instance, the sID simulator generates a part of a target ciphertext. Next, it generates a public key and some witness from the generated part of the target ciphertext and then constructs the entire target ciphertext. It can simulate a decryption oracle by using the witness. Though the witness is not a secret key, it enables the simulator to decrypt almost all of the ciphertexts. This technique is used in the proof of the most recently proposed PKE schemes [5,9,10,12].

Our approach for construction of an efficient MKEM applies the behavior of the sID simulator as key generation and decryption algorithms of KEM. In general, an MKEM can be designed on the basis of a KEM, by sharing a partial public key as a common public key among the users and by using the randomness-reuse technique. Sharing as many parts of public keys as possible for the purpose of batch process results in more efficient MKEMs. However, the users can share only some restricted parts of public keys which do not correspond to any individual secret information of users. We find that the behavior of the sID simulator is useful for the achievement of the above purpose.

Recall that in a security proof of PKE schemes, the sID simulator sets a part of an instance of a problem (e.g. the DDH problem) as a part of a public key and generates the entire public key and some witness. It simulates the decryption oracle by using the witness instead of a secret key (e.g. the discrete logarithm of the part of the instance) corresponding to the partial public key.

Table 1. The table intuitively shows the relation ship between items used in the sID simulator and the proposed MKEM

sID simulator	proposed MKEM
a part of instance	common key
public key	individual public key
witness	secret key
a part of target ciphertext	a part of secret key

behavior of the simulator	MKEM based on the simulator
how to generate a public key with a witness	key generation algorithm
how to simulate the decryption oracle	decryption algorithm

In the multi-recipient setting, we set the partial public key (of the sID simulator) as a common public key, and generates each user's individual public key and secret key using the common public key and the witness. Therefore, if users mimic the behavior of the sID simulator, they can decrypt almost all of the ciphertexts by using the secret key. Only the ciphertexts a user can not decrypt are ones using the same random value as selected by the user in the key generation phase, and the probability is $1/p$.

4 Our Scheme: Cash-Kiltz-Shoup Variant

In this section, we propose an MKEM based on the KEM [5]. The proposed scheme is the most efficient scheme among the MKEMs which is secure under the HDDH assumption without MAC. We construct the scheme from the behavior of the simulator which appears in the security proof [5].

$\mathcal{G}_{mkem}(1^\lambda)$**:** Choose a multiplicative group $\mathbb{G}$ with prime order p, a target collision resistant hash function $\mathsf{TCR} : \mathbb{G} \to \mathbb{Z}_p^*$, and a hash function $h : \mathbb{G} \to \mathcal{K}_D$. Then, pick two generators $g, c_1 \in_R \mathbb{G}$ at random and set the common key $I := (p, \mathbb{G}, g, c_1, TCR, h)$.

$\mathcal{K}_{mkem}(I)$**:** Pick $s, t, w_1, w_2, \hat{\alpha} \in_R \mathbb{Z}_p$ at random, compute $c_2 \leftarrow g^s c_1^{-t}$, $d_1 \leftarrow c_1^{-\hat{\alpha}} g^{w_1}$, and $d_2 \leftarrow c_2^{-\hat{\alpha}} g^{w_2}$. Then, set the public key $pk := (c_2, d_1, d_2)$ and the secret key $sk := (s, t, w_1, w_2, \hat{\alpha})$.

$\mathcal{E}_{mkem}(\boldsymbol{pk}, I)$**:** Pick $r \in_R \mathbb{Z}_p$ at random and compute $u \leftarrow g^r$ and $\alpha \leftarrow \mathsf{TCR}(u)$. Then, parse $\boldsymbol{pk}$ as each user's public key $(pk_1, pk_2, \ldots, pk_n)$, and for $i = 1, 2, \ldots, n$, parse pk_i as $(c_{2,i}, d_{1,i}, d_{2,i})$ and compute $v_{1,i} \leftarrow (c_1^\alpha d_{1,i})^r$ and $v_{2,i} \leftarrow (c_{2,i}^\alpha d_{2,i})^r$ for each user. Finally, compute the data key $dk \leftarrow h(c_1^r)$ and set the ciphertext $\phi := (u, \boldsymbol{v}_1, \boldsymbol{v}_2)$ where $\boldsymbol{v}_j := (v_{j,1}, v_{j,2}, \ldots, v_{j,n})$ for $j \in \{1, 2\}$.

$\mathcal{D}_{mkem}(i, sk_i, \phi, I)$**:** Parse ϕ as $(u, \boldsymbol{v}_1, \boldsymbol{v}_2)$, compute $\alpha \leftarrow \mathsf{TCR}(u)$, and check whether $\alpha \stackrel{?}{=} \hat{\alpha}$. If satisfied, output the special symbol **reject**. Otherwise parse $\boldsymbol{v}_j$ as $(v_{j,1}, v_{j,2}, \ldots, v_{j,n})$ for $j \in \{1, 2\}$, compute $\tilde{v}_1 \leftarrow (v_{1,i} u^{-w_1})^{1/(\alpha - \hat{\alpha})}$ and $\tilde{v}_2 \leftarrow (v_{2,i} u^{-w_2})^{1/(\alpha - \hat{\alpha})}$, and check whether $\tilde{v}_1^t \tilde{v}_2 \stackrel{?}{=} u^s$. If not, output **reject**. Otherwise, compute the session key $dk \leftarrow h(\tilde{v}_1)$.

Note that the event that **reject** $\leftarrow \mathcal{D}_{kem}(i, sk_i, \phi, I)$ such that $(dk, \phi) \leftarrow \mathcal{E}_{kem}(\boldsymbol{pk}, I)$ occurs with negligible probability. Firstly, to show the security of above scheme, we review the following lemma which is introduced in [5].

Lemma 1 (Trapdoor Test [5]). *Let $\mathbb{G}$ be a cyclic group of prime order p, and g be a element in $\mathbb{G}$. Suppose $c_1 \in_R \mathbb{G}$ and $s, t \in_R \mathbb{Z}_p$. We define $c_2 := g^s c_1^{-t}$ and denote u, v_1, v_2 are elements in $\mathbb{G}$. Then we have: (1) c_2 is uniformly distributed over $\mathbb{G}$; (2) c_1 and c_2 are independent; (3) if $c_1 = g^{x_1}$ and $c_2 = g^{x_2}$, then the probability that the truth value of $v_1^t v_2 \stackrel{?}{=} u^s$ does not agree with the truth value of $v_1 \stackrel{?}{=} u^{x_1} \wedge v_2 \stackrel{?}{=} u^{x_2}$ is at most $1/p$.*

Theorem 1. *Let $\mathbb{G}$ be an abelian group of order p, where p is a large prime, and let* TCR *be a target collision resistant hash function. Then, the above scheme is secure in the sense of* IND-CCA *under the HDDH assumption on a hash function h. In particular, for any adversary $\mathcal{A}$ which makes at most Q_d decryption queries,*

$$\boldsymbol{Adv}^{\text{ind-cca}}_{\Pi_{mkem}, \mathcal{A}}(\lambda) \leq \epsilon_{tcr} + \epsilon_{hddh} + Q_d/p.$$

Proof. We consider a sequence of games. In this proof, S_i denotes the event that $b = d$ in Game i and $\phi = (u^*, \boldsymbol{v}_1^*, \boldsymbol{v}_2^*)$ and α^* denote the target ciphertext and $\mathsf{TCR}(u^*)$.

Game 0. Let Game 0 be the original game.

$$\mathbf{Adv}^{\text{ind-cca}}_{\Pi_{mkem}, \mathcal{A}}(\lambda) = |\Pr[S_0] - 1/2|.$$

Game 1. Let Game 1 be like Game 0, but with the following difference. If an adversary asks a ciphertext containing $u \neq u^*$ such that $\mathsf{TCR}(u) = \mathsf{TCR}(u^*)$, Game 1 aborts. Using a property of target collision resistant hash functions, it is easy to show that

$$|\Pr[S_1] - \Pr[S_0]| \leq \epsilon_{tcr}.$$

Game 2. Let Game 2 be like Game 1, but with the following difference. For computing the secret key, instead of choose $\hat{\alpha}_i$ at random for $i = 1, 2, \ldots, n$, the experiment picks r^* at random, computes $u^* \leftarrow g^{r^*}$, and sets $\hat{\alpha_i} \leftarrow \alpha^* (:= \mathsf{TCR}(u^*))$ for all i. Moreover, we modify the decryption oracle. If the adversary asks a ciphertext containing u^*, then the decryption oracle immediately outputs **reject**. Using Lemma 1, we show that

$$|\Pr[S_2] - \Pr[S_1]| \leq Q_d/p.$$

Even if $\hat{\alpha}_i$ is fixed, $c_{2,i}, d_{1,i}$, and $d_{2,i}$ are uniformly distributed over $\mathbb{G}$. The distribution of public keys in Game 2 is same as that in Game 1. From the view of adversary, for all i, there are many potential secret key corresponding to the given public key. For any potential secret key $(w_{1,i}, w_{2,i}, \hat{\alpha}_i)$ except for $(\cdot, \cdot, \alpha^*)$, the decryption algorithm would compute $\tilde{v}_1 \leftarrow (v_{1,i}(u^*)^{-w_{1,i}})^{1/(\alpha^* - \hat{\alpha}_i)}$ and $\tilde{v}_2 \leftarrow (v_{2,i}(u^*)^{-w_{2,i}})^{1/(\alpha^* - \hat{\alpha}_i)}$ for a ciphertext containing u^*. If $\tilde{v}_1^{t_i}\tilde{v}_2 = (u^*)^{s_i}$, Lemma 1 guarantees with overwhelming probability that $(u^*)^{\log_g c_{1}} = \tilde{v}_1$ and $(u^*)^{\log_g c_{2,i}} = \tilde{v}_2$.

$$
\begin{aligned}
& (u^*)^{\log_g c_1} = \tilde{v}_1 && \wedge && (u^*)^{\log_g c_{2,i}} = \tilde{v}_2 \\
\iff\ & c_1^{r^*(\alpha^* - \hat{\alpha}_i)} = (v_{1,i}(g^{-w_{1,i}})^{r^*}) && \wedge && c_{2,i}^{r^*(\alpha^* - \hat{\alpha}_i)} = (v_{2,i}(g^{-w_{2,i}})^{r^*}) \\
\iff\ & v_{1,i} = (c_1^{\alpha^*} c_1^{-\hat{\alpha}_i} g^{w_{1,i}})^{r^*} && \wedge && v_{2,i} = (c_{2,i}^{\alpha^*} c_{2,i}^{-\hat{\alpha}_i} g^{w_{2,i}})^{r^*} \\
\iff\ & v_{1,i} = (c_1^{\alpha^*} d_{1,i})^{r^*} && \wedge && v_{2,i} = (c_{2,i}^{\alpha^*} d_{2,i})^{r^*} \\
\iff\ & v_{1,i} = v_{1,i}^* && \wedge && v_{2,i} = v_{2,i}^*
\end{aligned}
$$

Therefore, for any potential secret key, the probability that the decryption algorithm does not output **reject** for a ciphertext containing u^* queried by an adversary is at most $1/p$.

Furthermore, we show that

$$|\Pr[S_2] - 1/2| \leq \epsilon_{hddh}.$$

We assume that there is an adversary $\mathcal{A}$ which breaks Game 2. We construct an algorithm $\mathcal{B}$ which solves the HDDH problem using the adversary $\mathcal{A}$. For a given HDDH instance $(g, g^{\bar{a}}, g^{\bar{b}}, Z)$, $\mathcal{B}$ sets $u^* := g^{\bar{a}}$ and $c_1 := g^{\bar{b}}$. The algorithm $\mathcal{B}$ slightly changes the key generation algorithm to always set $\hat{\alpha}_i \leftarrow \alpha^* (:= \mathsf{TCR}(u^*))$ for all i. Then, for each i, $\mathcal{B}$ computes $v_{1,i}^* \leftarrow (u^*)^{w_{1,i}} (:= (c_1^{\alpha^*} d_{1,i})^{\bar{a}})$ and $v_{2,i}^* \leftarrow (u^*)^{w_{2,i}} (:= (c_{2,i}^{\alpha^*} d_{2,i})^{\bar{a}})$ and sets $(u^*, \boldsymbol{v}_1^*, \boldsymbol{v}_2^*)$ as a target ciphertext. Next, $\mathcal{B}$ provides $\mathcal{A}$ with $(Z, (u^*, \boldsymbol{v}_1^*, \boldsymbol{v}_2^*))$. If $\mathcal{A}$ makes decryption query, $\mathcal{B}$ simulates the decryption oracle in Game 2 by using secret keys that $\mathcal{B}$ generates. Finally, $\mathcal{A}$ outputs a bit d as his guess, and $\mathcal{B}$ outputs the same bit d. □

5 Our Scheme: Hanaoka-Kurosawa Variant

In this section, we propose an MKEM based on the KEM [9]. The proposed scheme is one of the most efficient schemes among the MKEMs whose reduction efficiency is tight and which is secure under the HDDH assumption. Similar to our scheme based on [5], we construct the scheme from the behavior of the simulator which appears in the security proof [9].

$\mathcal{G}_{mkem}(1^\lambda)$**:** Choose a multiplicative group $\mathbb{G}$ with prime order p, a target collision resistant hash function $\mathsf{TCR} : \mathbb{G} \to \mathbb{Z}_p^*$, and a hash function $h : \mathbb{G} \to \mathcal{K}_D$. Then, pick two generators $g, y_0 \in_R \mathbb{G}$ at random and set the common key $I := (p, \mathbb{G}, g, y_0, TCR, h)$.

$\mathcal{K}_{mkem}(I)$**:** Pick $s, t, F_s, F_t \in_R \mathbb{Z}_p$ at random, define the quadratic polynomial $f(x) := a_0 + a_1 x + a_2 x^2$ over $\mathbb{Z}_p$ such that $(f(0), f(s), f(t)) = (\log_g y_0, F_s, F_t)$, and for $i \in \{1, 2\}$, compute $y_i \leftarrow g^{a_i}$ without the knowledge of a_i. Then, set the public key $pk := (y_1, y_2)$ and the secret key $sk := (s, t, F_s, F_t)$.

$\mathcal{E}_{mkem}(\boldsymbol{pk}, I)$**:** Pick $r \in_R \mathbb{Z}_p$ at random and compute $u \leftarrow g^r$ and $\alpha \leftarrow \mathsf{TCR}(u)$. Then, parse $\boldsymbol{pk}$ as each user's public key $(pk_1, pk_2, \ldots, pk_n)$, and for $i = 1, \ldots, n$, parse pk_i as $(y_{1,i}, y_{2,i})$ and compute $v_i \leftarrow y_0^r y_{1,i}^{r\alpha} y_{2,i}^{r\alpha^2} (:= u^{f_i(\alpha)})$ for each user. Finally, compute the data key $dk \leftarrow h(y_0^r)$ and set the ciphertext $\phi := (u, v_1, v_2, \ldots, v_n)$.

$\mathcal{D}_{mkem}(i, sk_i, \phi, I)$**:** Parse ϕ as $(u, v_1, v_2, \ldots, v_n)$, compute $\alpha \leftarrow \mathsf{TCR}(u)$, and check whether $\alpha \in \{s, t\}$. If satisfied, output the special symbol **reject**. Otherwise, define the quadratic polynomial $f'(x)$ over $\mathbb{Z}_p$ such that $(f'(\alpha), f'(s), f'(t)) = (\log_u v_i, F_s, F_t)$ and compute the data key $dk \leftarrow h(u^{f'(0)})$ by using the Lagrange interpolation method.

Theorem 2. *Let $\mathbb{G}$ be an abelian group of order p, where p is a large prime, and let* TCR *be a target collision resistant hash function. Then, the above scheme is secure in the sense of* IND-CCCA *under the HDDH assumption on a hash function h. In particular, for any adversary $\mathcal{A}$ which makes at most Q_d decryption queries,*

$$\boldsymbol{Adv}^{\text{ind-ccca}}_{\Pi_{mkem},\mathcal{A}}(\lambda) \leq \epsilon_{tcr} + \epsilon_{hddh} + uncert_{\mathcal{A}} \cdot Q_d.$$

The security proof is almost the same proof in [9]. However, unlike the original scheme, invalid ciphertexts only get rejected implicitly using the security properties of DEM in the above proposed scheme. When an adversary sends a query including a part of target ciphertext, the analysis is slightly different from the proof in the original scheme.

Proof. We consider a sequence of games. In this proof, S_i denotes the event that $b = d$ in Game i and $\phi = (u^*, v_1^*, v_2^*, \ldots, v_n^*)$ and α^* denote the target ciphertext and $\mathsf{TCR}(u^*)$.

Game 0. Let Game 0 be the original game.

$$\mathbf{Adv}^{\text{ind-cca}}_{\Pi_{mkem},\mathcal{A}}(\lambda) = |\Pr[S_0] - 1/2|.$$

Game 1. Let Game 1 be like Game 0, but with the following difference. If an adversary asks a ciphertext containing $u \neq u^*$ such that $\mathsf{TCR}(u) = \mathsf{TCR}(u^*)$, Game 1 aborts. Using a property of target collision resistant hash functions, it is easy to show that

$$|\Pr[S_1] - \Pr[S_0]| \leq \epsilon_{tcr}.$$

Game 2. Let Game 2 be like Game 1, but with the following difference. For computing the secret key, instead of choose s_i at random for $i = 1, 2, \ldots, n$, the experiment picks r^* at random, computes $u^* \leftarrow g^{r^*}$, and sets $s_i \leftarrow$

$\alpha^*(:= \mathsf{TCR}(u^*))$ for all i. Moreover, we modify the decryption oracle. If the adversary asks a ciphertext containing u^*, then the decryption oracle immediately outputs **reject**. We show that

$$|\Pr[S_2] - \Pr[S_1]| \leq uncert_{\mathcal{A}} \cdot Q_d.$$

Suppose $(i, (u, v_1, v_2, \ldots, v_n), pred)$ is a query of $\mathcal{A}$ such that $pred(h(u^{f_i'(0)})) = 1$, but $u^{f_i(0)} \neq v_i$. We notice that for any $f(x)$, α^*, and $\alpha(:= \mathsf{TCR}(u))$, the value $f'(0)$ takes many different points according to different points for t_i included in the secret key of user i. This can be proven by a contradiction as follows: Fix $f(x)$, α^*, α, and $F'_\alpha \neq f(\alpha)$. For $t_i \in (\mathbb{Z}_p \setminus \{\alpha^*, \alpha\})$, let $f_{t_i}(x)$ be a polynomial of degree at most two such that $f_{t_i}(\alpha^*) = f(\alpha^*)$, $f_{t_i}(\alpha) = F'_\alpha$, and $f_{t_i}(t_i) = f(t_i)$. Then, we will show that for any $(t_i, \bar{t}_i) \in (\mathbb{Z}_p \setminus \{\alpha^*, \alpha\})^2$ and $t_i \neq \bar{t}_i$, $f_{t_i}(0) \neq f_{\bar{t}_i}(0)$. Suppose that $f_{t_i}(0) = f_{\bar{t}_i}(0)$. Then $f_{t_i}(x) = f_{\bar{t}_i}(x)$ because they intersect at three points, $x = 0$, α^* and α. In this case, $f(x) = f_{t_i}(= f_{\bar{t}_i})$ since they intersect at three points, $x = \alpha^*$, t_i, and $\bar{t}_i$. However, this is a contradiction since $f_{t_i}(\alpha) = F'_\alpha \neq f(\alpha)$. Hence, even if $\mathcal{A}$ has unlimited computational power, the event that $pred(h(u^{f'(0)})) = 1$ occurs with the probability of $uncert_{\mathcal{A}}$.

Furthermore, we show that

$$|\Pr[S_2] - 1/2| \leq \epsilon_{hddh}.$$

We assume that there is an adversary $\mathcal{A}$ which breaks Game 2. We construct an algorithm $\mathcal{B}$ which solves the HDDH problem using the adversary $\mathcal{A}$. For a

Table 2. Comparison for IND-CCA secure MKEMs. For efficiency, we count the number of pairings, multi (or sequential)-exponentiations, and single-exponentiations for encryption and decryption. For ciphertext length, we denote the length of a group, an authentication tag, and ciphertext length of secret key encryption scheme by $|\mathbb{G}|$, $|mac|$, and $|ske|$, respectively. We explicitly separate a DEM into an authentication tag and ciphertext of secret key encryption scheme. For comparison with other schemes, we apply randomness-reuse technique and transformation technique introduced in [16] to original KEMs. The schemes applied transformation technique which can convert KEM into MKEM by using DEMs have a property that reduction efficiency becomes lower in proportion to the number of recipients.

based scheme	security assumption	reduction cost	encryption #pairings + #[multi, single]-exp	decryption #pairings + #[multi, single]-exp	key size (I/pk/sk)	ciphertext length
CS98 [6]	DDH	1	$0 + [n, n+2]$	$0 + [1, 1]$	2/3/5	$2n\|\mathbb{G}\|$
KD04 [14]	DDH	$1/n$	$0 + [n, 2]$	$0 + [1, 0]$	2/2/4	$2\|\mathbb{G}\| + n\|mac\| + n\|ske\|$
BMW05 [4]	BDH	$1/n$	$0 + [n, n+1]$	$1 + [1, 0]$	2/3/3	$n\|\mathbb{G}\| + n\|ske\|$
Kiltz07 [12]	GHDDH	$1/n$	$0 + [n, n+1]$	$0 + [1, 0]$	1/2/2	$n\|\mathbb{G}\| + n\|mac\| + n\|ske\|$
HK07 [10]	HDDH	1	$0 + [n, 2]$	$0 + [1, 0]$	2/2/3	$n\|\mathbb{G}\| + \|mac\|$
CKS08 [5]	HDDH	$1/n$	$0 + [2n, n+1]$	$0 + [1, 0]$	1/4/4	$2n\|\mathbb{G}\| + n\|ske\|$
HK08 [9]	HDDH	$1/n$	$0 + [n, n+1]$	$0 + [1, 0]$	1/3/3	$n\|\mathbb{G}\| + n\|mac\| + n\|ske\|$
Ours (CKS08)	HDDH	1	$0 + [2n, 2]$	$0 + [2, 2]$	2/3/5	$2n\|\mathbb{G}\|$
Ours (HK08)	HDDH	1	$0 + [n, 2]$	$0 + [1, 0]$	2/2/4	$n\|\mathbb{G}\| + \|mac\|$

given HDDH instance $(g, g^{\bar{a}}, g^{\bar{b}}, Z)$, $\mathcal{B}$ sets $u^* := g^{\bar{a}}$ and $y_0 := g^{\bar{b}}$. The algorithm $\mathcal{B}$ slightly changes the key generation algorithm to always set $s_i \leftarrow \alpha^* (:= \mathsf{TCR}(u^*))$ for all i. Then, for each i, $\mathcal{B}$ computes $v_i \leftarrow (u^*)^{F_{s_i}} (:= (u^*)^{f_i(\alpha^*)})$ and sets $(u^*, v_1^*, v_2^*, \ldots, v_n^*)$ as a target ciphertext. Next, $\mathcal{B}$ provides $\mathcal{A}$ with $(Z, (u^*, v_1^*, v_2^*, \ldots, v_n^*))$. If $\mathcal{A}$ makes decryption query, $\mathcal{B}$ simulates the decryption oracle in Game 2 by using secret keys that $\mathcal{B}$ generates. Finally, $\mathcal{A}$ outputs a bit d as his guess, and $\mathcal{B}$ outputs the same bit d. □

6 Concluding Remarks

In this paper, we have proposed a new approach to design efficient MKEMs. We have focused on the simulator which uses an algebraic trick from selective-ID security in the security proof of a KEM. By observing the behavior of the simulator, we have constructed two efficient MKEMs. Our proposed scheme based on [5] is the most efficient scheme among the MKEMs which are secure in the sense of IND-CCA under the HDDH assumption without MAC. The other proposed scheme based on [9] is one of the most efficient schemes among the MKEMs whose reduction efficiency of the security proof is tight, and which is secure in the sense of IND-CCCA under the HDDH assumption. We show the comparison of previously proposed MKEMs and ours in Table 2. Our constructions also show that there exist single-recipient PKE schemes which can be transformed into efficient multi-recipient schemes, even they do not pass the reproducibility test.

References

1. Barbosa, M., Farshim, P.: Randomness reuse: Extensions and improvements. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 257–276. Springer, Heidelberg (2007)
2. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemeas. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 85–99. Springer, Heidelberg (2003)
3. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
4. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Atluri, V., Meadows, C., Juels, A. (eds.) ACM Conference on Computer and Communications Security, pp. 320–329. ACM, New York (2005)
5. Cash, D., Kiltz, E., Shoup, V.: The twin Diffie-Hellman problem and applications. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 127–145. Springer, Heidelberg (2008)
6. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
7. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing 33(1), 167–226 (2003)

8. Delerablée, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007)
9. Hanaoka, G., Kurosawa, K.: Efficient chosen ciphertext secure public key encryption under the computational Diffie-Hellman assumption. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 308–325. Springer, Heidelberg (2008)
10. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. Cryptology ePrint Archive, Report 2007/288 (2007), http://eprint.iacr.org/
11. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 553–571. Springer, Heidelberg (2007)
12. Kiltz, E.: Chosen-ciphertext secure key-encapsulation based on gap hashed Diffie-Hellman. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 282–297. Springer, Heidelberg (2007)
13. Kurosawa, K.: Multi-recipient public-key encryption with shortened ciphertext. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 48–63. Springer, Heidelberg (2002)
14. Kurosawa, K., Desmedt, Y.: A new paradigm of hybrid encryption scheme. In: Franklin, M.K. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
15. Smart, N.P.: Efficient key encapsulation to multiple parties. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 208–219. Springer, Heidelberg (2005)
16. Yasuda, K., Kobayashi, T., Aoki, K., Fujisaki, E., Fujioka, A.: Multicast in the kem-dem framework. In: Proceedings of Symposium on Cryptography and Information Security, SCIS 2005, Japan (2005)

Fair Threshold Decryption with Semi-Trusted Third Parties

Jeongdae Hong[1], Jinil Kim[1], Jihye Kim[2], Matthew K. Franklin[3], and Kunsoo Park[1]

[1] School of Computer Science and Engineering, Seoul National University
{jdhong,jikim,kpark}@theory.snu.ac.kr
[2] ISaC and Department of Mathematical Sciences, Seoul National University
jihyek@snu.ac.kr
[3] Department of Computer Science, University of California, Davis
franklin@cs.ucdavis.edu

Abstract. A threshold decryption scheme is a multi-party public key cryptosystem that allows any sufficiently large subset of participants to decrypt a ciphertext, but disallows the decryption otherwise. Many threshold cryptographic schemes have been proposed so far, but fairness is not generally considered in this earlier work. In this paper, we present fair threshold decryption schemes, where either all of the participants can decrypt or none of them can. Our solutions employ semi-trusted third parties (STTP) and off-line semi-trusted third parties (OTTP) previously used for fair exchange. We consider a number of variants of our schemes to address realistic alternative trust scenarios. Although we describe our schemes using a simple hashed version of ElGamal encryption, our methods generalize to other threshold decryption schemes and threshold signature schemes as well.

1 Introduction

A threshold decryption scheme is a multi-party public key cryptosystem that allows any sufficiently large subset of participants to decrypt a ciphertext, but disallows the decryption otherwise. In a threshold decryption scheme, a secret key is typically split into secret key shares for the participants using a threshold secret sharing scheme. When a sufficiently large subset of participants wants to decrypt a ciphertext, each party computes a partially decrypted value using its secret key share. Any party who collects sufficiently many partially decrypted values can decrypt.

In this paper, we focus on *fairness* of threshold decryption, which is not easy to achieve without a fully trusted third party (TTP). In many scenarios, it is very desirable that all participants in a threshold decryption procedure should receive the correct decrypted plaintext simultaneously, even when those participants are mutually mistrustful. In a stand-alone setting, where valuable data is encrypted, it is a security problem if some cheating participants to the threshold decryption procedure can recover the plaintext while the other participants cannot. In a more complex setting, where the threshold decryption is part of a larger secure

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 309–326, 2009.

multi-party protocol (e.g., [31]), the security of the overall protocol may be compromised unless ciphertexts can be decrypted with fairness. Although we focus on threshold decryption in this work, essentially all of our methods hold for the case of threshold signatures as well (which is also an important primitive in many scenarios).

The previous threshold decryption schemes sometimes include a party called *Combiner*, who collects the decryption shares and computes the plaintext. With respect to fairness, *Combiner* would need to be a trusted third party (TTP) since it obtains the plaintext earlier than others. However, TTP is typically undesirable because all of the parties should totally trust it. Alternatively, each party in the threshold decryption can become a *Combiner* by himself if he knows the combining algorithm and all the decryption shares are exchanged among the parties. In that case, however, unfairness occurs when the first party who obtains all of the decryption shares quits the protocol without sending his decryption share. Even worse, a malicious party may obtain all decryption shares exclusively by repeating decryptions without sending his decryption share. Robust threshold decryption can be helpful here, but it does not by itself yield fairness.

We employ semi-trusted third parties (STTP) and off-line semi-trusted third parties (OTTP).[1] The STTP is an additional participant that follows the prescribed protocol correctly, while recording all communication in an attempt to learn something more about plaintexts ("semi-honest" or "honest-but-curious" adversary). The STTP is an on-line participant, since it connects to all the parties before the protocol starts, and communicates with them during the protocol even when all the parties are honest.

Our first solution uses a single STTP to achieve fair threshold decryption. However, in practice, the reconstructing parties may fail to agree on a mutually satisfactory STTP. For example, imagine a situation where the parties of organization **A** trust only STTP α, while the parties of organization **B** trust only STTP β. To cover this kind of situation, we give a second solution that uses multiple "weak" STTPs to achieve fair threshold decryption. A "weak" STTP works semi-honestly for all the parties that trust it, but may work maliciously for the other parties. We present fair threshold decryption schemes for two natural variants of the multiple weak STTPs setting.

Our third solution uses a single OTTP to achieve fair threshold decryption. The OTTP is a semi-honest additional participant that is not involved in the protocol unless one or more of the reconstructing parties crashes or attempts to cheat. The OTTP is an off-line participant, since it does not connect to any of the parties during normal protocol execution. This kind of protocol is often called "optimistic", since troubles are resolved afterwards rather than prevented beforehand.

For all of our solutions, no information leaks to the sender about the decryption policy (i.e., encryption looks like ordinary public key encryption). This has

[1] Some previous work in fair exchange used the term "OTTP" to indicate off-line third party which is fully-trusted rather than semi-trusted.

always been an essential design principle for threshold cryptography, and it rules out simple approaches where the sender splits his message and encrypts using multiple keys.

Related Work: Many threshold cryptographic schemes have been proposed so far [15,37,24,27,25,39,19,23], but fairness is not generally considered in this earlier work. Cleve [13] showed the impossibility of completely fair protocols without an honest majority for arbitrary functions, but Gordon et al. [30] reopened the question for specific functions of interest. Particularly, if $t < n/3$ where n is the total number of parties and t is the number of corrupted parties, fairness for secure multi-party computation (MPC) protocols can be achieved without information-theoretic or computational assumptions [5,12,29,28]. If $n/3 \leq t < n/2$, a broadcast channel is necessary to achieve fairness [36,29,28]. Our consideration is on the general case where t can be any value between 1 and n. The "gradual release" paradigm can give a relaxation of complete fairness that is useful in many contexts (see, e.g., [6,34,4,8,9,35,22,21]). On-line semi-trusted parties have been used for fair exchange and related functions (e.g., [20,41,14]). Off-line semi-trusted parties have been used for "optimistic" fair exchange (e.g., [1,2,3,40,7,17,16]) and two-party optimistic fair secure computation [10]. Lindell [33] considers a related setting in which the off-line semi-trusted party is replaced with a legal infrastructure that respects digital signatures. Lepinski et al. [32] used physical assumptions to realize fair secure function evaluation (SFE), which includes fair MPC as a special case.

The rest of the paper is organized as follows. Section 2 gives some preliminary cryptographic background. We present security models and definitions in Section 3. We present fair threshold decryption with (on-line) STTP (both strong and multiple weak) in Section 4 and optimistic fair threshold decryption with OTTP in Section 5.

2 Cryptographic Background

In this section, we briefly review the scenario of threshold decryption and a threshold version of Hash-ElGamal cryptosystem.

2.1 Threshold Decryption

The scenario of $(t+1, \ell)$ - threshold decryption is as follows. There are a dealer, ℓ shareholders who can participate in the decryption, and a combiner who actually decrypts the ciphertext. The dealer initializes a public key/secret key pair of the underlying non-threshold cryptosystem and splits the secret key into ℓ *secret key shares*. Each shareholder P_i keeps the corresponding secret key share s_i privately. Anyone can encrypt a message by the public key. When a group of $t+1$ shareholders wants to decrypt a ciphertext, each shareholder computes a partially decrypted value called *decryption share* by its secret key share. Then, the combiner collects $t+1$ decryption shares and obtains the plaintext by the combining algorithm. Throughout this paper, we use the following notations.

- *secret key share* - denotes a piece of the secret key shared by the secret sharing scheme.
- *decryption share* - denotes a partially decrypted value of a ciphertext by a secret key share.
- *reconstruction group* - denotes a group of $t+1$ parties participating in the decryption process.

2.2 Threshold Version of Hash-ElGamal Cryptosystem

Throughout the paper, we use the threshold version of Hash-ElGamal cryptosystem [18], which allows us to focus on the main ideas of our schemes and the security arguments introduced by the fairness.

Set-Up: Let G be a multiplicative group of large prime order q. Let H be a hash function from G to plaintext-length bitstrings. Let g be a generator of G.

Key Generation: A dealer does the following:

1. Chooses a secret key $SK = x \in_R [1..q-1]$ and computes public key $PK = g^x$.
2. Picks a random polynomial $f(\cdot)$ with degree t for Shamir's secret sharing scheme [38] whose coefficients are picked in $[0..q-1]$ and $f(0) = x$.
3. For all $1 \leq i \leq \ell$, computes *secret key share* $s_i = f(i) \bmod q$, verification key $VK_i = g^{s_i}$, sends s_i to party P_i, and publishes $g, PK, \{(i, VK_i)\}_{1 \leq i \leq \ell}$.

Encryption: To encrypt a message m, one randomly chooses $r \in_R [1..q-1]$ and computes $E(m) = (g^r, m \oplus H(g^{rx}))$.

Threshold Decryption: Let $E(m) = (u, v)$, and let $S \subseteq [1..\ell]$ be a reconstruction group with $|S| = t+1$. For each $i \in S$, P_i sends its *decryption share* $w_i = g^{rs_i}$ to the combiner. The combiner computes $g^{rx} = \prod_{i \in S}(w_i)^{\lambda_i}$, where $\{\lambda_i\}_{i \in S}$ are the appropriate Lagrange coefficients (i.e., $\lambda_i = \prod_{b \in S \setminus \{i\}} \frac{i}{b-i}$). Then the combiner computes $v \oplus H(g^{rx}) = m$.

Robust Threshold Decryption: P_i also sends to the combiner a zk-proof of equality of discrete logarithms that $\text{disclog}_g(VK_i) = \text{disclog}_{g^r}(w_i)$. The combiner rejects the decryption share from P_i unless this zk-proof is good.

This threshold decryption scheme is semantically secure against a chosen plaintext attack if H is modeled as a random oracle, and if the Computational Diffie-Hellman problem (CDH) is hard in G (where the chosen plaintext attack is performed by an adversary that can see up to t shares of the decryption key). The robust threshold decryption scheme has similar security if (in addition) the proof of equality of discrete logarithms is sound, complete and zero knowledge.

3 Security Models and Definitions

In this section, we introduce members of our scenarios and define *fairness* of threshold decryption.

3.1 Members and Security Models

The members of our fair threshold decryption scheme consist of a dealer D, ℓ shareholders and additional semi-trusted third parties (STTPs).

- **Dealer.** A dealer D initializes the scheme as in the usual threshold decryption. If desired, a distributed key generation protocol can replace this trusted dealer using standard methods (e.g., [26]).
- **Shareholder.** A shareholder is a legal member with a secret key share who can participate in the decryption. We assume that each shareholder works in a malicious model. That is, it may arbitrarily deviate from a specified protocol. It may refuse to participate in the protocol or abort the protocol prematurely.
- **Strong STTP.** A strong STTP is an STTP trusted to work semi-honestly by all the shareholders. Only one strong STTP suffices in our fair threshold decryption scheme.
- **Weak STTP.** A weak STTP is an extended notion of STTP which is trusted to work semi-honestly by other weak STTPs and some of the shareholders but not trusted by others. To all the shareholders who trust it, it faithfully works in semi-honest model, while it can work maliciously to other shareholders. In our schemes with multiple weak STTPs, several weak STTPs work together like one strong STTP.
- **Off-line STTP (OTTP).** An off-line STTP is an STTP trusted by all the shareholders but it does not attend the protocol if all the shareholders behave honestly.

Every on-line (strong or weak) STTP has its secret key share distributed privately during the key generation and can compute its decryption share like a shareholder.

Communication Model: We assume that every pair of participants have a private and reliable communication channel connecting them. This includes all combinations of shareholder-to-shareholder, shareholder-to-STTP, and STTP-to-STTP communication.

3.2 Formal Definition of Fairness

We define fairness of $(t+1, \ell)$-threshold decryption as follows.

Definition 1. *(Fairness of threshold decryption) If any shareholder P_i decrypts a ciphertext, then there exists at least one reconstruction group S with $P_i \in S$ and $|S| = t+1$ such that all the shareholders of S can get the plaintext.*

In the above definition, fairness means that all the shareholders of S can obtain the plaintext or no shareholder can. Any shareholder out of S should not be able to decrypt the ciphertext unless some shareholders of S send the plaintext to it. This requirement is important for applications in which the plaintext should be kept secret among the participants (e.g., [31]).

Our definition of fairness implies that any fair threshold decryption scheme should be robust against such an attack whereby malicious shareholders initiate two or more reconstruction protocols for the same ciphertext (e.g., with disjoint subsets of honest shareholders), aborting so that no honest shareholder succeeds in any single reconstruction effort, while the malicious shareholders can decrypt by combining honest decryption shares from all of the reconstruction efforts.

When the number of corrupted shareholders is less than $(t+1)/2$ (i.e. majority of any reconstruction group is honest), fairness can be achieved by general results of fair MPC [5,12,29,28]. We employ STTPs to cover the general case where up to t shareholders are corrupted. The following definition states fairness of threshold decryption when STTPs are involved in.

Definition 2. *(Fairness of threshold decryption with STTPs) A threshold decryption with STTPs is fair if all the shareholders achieve fairness in Definition 1 with the help of STTPs, while no STTP can learn anything about the decrypted message in polynomial time.*

4 Fair Threshold Decryption with (On-Line) STTP

4.1 Security Notions

We formally define two security notions, *STTP-assistance* and *STTP-obliviousness* by challenge-adversary games. Informally, *STTP-assistance* means that if any one of STTPs does not contribute its decryption share, no coalition of shareholders can decrypt any ciphertext even with the secret key shares of the other STTPs. And *STTP-obliviousness* means that no STTP, even with t secret key shares of shareholders, can learn anything about the decrypted message during polynomial number of decryptions on any ciphertexts.

STTP-assistance. We say that a threshold scheme satisfies STTP-assistance if any polynomially bounded adversary $\mathcal{A}$ cannot win the following game with non-negligible probability. The game proceeds between $\mathcal{A}$ and a challenger $\mathcal{CH}$ where there are k STTPs:

1. $\mathcal{CH}$ runs Set-Up and Key Generation algorithms taking a security parameter. $\mathcal{CH}$ gives $\mathcal{A}$ the resulting common parameters.
2. $\mathcal{A}$ receives all the secret key shares of ℓ shareholders and $k-1$ secret key shares of the STTPs from $\mathcal{CH}$.
3. $\mathcal{A}$ adaptively makes a polynomial number of queries to $\mathcal{CH}$ on any messages. For each message M, $\mathcal{CH}$ generates encryption C of M and responds with C and the corresponding decryption share of the remaining STTP with zk-proof.
4. $\mathcal{A}$ selects two target messages (M_0, M_1). $\mathcal{CH}$ picks one message M_b by selecting a random bit $b \leftarrow \{0,1\}$ and sends a ciphertext C_b of M_b to $\mathcal{A}$.
5. Repeat step 3.
6. $\mathcal{A}$ outputs b' (and wins if $b' = b$).

STTP-obliviousness. We say that a threshold scheme with STTP satisfies STTP-obliviousness if it satisfies any polynomially bounded adversary $\mathcal{A}$ cannot win the following game with non-negligible probability. The game proceeds between $\mathcal{A}$ and a challenger $\mathcal{CH}$ where there are k STTPs:

1. $\mathcal{CH}$ runs Set-Up and Key Generation algorithms taking a security parameter. $\mathcal{CH}$ gives $\mathcal{A}$ the resulting common parameters.
2. $\mathcal{A}$ is given all the secret key shares of k STTPs and t secret key shares of shareholders.[2]
3. $\mathcal{A}$ adaptively makes a polynomial number of queries to $\mathcal{CH}$ on (M, S) where $S \subseteq [1..\ell]$, $|S| = t+1$. For each (M, S), $\mathcal{CH}$ generates encryption C of M, and responds with C and the corresponding decryption shares with zk-proofs following the protocol on behalf of the shareholders of S.
4. $\mathcal{A}$ selects two target messages (M_0, M_1). $\mathcal{CH}$ picks one message M_b by selecting a random bit $b \leftarrow \{0, 1\}$ and sends a ciphertext C_b of M_b to $\mathcal{A}$. $\mathcal{CH}$ simulates executions of all the shareholders so that $\mathcal{A}$ can participate in the decryption of C_b on behalf of the STTPs.
5. Repeat step 3.
6. $\mathcal{A}$ outputs b' (and wins if $b' = b$).

4.2 Strong STTP Fair Threshold Hash ElGamal

Description: The main idea of this scheme is that the secret key is split into $\ell+1$ secret key shares instead of ℓ secret key shares so that a secret key share is assigned for the strong STTP. However, we define the secret key share of the strong STTP as a special one, thus even more than $t+1$ shareholders cannot decrypt a ciphertext without the decryption share of the strong STTP. The strong STTP can *trigger* the decryption by sending its decryption share after the shareholders of S exchange their decryption shares with one another. This scenario is applicable to all schemes which follow the general threshold decryption scenario. Fig. 1 shows this scenario graphically. The details of the protocol are as follows.

Key Generation: D chooses $x, R \in_R [1..q-1]$, and computes $PK = g^x$, $VK_{STTP} = g^R$. D picks an otherwise random degree t polynomial $f(\cdot)$ with coefficients in $[0..q-1]$ such that $f(0) = (x - R) \bmod q$. Then for all i $(1 \le i \le \ell)$, D computes $s_i = f(i) \bmod q$, $VK_i = g^{s_i}$, and sends s_i to shareholder P_i. D sends $s_{STTP} = R$ to the STTP. Lastly, D publishes $g, PK, VK_{STTP}, \{(i, VK_i)\}_{1 \le i \le \ell}$.

Strong STTP Fair Threshold Decryption: Let $E(m) = (u, v)$, and let $S \subseteq [1..\ell]$, $|S| = t+1$. The STTP do not have to know (u, v) or S.

For each $i, j \in S$, P_i sends $w_i = u^{s_i}$ and a zk-proof that $\mathrm{disclog}_g(VK_i) = \mathrm{disclog}_u(w_i)$ to P_j. If P_i succeeds in verifying $t+1$ decryption shares (including her own), then P_i sends a $(READY, S, (u, v))$ signal to the STTP. When the STTP receives consistent and well-formed $READY$ signals from at least $t+1$ shareholders, the STTP sends to each $READY$ signaller $w_{STTP} = u^{s_{STTP}}$ and

[2] This can be regarded as a passive collusion among the STTPs and t shareholders where the colluding shareholders provide the STTPs with their secret key shares.

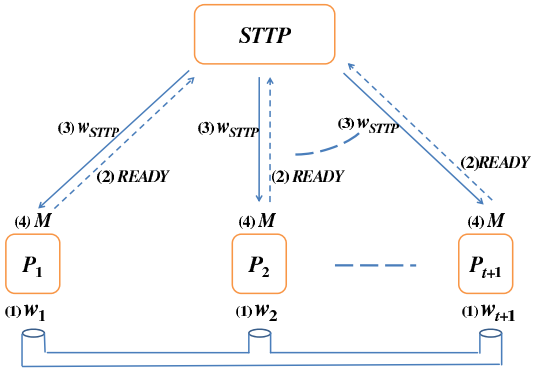


Fig. 1. Fair Threshold Decryption with one strong STTP

a zk-proof that $\text{disclog}_g(VK_{STTP}) = \text{disclog}_u(w_{STTP})$. If P_i was a *READY* signaller, and if the STTP sent a valid decryption share, then P_i can now decrypt, since $g^{rx} = w_{STTP} \cdot \prod_{i \in S} w_i^{\lambda_i}$ where $\lambda_i = \prod_{b \in S \setminus \{i\}} \frac{i}{b-i}$.

Non-Threshold Access Structure: Instead of viewing this as a $(t+1)$-out-of-ℓ threshold decryption scheme with ℓ shareholders $(P_1, \ldots, P_\ell)$ and a separate STTP, we can view it as a "non-threshold" access structure with $\ell + 1$ shareholders $(STTP, P_1, \ldots, P_\ell)$. Specifically, the access structure Γ' for successful decryption is $STTP \wedge \Gamma_{t+1,\ell}$, where $\Gamma_{t+1,\ell}$ is the $(t+1)$-out-of-ℓ threshold access structure on $\{P_1, \ldots, P_\ell\}$. Strong STTP Fair Threshold Decryption is possible whenever this non-threshold access structure Γ' can be realized.

Communication Cost: Each shareholder's communication cost is $O(t)$ and the total communication cost is $O(t^2)$ except the key generation step. These costs are necessary for each shareholder to collect decryption shares from $t+1$ shareholders.

Security Proof of Strong-STTP Version: Now we prove that the strong-STTP protocol satisfies *STTP-assistance*, *STTP-obliviousness*, and *fairness* from the CDH assumption.

Theorem 1. *(STTP-assistance) Under the CDH assumption, the strong-STTP protocol satisfies STTP-assistance in the random oracle model.*

Proof. Let us assume the existence of an adversary $\mathcal{A}$ able to break STTP-assistance. We now describe that a challenger $\mathcal{CH}$ can solve the CDH problem using the adversary $\mathcal{A}$ with non-negligible probability. When starting the STTP-assistance game, $\mathcal{CH}$ gets an instance of CDH problem (g, g^x, g^y) whose goal is computing g^{xy}.

In step 1, $\mathcal{CH}$ chooses a random polynomial $f(\cdot)$ with degree t. He simulates the strong-STTP protocol with initializing $PK = g^x$, $s_1 = f(1), ..., s_\ell = f(\ell)$. Then, s_{STTP} is automatically chosen as $s_{STTP} = x - f(0)$ but $\mathcal{CH}$ knows neither x nor s_{STTP}. The verification keys are easily computed by $VK_1 = g^{s_1}, ..., VK_\ell =$

$g^{s_\ell}, VK_{STTP} = g^x/g^{f(0)}$. $\mathcal{CH}$ sends common parameters $(PK, VK_1, ..., VK_\ell, VK_{STTP})$ to $\mathcal{A}$.

In step 2, $\mathcal{CH}$ sends $(s_1, ..., s_\ell)$ to $\mathcal{A}$. $\mathcal{A}$ cannot distinguish these from a set of normal parameters because all of them are valid.

In step 3, for each query M of $\mathcal{A}$, $\mathcal{CH}$ generates a ciphertext $Enc(M) = (g^r, M \oplus H(g^{rx}))$ where $r \in_R [0..q-1]$ and responds with $Enc(M)$, $w_{STTP} = (VK_{STTP})^r$ with the corresponding simulated zk-proof.

In step 4, for given two messages M_0, M_1 from $\mathcal{A}$, $\mathcal{CH}$ picks one message M_b by selecting a random bit $b \leftarrow \{0, 1\}$ and generates a dummy ciphertext: $\hat{Enc}(M_b) = (g^y, M_b \oplus h)$ where $h \in_R [0..q-1]$. $\mathcal{CH}$ sends $\hat{Enc}(M_b)$ to $\mathcal{A}$.

In step 5, for every query M, $\mathcal{CH}$ works the same way as in step 3. Finally, $\mathcal{A}$ outputs b' such that $Pr[b' = b]$ is non-negligibly higher than $1/2$.

Claim 1: Unless $\mathcal{A}$ queries g^{xy} to the random oracle $H(\cdot)$, $\mathcal{A}$ cannot obtain any non-negligible advantage in guessing M_b.

If $\mathcal{A}$ does not query g^{xy} to the random oracle, $M_b \oplus h$ in $\hat{Enc}(M_b)$ is the same as the one-time pad whose key is h. Since the one-time pad has perfect secrecy, $\mathcal{A}$ cannot obtain anything from $M_b \oplus h$. It implies no such adversary can obtain non-negligible advantage from any dummy ciphertext.

Claim 2: If $\mathcal{A}$ queries g^{xy} to the random oracle, $\mathcal{CH}$ can solve the CDH problem with non-negligible probability.

The number of queries, q_h, recorded in the random oracle is polynomially bounded due to $\mathcal{A}$'s running time. If $\mathcal{A}$ has a non-negligible advantage Adv, $\mathcal{A}$ can solve the CDH problem with probability Adv/q_h which is also non-negligible. It contradicts the CDH assumption.

By both claims, no adversary can obtain non-negligible advantage from the above game if the CDH problem is hard. That implies STTP-assistance holds for the strong-STTP protocol under the CDH assumption in the random oracle model.

Theorem 2. *(STTP-obliviousness) Under the CDH assumption, the strong-STTP protocol satisfies STTP-obliviousness in the random oracle model.*

Proof. The proof is similar to that of Theorem 1 except simulation setup. In step 1, $\mathcal{CH}$ simulates verification keys $VK_1 = g^{s_1}, ..., VK_t = g^{s_t}, VK_{STTP} = g^R$. For each $i \in [t+1..\ell]$, $\mathcal{CH}$ sets $VK_i = (\frac{PK}{VK_1^{\lambda_1} ... VK_t^{\lambda_t} \cdot VK_{STTP}})^{1/\lambda_i}$. $\mathcal{CH}$ sends common parameters $(PK, VK_1, ..., VK_\ell, VK_{STTP})$ to $\mathcal{A}$. In step 2, $\mathcal{CH}$ sends t secret key shares $(s_1, ..., s_t)$ and STTP's secret key share R to $\mathcal{A}$.

Then, since Claim 1, 2 of Theorem 1 hold in the same way, no adversary can obtain non-negligible advantage from the above game if the CDH problem is hard. That leads to STTP obliviousness under the CDH assumption in the random oracle model.

Theorem 3. *(Fairness) The strong-STTP protocol satisfies fairness of threshold decryption with STTP in Definition 2.*

Proof. By Theorem 1, no t shareholders can succeed in decryption without following the protocol until the STTP's sending its decryption share. The STTP

contributes its decryption share only after all the shareholders of S exchange their decryption shares with one another. Since the STTP sends its share to the shareholders at the same time, it guarantees that they can get the decrypted message simultaneously. From this and Theorem 2, the strong-STTP protocol satisfies the fairness with STTP in Definition 2.

4.3 Multiple-Weak STTP Fair Threshold Hash ElGamal

Description. When all the shareholders cannot agree on a mutually satisfactory STTP, they can make use of several weak STTPs instead of one strong STTP. The members of this scenario are a dealer D, ℓ shareholders and k weak STTPs. We assume that each shareholder trusts at least one weak STTP. Moreover, we assume that any subset of less than $t+1$ shareholders has at least one common trustworthy weak STTP, which we call the *technical covering condition*.

The reliability between weak STTPs and all the shareholders is public. Let S be a reconstruction group of $t+1$ shareholders that collaboratively want to decrypt the ciphertext. Before introducing our scheme, we formally define the technical covering condition as follows.

Definition 3. *(Technical Covering Condition) Given a reconstruction group S with $|S| = t+1$, and k weak STTPs, let $T_i \subseteq [1..k]$ be the subset of indices of the weak STTPs that P_i trusts (i.e., P_i's trustworthy weak STTPs), and let $U_i = [1..k] - T_i$ be that of the weak STTPs that P_i do not trust (i.e., P_i's untrustworthy weak STTPs). For any subset $F \subset [1..\ell]$ with $|F| \leq t$, if $\cap_{i \in F} T_i$ is non-empty, then we say the technical covering condition is satisfied.*

Once we assume that the technical covering condition is satisfied, we can use the following scheme for fair threshold decryption. In this scenario, each shareholder P_i of S collects the decryption shares of U_i (his untrustworthy weak STTPs) so that it can proceed the threshold decryption without doubting them. Then, all the shareholders of S exchange their decryption shares with one another and send *READY* signals to all the weak STTPs. When each weak STTP receives *READY* signals from all the shareholders of S, it sends its own *READY* signals to all other weak STTPs. When each weak STTP receives *READY* signals from all the remaining weak STTPs, it *triggers* the decryption by sending its decryption share to the shareholders who trust it. Fig. 2 shows this scenario graphically. The details of the protocol are as follows.

Key Generation: D chooses random $x, R_1, \ldots, R_k \in [1..q-1]$, and computes $PK = g^x$, $VK_{STTP_1} = g^{R_1}, \ldots, VK_{STTP_k} = g^{R_k}$. D picks an otherwise random degree t polynomial $f(\cdot)$ with coefficients in $[0..q-1]$ such that $f(0) = (x - \sum_{i=1}^{k} R_i) \bmod q$. Then for all i, $1 \leq i \leq \ell$, D computes $s_i = f(i) \bmod q$, $VK_i = g^{s_i}$, and sends s_i to P_i. Then for all j, $1 \leq j \leq k$, D sends $s_{STTP_j} = R_j$ to $STTP_j$. Lastly, D publishes $g, PK, \{(j, VK_{STTP_j})\}_{1 \leq j \leq k}, \{(i, VK_i)\}_{1 \leq i \leq \ell}$.

Multiple-Weak STTP Fair Threshold Decryption: Let $E(m) = (u, v)$, and let $S \subseteq [1..\ell]$, $|S| = t+1$. Let T_i and U_i be the same as those of Definition 3.1. We assume that the technical covering condition is satisfied. We assume that the STTPs know (u, v), S and $\{T_i\}_{i \in S}$.

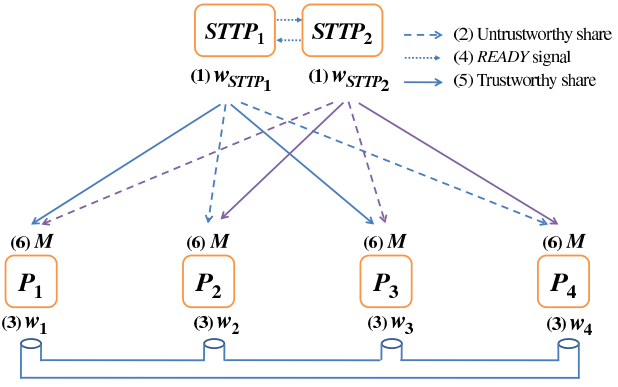


Fig. 2. An example of fair threshold decryption with weak STTPs where $t+1=4, k=2$. Odd-numbered shareholders trust $STTP_1$ but do not trust $STTP_2$, while even-numbered shareholders trust $STTP_2$ but do not trust $STTP_1$. Dotted line indicates that each STTP sends its decryption share to the shareholders who do not trust it in step 1, and solid line indicates that each STTP sends it to the shareholders who trust it in step 4.

1. For every $j \in [1..k]$, and for every $i \in [1..\ell]$ such that $j \in U_i$, $STTP_j$ sends $w_{STTP_j} = u^{R_j}$ and a zk-proof that $\text{disclog}_g(VK_{STTP_j}) = \text{disclog}_u(w_{STTP_j})$ to P_i.
2. Each P_i checks the decryption shares it received from U_i (its untrustworthy weak STTPs) and it halts if any is bad. Otherwise, P_i sends $w_i = u^{s_i}$ and a zk-proof that $\text{disclog}_g(VK_i) = \text{disclog}_u(w_i)$ to every shareholder in S. When P_i receives good decryption shares from all the other shareholders in S, then P_i sends a *READY* signal to all the STTPs.
3. Each STTP waits for *READY* signals from all the shareholders in S, and then sends its own *READY* signal to all the other STTPs. Each STTP goes on to the next step after receiving *READY* signals from all the other STTPs.
4. For every $j \in [1..k]$, and for every $i \in [1..\ell]$ such that $j \in T_i$, $STTP_j$ sends w_{STTP_j} and a zk-proof that $\text{disclog}_g(VK_{STTP_j}) = \text{disclog}_u(w_{STTP_j})$ to P_i.
5. Each P_i can now decrypt, since $g^{rx} = \prod_{1 \le j \le k} w_{STTP_j} \cdot \prod_{i \in S} w_i^{\lambda_i}$ where $\lambda_i = \prod_{b \in S \setminus \{i\}} \frac{i}{b-i}$.

Non-Threshold Access Structure: Instead of viewing this as a $(t+1)$-out-of-ℓ threshold decryption scheme with ℓ shareholders $(P_1, \ldots, P_\ell)$ and k separate STTPs $(STTP_1, \ldots, STTP_k)$, we can view it as a non-threshold access structure with $\ell + k$ shareholders $(STTP_1, \ldots, STTP_k, P_1, \ldots, P_\ell)$. Specifically, the access structure Γ' for successful decryption is $(STTP_1 \wedge \ldots \wedge STTP_k) \wedge \Gamma_{t+1,\ell}$, where $\Gamma_{t+1,\ell}$ is the $(t+1)$-out-of-ℓ threshold access structure on $\{P_1, \ldots, P_\ell\}$. Multiple-Weak STTP Fair Threshold Decryption is possible whenever this non-threshold access structure Γ' can be realized.

Communication Cost: Each shareholder's communication cost is $O(t+k)$ and the total communication cost is $O((t+k)^2)$.

Security of Multiple Weak-STTP Version: The formal proof of the security is a simple extension of that for the strong STTP protocol, which will be presented in the full version.

5 Optimistic Fair Threshold Decryption

5.1 Security Notions

Our notion of optimistic fair threshold decryption uses an OTTP which is semi-honest but not attending the protocol if all the shareholders behave honestly. We do not require optimistic protocols to satisfy *STTP-assistance* because any $t+1$ honest shareholders can decrypt the ciphertext without OTTP. We use the following security notions for optimistic fair threshold decryption.

Security for threshold decryption. Informally, a $(t+1)$-threshold decryption scheme is secure if t number of shareholders cannot decrypt any ciphertext. Formally, we say that a threshold scheme is semantically secure if any polynomially bounded adversary $\mathcal{A}$ cannot win the following game with non-negligible probability. The game proceeds between $\mathcal{A}$ and $\mathcal{CH}$:

1. $\mathcal{CH}$ runs Set-Up and Key Generation algorithms taking a security parameter. $\mathcal{CH}$ gives $\mathcal{A}$ the resulting common parameters and t secret key shares.
2. $\mathcal{A}$ adaptively makes a polynomial number of queries to $\mathcal{CH}$ on (M, S) where $S \subseteq [1..\ell]$, $|S| = t+1$. For each (M, S), $\mathcal{CH}$ generates encryption C of M and responds with C and the messages each shareholder sends in the execution of optimistic decryption on (C, S).
3. $\mathcal{A}$ adaptively queries for OTTP's responses to $\mathcal{CH}$.
4. $\mathcal{A}$ selects two target messages (M_0, M_1). $\mathcal{CH}$ picks one message M_b by selecting a random bit $b \leftarrow \{0,1\}$ and sends a ciphertext C_b of M_b to $\mathcal{A}$. $\mathcal{CH}$ also sends to $\mathcal{A}$ all the intermediate information except decryption shares.
5. Repeat step 2.
6. $\mathcal{A}$ outputs b' (and wins if $b' = b$).

OTTP obliviousness. We say that a threshold scheme is OTTP-oblivious if any polynomially bounded adversary $\mathcal{A}$ cannot win the following game with non-negligible probability. The game proceeds between $\mathcal{A}$ and $\mathcal{CH}$:

1. $\mathcal{CH}$ runs Set-Up and Key Generation algorithms taking a security parameter. $\mathcal{CH}$ gives $\mathcal{A}$ the resulting common parameters and OTTP's secret key.
2. $\mathcal{A}$ adaptively makes a polynomial number of queries to $\mathcal{CH}$ on (M, S) where $S \subseteq [1..\ell]$, $|S| = t+1$. For each (M, S), $\mathcal{CH}$ generates encryption C of M and responds with C and the messages each shareholder sends in the execution of optimistic decryption on (C, S).
3. $\mathcal{A}$ selects two target messages (M_0, M_1). $\mathcal{CH}$ picks one message M_b by selecting a random bit $b \leftarrow \{0,1\}$ and sends a ciphertext C_b of M_b to $\mathcal{A}$. $\mathcal{A}$ participates in a decryption protocol on (C_b, S) on behalf of the OTTP.
4. Repeat step 2.
5. $\mathcal{A}$ outputs b' (and wins if $b' = b$).

5.2 Optimistic Fair Threshold Hash ElGamal

Description: The main idea of this protocol is that all the shareholders of S exchange the promises of decryption shares before they exchange the decryption shares. A promise of decryption share is an encrypted value containing partial information of the decryption share using OTTP's public key. It assures the receiver that he can obtain the decryption shares with the help of OTTP. Thus, once each shareholder receives all the promises, it can obtain all the decryption shares even when some shareholders behave maliciously.

In this protocol, the OTTP does not respond to any query before all the promises are exchanged. To guarantee this, each shareholder sends its signed *READY* signal to all the other shareholders of S, so that only shareholders who receive all the signed *READY* signals can query to the OTTP in order to get the decryption shares that they have not received. The OTTP accepts only queries enclosed with all the signed *READY* signals of S. (We can regard it as OTTP's decryption policy.) Once the OTTP receives a query with the signed *READY* signals, it can be assured that the shareholders of S already received all the promises of S. At this moment, the OTTP sends all the signed *READY* signals to the other shareholders of S. It guarantees that the shareholders have the right to query to the OTTP.

One main tool is *verifiable encryption*, which allows someone to prove that an encrypted value is the discrete logarithm of an unencrypted value (with respect to an unencrypted base). Let ENC be a public key encryption scheme that supports verifiable encryption as in Camenisch-Shoup [11]. The details of the protocol are as follows.

Key Generation: A dealer initializes common parameters of ordinary threshold Hash-ElGamal as in Section 2.2. OTTP creates a key pair (SK_{OTTP}, PK_{OTTP}) for $ENC(\cdot)$.

Optimistic Fair Threshold Decryption: Let $E(m) = (u, v)$, $S \subseteq [1..\ell]$, $|S| = t + 1$. We assume that all of the shareholders agree on the session information *inf* which includes S, $E(m)$.

1. In Round 1, for every $i, j \in S$:
   (a) P_i sends the following promise to P_j (unfair, blinded, partially signed): $(\alpha_i, \beta_i, inf, g^{r/\alpha_i}, ENC_{OTTP}(\beta_i s_i), \sigma_i, proof(\mathrm{disclog}_g(VK_i^{\beta_i}) = \beta_i s_i))$, where $\alpha_i, \beta_i \in_R [1..q-1]$ chosen by P_i and σ_i is the signature by P_i of $(inf, g^{r/\alpha_i}, ENC_{OTTP}(\beta_i s_i))$. The disclog proof is Camenisch-Shoup style on the verifiably encrypted value $\beta_i s_i$.
   (b) P_j checks if the promise from P_i is valid: well-formed, well-signed, well-blinded, disclog proof.
   (c) If P_i receives $t + 1$ valid promises of S (including its own), P_i proceeds to Round 2.
2. In Round 2, for every $i \in S$:
   (a) P_i sends $(READY, inf)$ and its signature to the other shareholders in S.

(b) If P_i receives $t+1$ signed $(READY, inf)$ signals of S (including its own) from S or (possibly) OTTP, P_i proceeds to Round 3.

3. In Round 3, for every $i \in S$:
   (a) P_i sends to the other shareholders in S: $(g^{rs_i}, proof(\text{disclog}_{g^r}(g^{rs_i}) = \text{disclog}_g(VK_i)))$. Here the proof is an ordinary zk-proof for equality of discrete logs.
   (b) If P_i receives $t+1$ good decryption shares (including its own), then P_i decrypts as in ordinary (robust) threshold decryption, and halts. Otherwise, P_i proceeds to Round 4.
4. In Round 4, for every $i \in S$:
   (a) Let $\hat{S}$ be the subset of S which did not send P_i a good decryption share in Round 3.
   (b) P_i sends OTTP all the signed $(READY, inf)$ signals of S, and the partially signed promise $(inf, g^{r/\alpha_j}, ENC_{OTTP}(\beta_j s_j), \sigma_j)$ from every shareholder j in $\hat{S}$.
   (c) OTTP checks all the signed $(READY, inf)$ signals of S and all the signed part of promises of $\hat{S}$. If any of them is invalid, OTTP rejects the query.
   (d) If it is the first query of the session inf, OTTP sends all the signed $(READY, inf)$ signals of S to all the shareholders of S to finish their waiting in Round 2.
   (e) OTTP notifies each P_j in $\hat{S}$ that P_i asks OTTP so that P_j does not wait for P_i's decryption share in Round 3.
   (f) OTTP decrypts all the $(ENC_{OTTP}(\beta_j s_j))_{j \in \hat{S}}$.
   (g) OTTP computes and sends to P_i all the $((g^{r/\alpha_j})^{\beta_j s_j})_{j \in \hat{S}}$.
   (h) P_i now unblinds all the $((g^{r/\alpha_j})^{\beta_j s_j})_{j \in \hat{S}}$ using all the $(\alpha_j, \beta_j)_{j \in \hat{S}}$. P_i can obtain the plaintext as in ordinary threshold decryption, and halts.

Remark 1: The promises of step 1 are "blinded" so that the OTTP cannot compute any decryption share even if it responds to a number of queries. In the blinded promises, we use two random numbers α_i, β_i to hide the secret key shares and the decryption shares from the OTTP. Any shareholder, who receives a blinded promise, can verify its validity by checking $(g^{r/\alpha_i})^{\alpha_i} = g^r$ and $proof(\text{disclog}_g(VK_i^{\beta_i}) = \beta_i s_i)$.

Remark 2: This protocol does not satisfy a useful property *timely termination* [2] with which a shareholder can leave the protocol immediately in a fair manner without waiting for the responses of other shareholders. For instance, let's consider the following case in which a shareholder P_i is in Round 2, and the other shareholders in Round 3 do not send their decryption shares for a long time. In this case, P_i can neither go to Round 3 since he has not collected all the signed $READY$ signals, nor leave the protocol since the other shareholders will succeed in decryption by querying the OTTP. Nevertheless, this protocol is *fair* since P_i can succeed in decryption whenever another shareholder succeeds in decryption if he has not leaved the protocol. We can modify this protocol so

that timely termination is satisfied. For space constraints, this variant will be presented in the full version.

Communication Cost: Each shareholder's communication cost is $O(t)$ and the total communication cost is $O(t^2)$.

Security Proof of OTTP Version

Theorem 4. *(Security for threshold decryption) Under the CDH assumption, unforgeability of the signature scheme and the security of verifiable encryption, the above optimistic protocol is a semantically secure threshold decryption protocol in the random oracle model.*

Proof. The proof is very similar with the proof for Theorem 2. Assume that $\mathcal{A}$ able to break security for threshold decryption. Using this adversary $\mathcal{A}$, we can build an algorithm to solve the CDH problem with non-negligible probability: Given an instance of CDH problem (g, g^x, g^y), the algorithm computes g^{xy}. $\mathcal{CH}$ initializes $PK = g^x$, $s_1, ..., s_t \in_R [0..q-1]$, and $VK_1 = g^{s_1}, ..., VK_t = g^{s_t}$. For each $i \in [t+1..\ell]$, $\mathcal{CH}$ sets $VK_i = (\frac{PK}{VK_1^{\lambda_1} ... VK_t^{\lambda_t}})^{1/\lambda_i}$. $\mathcal{CH}$ also generates (SK_{OTTP}, PK_{OTTP}). $\mathcal{A}$ is given public parameters $(PK, VK_1, ..., VK_\ell, PK_{OTTP})$ and t secret key shares $(s_1, ..., s_t)$. In step 2, for each query (M, S) of $\mathcal{A}$, $\mathcal{CH}$ responds with $t+1$ decryption shares $\{w_j = (VK_j)^r\}_{j \in S}$. He also computes $t+1$ promises with the simulated signatures σ_i and zk-proofs $proof(disclog_g(VK_i^{\beta_i} = \beta_i s_i)$. In step 4, for given two messages M_0, M_1 from $\mathcal{A}$, $\mathcal{CH}$ picks one message M_b and sends a dummy ciphertext $\hat{Enc}(M_b)$. $\mathcal{CH}$ also simulates promises, signatures and zk-proofs. Under the unforgeability of the signature scheme, $\mathcal{A}$'s query to the OTTP on input created by $\mathcal{A}$ cannot pass the validity test of the input. Moreover, under the security of verifiable encryption, no information is revealed from the promises in Round 1. *READY* signals in Round 2 do not contain any information about the plaintext. Thus, similar to Theorem 2, $\mathcal{A}$ can win the game in polynomial time with non-negligible probability only by querying g^{xy} to the random oracle, which contradicts to the CDH assumption.

Theorem 5. *(OTTP-obliviousness) Under the CDH assumption, the optimistic fair threshold satisfies OTTP-obliviousness.*

Proof. The proof is the same as the proof for Theorem 4 except: $\mathcal{A}$ is given SK_{OTTP} instead of t secret shares. Still, it is clear that unless α_i, β_i are given, $\mathcal{A}$ cannot learn anything about s_i from the partially signed promises. Thus, the only way to win the game is to make a query on g^{xy}. Again, it contradicts to the CDH assumption.

Theorem 6. *(Fairness) Under the CDH assumption, unforgeability of the signature scheme, security of the verifiable encryption, the optimistic protocol satisfies fairness of threshold decryption with OTTP in Definition 2.*

Proof. Let F be a coalition of at most t cheating shareholders. All the information which F can receive comes from the promises in Round 1, the *READY*

signals in Round 2, and the decryption shares in Round 3. Since *READY* signals have nothing to do with any secret information, we only need to consider promises and decryption shares.

Claim 1: If F decrypts the ciphertext without querying to the OTTP, all the honest shareholders of S can decrypt it too.

If F does not receive decryption shares from the shareholders in Round 3, F fails to decrypt by Theorem 4. Otherwise, there exists at least one honest shareholder P_i in Round 3 who sends its decryption share to F. In Round 1 and 2, P_i received all the promises and the signed *READY* signals of S. It implies that P_i can obtain all the decryption shares with the help of the OTTP whenever it wants. Furthermore, from the signed *READY* signals which P_i received, it is obvious that all the honest shareholders had received all the promises in Round 1 and proceeded to (at least) Round 2. When P_i queries to the OTTP, all the honest shareholders in Round 2 can proceed to Round 3 due to the signed *READY* signals from the OTTP. Then, all the honest shareholders of S can decrypt the ciphertext with the help of the OTTP and Claim 1 holds.

Claim 2: If F decrypts the ciphertext by querying to the OTTP, all the honest shareholders of S can decrypt it too.

F can query to the OTTP only when he has all the signed *READY* signals. It implies that all the shareholders of S have sent their signed *READY* signals to F. It also implies that all the honest shareholders had received all the promises in Round 1 and proceeded to Round 2. Thus, whenever F queries to the OTTP, all the honest shareholders can proceed to Round 3 by receiving the signed *READY* signals which the OTTP sends. Then, all the honest shareholders of S can query to the OTTP and Claim 2 holds.

By both claims, F cannot cause any unfairness whatever they do. Thus, the OTTP protocol satisfies the fairness of Definition 1.

Acknowledgments

This work was supported by Korea Research Council of Fundamental Science and Technology. The ICT at Seoul National University provides research facilities for this study. Jihye Kim was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MEST) (No. R01-2008-000-11287-0, No. 20090058574).

References

1. Asokan, N., Schunter, M., Waidner, M.: Optimistic protocols for fair exchange. In: ACM Conference on Computer and Communications Security, pp. 7–17 (1997)
2. Asokan, N., Shoup, V., Waidner, M.: Optimistic fair exchange of digital signatures. IEEE J. Selected Areas in Communication 18(4), 593–610 (2000)
3. Bao, F., Deng, R.H., Mao, W.: Efficient and practical fair exchange protocols with off-line TTP. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 77–85 (May 1998)

4. Ben-Or, M., Goldreich, O., Micali, S., Rivest, R.L.: A fair protocol for signing contracts. IEEE Transactions on Information Theory 36(1), 40–46 (1990)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC 1988: Proceedings of the twentieth annual ACM symposium on Theory of computing, pp. 1–10. ACM, New York (1988)
6. Blum, M.: How to exchange (secret) keys. ACM Trans. Comput. Syst. 1(2), 175–193 (1983)
7. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
8. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
9. Boudot, F., Schoenmakers, B., Traoré, J.: A fair and efficient solution to the socialist millionaires' problem. Discrete Applied Mathematics 111(1-2), 23–36 (2001)
10. Cachin, C., Camenisch, J.L.: Optimistic fair secure computation. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 93–111. Springer, Heidelberg (2000)
11. Camenisch, J., Shoup, V.: Practical verifiable encryption and decryption of discrete logarithms. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 126–144. Springer, Heidelberg (2003)
12. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: STOC 1988: Proceedings of the twentieth annual ACM symposium on Theory of computing, pp. 11–19. ACM Press, New York (1988)
13. Cleve, R.: Limits on the security of coin flips when half the processors are faulty (extended abstract). In: STOC, pp. 364–369 (1986)
14. Cox, B., Tygar, J.D., Sirbu, M.: Netbill security and transaction protocol. In: First USENIX workshop on Electronic Commerce, pp. 77–88 (1995)
15. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
16. Dodis, Y., Lee, P.J., Yum, D.H.: Optimistic fair exchange in a multi-user setting. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 118–133. Springer, Heidelberg (2007)
17. Dodis, Y., Reyzin, L.: Breaking and repairing optimistic fair exchange from PODC 2003. In: Digital Rights Management Workshop, pp. 47–54 (2003)
18. ElGamal, T.: A public key cryptosystem and signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
19. Fouque, P., Poupard, G., Stern, J.: Sharing decryption in the context of voting of lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
20. Franklin, M.K., Reiter, M.K.: Fair exchange with a semi-trusted third party. In: Proceedings of the 4th ACM Conference on Computer and Communications Security (CCS), pp. 1–5 (April 1997)
21. Garay, J.A., MacKenzie, P., Yang, K.: Efficient and secure multi-party computation with faulty majority and complete fairness. Cryptology ePrint Archive, Report 2004/009 (2004), http://eprint.iacr.org/
22. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource fairness and composability of cryptographic protocols. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 404–428. Springer, Heidelberg (2006)
23. Gennaro, R., Halevi, S., Krawczyk, H., Rabin, T.: Threshold RSA for dynamic and ad-hoc groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 88–107. Springer, Heidelberg (2008)

24. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust and Efficient Sharing of RSA Functions. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 157–172. Springer, Heidelberg (1996)
25. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. Inf. Comput. 164(1), 54–84 (2001)
26. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptology 20(1), 51–83 (2007)
27. Gennaro, R., Rabin, T., Jarecki, S., Krawczyk, H.: Robust and efficient sharing of RSA functions. J. Cryptology 13(2), 273–300 (2000)
28. Goldreich, O.: Secure multi-party computation (working draft, version 1.2) (2000), http://www.wisdom.weizmann.ac.il/oded/pp.html
29. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC 1987: Proceedings of the nineteenth annual ACM symposium on Theory of computing, pp. 218–229. ACM Press, New York (1987)
30. Gordon, S.D., Hazay, C., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC, pp. 413–422 (2008)
31. Kissner, L., Song, D.: Privacy-preserving set operations. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 241–257. Springer, Heidelberg (2005)
32. Lepinski, M., Micali, S., Peikert, C., Shelat, A.: Completely fair sfe and coalition-safe cheap talk. In: PODC 2004: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing, pp. 1–10. ACM, New York (2004)
33. Lindell, A.Y.: Legally-enforceable fairness in secure two-party computation. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 121–137. Springer, Heidelberg (2008)
34. Luby, M., Micali, S., Rackoff, C.: How to simultaneously exchange a secret bit by flipping a symmetrically-biased coin. In: FOCS, pp. 11–21 (1983)
35. Pinkas, B.: Fair secure two-party computation. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 87–105. Springer, Heidelberg (2003)
36. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: STOC 1989: Proceedings of the twenty-first annual ACM symposium on Theory of computing, pp. 73–85. ACM Press, New York (1989)
37. Santis, A.D., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: STOC, pp. 522–533 (1994)
38. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
39. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
40. Zhou, J., Deng, R.H., Bao, F.: Some remarks on a fair exchange protocol. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 46–57. Springer, Heidelberg (2000)
41. Zhou, J., Gollmann, D.: An efficient non-repudiation protocol, pp. 126–132. IEEE Computer Society Press, Los Alamitos (1997)

Conditional Proxy Broadcast Re-Encryption*

Cheng-Kang Chu[1], Jian Weng[1,2],
Sherman S.M. Chow[3], Jianying Zhou[4], and Robert H. Deng[1]

[1] School of Information Systems
Singapore Management University, Singapore
{ckchu,jianweng,robertdeng}@smu.edu.sg
[2] Department of Computer Science
Jinan University, Guangzhou 510632, P.R. China
[3] Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu
[4] Institute for Infocomm Research, Singapore
jyzhou@i2r.a-star.edu.sg

Abstract. A proxy re-encryption (PRE) scheme supports the delegation of decryption rights via a proxy, who makes the ciphertexts decryptable by the delegatee. PRE is useful in various applications such as encrypted email forwarding. In this paper, we introduce a more generalized notion of conditional proxy broadcast re-encryption (CPBRE). A CPBRE scheme allows Alice to generate a re-encryption key for some condition specified during the encryption, such that the re-encryption power of the proxy is restricted to that condition only. This enables a more fine-grained delegation of decryption right. Moreover, Alice can delegate decryption rights to a set of users at a time. That is, Alice's ciphertexts can be re-broadcasted. This saves a lot of computation and communication cost. We propose a basic CPBRE scheme secure against chosen-plaintext attacks, and its extension which is secure against replayable chosen-ciphertext attacks (RCCA). Both schemes are unidirectional and proved secure in the standard model. Finally, we show that it is easy to get a unidirectional RCCA-secure identity-based proxy re-encryption from our RCCA-secure CPBRE construction.

Keywords: proxy re-encryption, conditional proxy re-encryption, broadcast encryption, hierarchical identity-coupling broadcast encryption.

1 Introduction

Proxy re-encryption (PRE) schemes enable (by delegating a transformation-key to) a semi-trusted proxy to transform Alice's ciphertext into one encrypting the same message which is decryptable by Bob, without allowing the proxy any ability to perform tasks outside of the delegation. PRE found applications [3,8] in

* Funded by A*STAR project SEDS-0721330047.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 327–342, 2009.

digital rights management, distributed file storage systems, and email forwarding. For example, users can assign their email server as the proxy such that it can re-encrypt the emails for different users without knowing the email content.

Although PRE is useful in many applications, we found that sometimes we need more than the basic. In corporate email forwarding, Alice may ask the proxy to re-encrypt her emails to her colleague Bob when she is on leave. However, this is not enough in the following scenarios:

1. Alice does not want Bob to read all her private emails.
2. For some business emails, Alice has to forward them to more than one colleague other than Bob, in an extreme case, the whole staff of the company.

Using a traditional PRE, a proxy is too powerful as it has the ability to re-encrypt *all* Alice's emails to Bob once the re-encryption key is given. For more than one delegatees, Alice needs to generate a re-encryption key for *each* staff member, and the proxy also needs to re-encrypt emails for *each* of them.

We believe there is a better way to handle these situations. We envision a more generalized notion of *Conditional Proxy Broadcast Re-Encryption* (CPBRE). Alice can specify a condition to generate a *conditional* re-encryption key, such that the re-encryption power of the proxy is restricted to that condition only. Moreover, Alice can delegate the decryption rights to a set of users at a time, which means Alice's ciphertexts can be *re-broadcasted*. In this paper, we formalize this notion, propose a basic CPBRE scheme secure against chosen-plaintext attacks (CPA), and an extension that is secure against replayable chosen-ciphertext attacks (RCCA) [9]. RCCA is a weaker variant of chosen-ciphertext attack (CCA) in which a harmless mauling of the challenge ciphertext is tolerated. Both schemes are unidirectional and secure in the standard model.

The new CPBRE is much more flexible. Back to our email-forwarding example, Alice can use the keywords "business", "private" and "golf" as the conditions, to allow forwarding of her encrypted emails to her colleagues, family members, and golf club members respectively. For each group, Alice only requires to produce one re-encryption key and the proxy only requires to transform a single ciphertext. This saves a lot of computation and communication cost.

Finally, being a generalization of PRE, it is easy to use our RCCA-secure CPBRE construction to build a RCCA-secure unidirectional identity-based proxy re-encryption (IB-PRE), which is the first of its kind.

1.1 Related Works

Following Blaze, Bleumer and Strauss's seminal work [3] which presented a bidirectional CPA-secure PRE scheme, many PRE schemes have been proposed. Ateniese *et al.* [1] presented a CPA-secure unidirectional PRE. Canetti and Hohenberger [8] presented a CCA-secure bidirectional PRE. Later, Libert and Vergnaud [14] presented a RCCA-secure unidirectional PRE. These PRE schemes [1,8,14] rely on the somewhat costly bilinear pairings. Without pairings, Deng *et al.* [11] proposed a CCA-secure bidirectional PRE. Subsequently, Weng *et al.* [19] and Shao and Cao [16] presented CCA-secure unidirectional PRE.

Proxy re-encryption has also been studied in identity-based encryption (IBE) settings. Based on Boneh-Boyen IBE [4], Boneh, Goh and Matsuo [6] described a hybrid proxy re-encryption system. Green and Ateniese [12] presented a CPA-secure IB-PRE and a CCA-secure IB-PRE, which are proven in the random oracle model and only support single-use (i.e., the ciphertext can only be re-encrypted once). Matsuo [15] also proposed a CPA-secure IB-PRE scheme. Later, Chu and Tzeng [10] tried to propose a CCA-secure multi-use IB-PRE scheme without random oracles. However, as Shao and Cao [16] stated, [12,10] are unable to resist a "chain collusion attack" (described later). Up to now, there is still no CCA-secure (or RCCA-secure) IB-PRE scheme in the standard model.

Tang [17] introduced the primitive of type-based proxy re-encryption, which allows the proxy to re-encrypt a specific type of delegator's ciphertexts. Independently, Weng *et al.* [18] introduced a similar primitive named "conditional proxy re-encryption", in which the proxy can re-encrypt a ciphertext under a specific condition iff he has the re-encryption key with respect to this condition. However, both of these schemes are proved in the random oracle model. Finally, Libert and Vergnaud [13] introduced the notion of traceable proxy re-encryption, where malicious proxies leaking their re-encryption keys can be identified.

2 Definition

We briefly describe the assumptions and underlying encryption schemes that will be used in our constructions, then the definition of CPBRE will be given.

2.1 Pairing and Related Computational Assumption

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two (multiplicatively) cyclic groups of prime order p. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map with the following properties:

- Bilinear: for all $g_1, g_2 \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- Non-degenerate: for some $g \in \mathbb{G}$, $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$, and the bilinear map are efficiently computable.

Our schemes are based on the Decisional Bilinear Diffie-Hellman Exponent (BDHE) problem in $(\mathbb{G}, \mathbb{G}_T)$ [5] – given $2n+1$ elements

$$(\tilde{g}, g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^n}, g^{\alpha^{n+2}}, \ldots, g^{\alpha^{2n}}) \in \mathbb{G}^{2n+1},$$

and an element $R \in \mathbb{G}_T$, decide if $R = e(g, \tilde{g})^{\alpha^{n+1}}$.

In the rest of this paper, we will use g_i to denote the term g^{α^i}.

Definition 1. *The Decisional n-BDHE assumption holds in* $(\mathbb{G}, \mathbb{G}_T)$ *if no polynomial time algorithm* $\mathcal{A}$ *has non-negligible advantage in solving the Decisional n-BDHE problem in* $(\mathbb{G}, \mathbb{G}_T)$*, where the advantage of* $\mathcal{A}$ *is* ε *if*

$$\left| \begin{array}{r} \Pr[\mathcal{A}(\tilde{g}, g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, e(g_{n+1}, \tilde{g})) = 1] \\ - \Pr[\mathcal{A}(\tilde{g}, g, g_1, g_2, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, R) = 1] \end{array} \right| \geq \varepsilon.$$

2.2 Hierarchical Identity-Coupling Broadcast Encryption

Attrapadung, Furukawa and Imai [2] proposed a notion of Hierarchical Identity-Coupling Broadcast Encryption (HICBE). We review its model and security definition here. For an identity $ID = \{id_1, id_2, \ldots, id_l\}$, we denote ID_j as $\{id_1, id_2, \ldots, id_j\}$ for $1 \leq j \leq l$. An HICBE consists of the following algorithms.

- **Setup**(n): on input the maximum number of users, output the public key PK and master secret key MK.
- **KeyGen**(PK, MK, i): on input the public key PK, the master secret key MK and an index $i \in \{1, 2, \ldots, n\}$, output the root secret key sk_i of user i.
- **Derive**$(PK, sk_{i,ID_{l-1}}, i, ID)$: on input the public key PK, the secret key $sk_{i,ID_{l-1}}$ of user i coupling with the $(l-1)$-level identity ID_{l-1}, an index $i \in \{1, 2, \ldots, n\}$ and an l-level identity ID, output the secret key $sk_{i,ID}$.
- **Encrypt**(PK, S, ID, m): on input the public key PK, an index set $S \subseteq \{1, 2, \ldots, n\}$, an identity ID and a message m, output the ciphertext C.
- **Decrypt**$(PK, sk_{i,ID}, i, S, ID, C)$: on input the public key PK, the secret key $sk_{i,ID}$, an index i, a set S, an identity ID and a ciphertext C, output the plaintext m.

The selective identity-and-set security of HICBE is defined by the following game between an adversary $\mathcal{A}$ and a challenger. Both are given n as input.

1. Init. $\mathcal{A}$ picks a set $S^* \subseteq \{1, 2, \ldots, n\}$ and an identity ID^* to be attacked.
2. Setup. Perform **Setup**(n) to get (PK, MK) and give PK to $\mathcal{A}$.
3. Query phase 1. $\mathcal{A}$ can issue the following queries:
   - Extract(i, ID): if $i \notin S^*$ or ID is not a prefix of ID^* or ID^* itself, return $sk_{i,ID} \leftarrow$ **Derive**(PK, sk_i, i, ID) where $sk_i \leftarrow$ **KeyGen**(PK, MK, i); otherwise, return '$\perp$'.
   - Decrypt(i, S, ID, C): return $m \leftarrow$ **Decrypt**$(PK, sk_{i,ID}, i, S, ID, C)$, where $sk_{i,ID} \leftarrow$ **Derive**(PK, sk_i, i, ID), $sk_i \leftarrow$ **KeyGen**(PK, MK, i).
4. Challenge. $\mathcal{A}$ presents (m_0, m_1). Return $C^* \leftarrow$ **Encrypt**(PK, S^*, ID^*, m_b) to $\mathcal{A}$, where $b \in_R \{0, 1\}$.
5. Query phase 2. $\mathcal{A}$ continues making queries as in Query phase 1 except that $\mathcal{A}$ cannot issue the decryption query on (i, S^*, ID, C^*) such that $i \in S^*$ and ($ID = ID^*$ or ID is a prefix of ID^*).
6. Guess. $\mathcal{A}$ outputs the guess $b' \in \{0, 1\}$.

We say $\mathcal{A}$ wins the game if $b' = b$. The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - \frac{1}{2}|$. An HICBE scheme is IND-sID-sSet-CCA-secure if for any probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in this game is negligible. The CPA security is defined in the same way as CCA security except that $\mathcal{A}$ is not allowed to issue the decryption queries.

Attrapadung, Furukawa and Imai provided two constructions based on BGW [5] broadcast encryption. In this paper, we use the HICBE based on BB-IBE [4] to build our CPBRE constructions. The security of this HICBE can be asserted by the following theorem, details can be found in [7] and the full version of [2].

Theorem 1. *Suppose the Decisional n-BDHE assumption holds. The (BGW+BB) HICBE scheme for n users is IND-sID-sSet-CCA-secure.*

2.3 Conditional Proxy Broadcast Re-Encryption

We define our new notion of conditional proxy broadcast re-encryption as follows.

Definition 2. *A conditional proxy broadcast re-encryption scheme consists of the following algorithms:*

- ***Setup***(n)*: used for the generation of the system public key and master secret key of n users. On input the maximum number of users, output the public key PK and master secret key MK.*
- ***KeyGen***(PK, MK, i)*: used for the generation of user i's secret key. On input the public key PK, the master secret key MK and an index $i \in \{1, 2, \ldots, n\}$, output the secret key sk_i.*
- ***Encrypt***(PK, S, w, m)*: used for the generation of a regular ciphertext of m for the set S under condition w. On input the public key PK, an index set $S \subseteq \{1, 2, \ldots, n\}$, a condition w and a message m, output the ciphertext C.*
- ***RKGen***(PK, sk_i, S', w)*: used for the generation of a re-encryption key from i to S' under condition w. On input the public key PK, the secret key sk_i, an index set S' and a condition w, output the re-encryption key $d_{i \to S'|w}$.*
- ***ReEnc***$(PK, d_{i \to S'|w}, i, S, S', w, C)$*: used for the generation of a re-encrypted ciphertext from C. On input the public key PK, the re-encryption key $d_{i \to S'|w}$, the original recipient i, the original set S, the new set S', the condition w and a ciphertext C, output the re-encrypted ciphertext C_R or '$\perp$'.*
- ***Decrypt-I***(PK, sk_i, i, S, w, C)*: used for the decryption of the regular ciphertext C. On input the public key PK, the secret key sk_i, an index i, a set S, a condition w and a ciphertext C, output the plaintext m or '$\perp$'.*
- ***Decrypt-II***$(PK, sk_{i'}, i, i', S, S', w, C_R)$*: used for the decryption of the re-encrypted ciphertext C_R. On input the public key PK, the delegatee's secret key $sk_{i'}$, two indices i and i', two sets S and S', a condition w and a re-encrypted ciphertext C_R, output the plaintext m or '$\perp$'.*

Correctness. *For any integer n, any sets S and S', any indices $i \in S$ and $i' \in S'$, any condition w and any message m,*

$$\Pr\left[\begin{array}{l}\textbf{\textit{Decrypt-I}}(PK, sk_i, i, S, w, C) = m : (PK, MK) \leftarrow \textbf{\textit{Setup}}(n), \\ \quad sk_i \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i), C \leftarrow \textbf{\textit{Encrypt}}(PK, S, w, m)\end{array}\right] = 1,$$

$$\Pr\left[\begin{array}{l}\textbf{\textit{Decrypt-II}}(PK, sk_{i'}, i, i', S, S', w, C_R) = m : \\ \qquad (PK, MK) \leftarrow \textbf{\textit{Setup}}(n), sk_i \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i), \\ \quad sk_{i'} \leftarrow \textbf{\textit{KeyGen}}(PK, MK, i'), d_{i \leftarrow S'|w} \leftarrow \textbf{\textit{RKGen}}(PK, sk_i, S', w), \\ C \leftarrow \textbf{\textit{Encrypt}}(PK, S, w, m), C_R \leftarrow \textbf{\textit{ReEnc}}(PK, d_{i \leftarrow S'|w}, i, S, S', w, C)\end{array}\right] = 1.$$

Now, we proceed to define the security model for CPBRE. Here we consider the security in the replayable CCA sense [9,8,14]. For traditional public key cryptosystems, in such a relaxed security notion, an adversary who can simply modify a given ciphertext into another encryption of the same plaintext is not deemed successful [14]. To define the RCCA security for CPBRE systems, we

here disallow the adversary to ask for a decryption of any re-randomized version of the -encrypted ciphertext re-encrypted from the challenge ciphertext.

Definition 3. *The chosen-ciphertext security of a CPBRE scheme against a static adversary is defined by the following game between an adversary $\mathcal{A}$ and a challenger. Both the challenger and $\mathcal{A}$ are given n as input.*

1. *Init. $\mathcal{A}$ chooses a set $S^* \subseteq \{1, 2, \ldots, n\}$ and a condition w^* that it wants to attack.*
2. *Setup. Perform $\boldsymbol{Setup}(n)$ to get (PK, MK) and give PK to $\mathcal{A}$.*
3. *Query phase 1. We define the following oracles.*
   (a) $\textsc{Extract}(i)$*: return* $sk_i \leftarrow \boldsymbol{KeyGen}(PK, MK, i)$.
   (b) $\textsc{RKExtract}(i, S', w)$*: return* $d_{i \to S'|w} \leftarrow \boldsymbol{RKGen}(PK, sk_i, S', w)$*, where* $sk_i \leftarrow \boldsymbol{KeyGen}(PK, MK, i)$.
   (c) $\textsc{ReEncrypt}(i, S, S', w, C)$*: return* $C_R \leftarrow \boldsymbol{ReEnc}(PK, d_{i \to S'|w}, C)$*, where* $d_{i \to S'|w} \leftarrow \boldsymbol{RKGen}(PK, sk_i, S', w)$ *and* $sk_i \leftarrow \boldsymbol{KeyGen}(PK, MK, i)$.
   (d) $\textsc{Decrypt-I}(i, S, w, C)$*: return* $m \leftarrow \boldsymbol{Decrypt\text{-}I}(PK, sk_i, i, S, w, C)$*, where* $sk_i \leftarrow \boldsymbol{KeyGen}(PK, MK, i)$.
   (e) $\textsc{Decrypt-II}(i, i', S, S', w, C_R)$*: compute* $sk_{i'} \leftarrow \boldsymbol{KeyGen}(PK, MK, i')$ *and return* $m \leftarrow \boldsymbol{Decrypt\text{-}II}(PK, sk_{i'}, i, i', S, S', w, C_R)$.

   $\mathcal{A}$ can issue these queries except
   - $\textsc{Extract}(i)$ *for any* $i \in S^*$ *and*
   - *both* $\textsc{RKExtract}(i, S', w^*)$ *and* $\textsc{Extract}(i')$ *for any* S', $i \in S^*$ *and* $i' \in S'$.
4. *Challenge. $\mathcal{A}$ presents (m_0, m_1). Return $C^* = \boldsymbol{Encrypt}(PK, S^*, w^*, m_b)$ to $\mathcal{A}$, where $b \in_R \{0, 1\}$.*
5. *Query phase 2. $\mathcal{A}$ continues making queries as in the Query phase 1, except for the following queries*
   - $\textsc{Extract}(i)$ *for any* $i \in S^*$;
   - $\textsc{RKExtract}(i, S', w^*)$ *and* $\textsc{Extract}(i')$ *for any* S', $i \in S^*$ *and* $i' \in S'$;
   - $\textsc{Decrypt-I}(i, S^*, w^*, C^*)$ *for any* $i \in S^*$;
   - $\textsc{ReEncrypt}(i, S', w^*, C^*)$ *and* $\textsc{Extract}(i')$ *for any* S', $i \in S^*$ *and* $i' \in S'$;
   - $\textsc{Decrypt-II}(i, i', S^*, S', w^*, C_R^*)$ *for any* S' *and* C_R^**, where* $i \in S^*$, $i' \in S'$ *and*

$$\boldsymbol{Decrypt\text{-}II}(PK, sk_{i'}, i, i', S^*, S', w^*, C_R^*) \in \{m_0, m_1\}.$$

6. *Guess. $\mathcal{A}$ outputs the guess $b' \in \{0, 1\}$. We say $\mathcal{A}$ wins the game if $b' = b$.*

The advantage of $\mathcal{A}$ is defined as $|\Pr[b' = b] - \frac{1}{2}|$. A CPBRE scheme is IND-sCond-sSet-RCCA-secure if for any probabilistic polynomial-time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in this game is negligible. The CPA security is defined in the same way as RCCA security except that $\mathcal{A}$ is not allowed to query ReEncrypt, Decrypt-I *and* Decrypt-II *oracles, similar to the definition in [12].*

3 CPA-Secure CPBRE

We start by presenting a CPA version of our final scheme. Without loss of generality, we assume a condition is always specified for every encryption.

3.1 Construction

The basic CPBRE scheme is described as follows.

– **Setup**(n). Pick a prime p and generates groups $\mathbb{G}, \mathbb{G}_T$, bilinear map e and a generator g as defined in Section 2.1. Randomly choose $\alpha, \gamma \in_R \mathbb{Z}_p$ and compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$. Define the function $F(x) = g_1^x h$, where $h \in_R \mathbb{G}$. Let $H' : \mathbb{G}_T \to \mathbb{G}$ be a target collision resistant (TCR) hash. Compute $v = g^\gamma$. Output the public/secret key:

$$PK = (v, g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, F, H'), \quad MK = \gamma.$$

– **KeyGen**(PK, MK, i). The private key for i is defined as

$$sk_i = g_i^\gamma.$$

– **Encrypt**(PK, S, w, m). For the set $S \subseteq \{1, 2, \ldots, n\}$ and the condition $w \in \mathbb{Z}_p$, pick $t \in_R \mathbb{Z}_p$, the ciphertext for message $m \in \mathbb{G}_T$ is output as

$$C = (c_1, c_2, c_3, c_4) = (m \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F(w)^t).$$

– **RKGen**(PK, sk_i, S', w). Randomly choose $s \in \mathbb{Z}_p$, output

$$d_{i \to S'|w} = (sk_i \cdot F(w)^s, C'), \text{ where } C' \leftarrow \textbf{Encrypt'}(PK, S', g^s),$$

where the algorithm **Encrypt'**(PK, S, m) for $S \subseteq \{1, 2, \ldots, n\}$ and $m \in \mathbb{G}$ picks $t \in_R \mathbb{Z}_p$, $\sigma \in_R \mathbb{G}_T$ and outputs the ciphertext as

$$C' = (c'_0, c'_1, c'_2, c'_3) = (m \cdot H'(\sigma), \sigma \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t).$$

– **ReEnc**$(PK, d_{i \to S'|w}, i, S, S', w, C)$. Let $C = (c_1, c_2, c_3, c_4)$ and $d_{i \to S'|w} = (d, C')$. Compute

$$\tilde{c}_1 = c_1 \cdot e(d \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3) \quad \text{and} \quad \tilde{c}_2 = c_4.$$

The re-encrypted ciphertext is output as

$$C_R = (\tilde{c}_1, \tilde{c}_2, C').$$

– **Decrypt-I**(PK, sk_i, i, S, w, C). Let $C = (c_1, c_2, c_3, c_4)$. Output

$$m = c_1 \cdot e(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3).$$

– **Decrypt-II**$(PK, sk_{i'}, i, i', S, S', w, C_R)$. Let $C_R = (\tilde{c}_1, \tilde{c}_2, C')$ and $C' = (c'_0, c'_1, c'_2, c'_3)$. Recover

$$g^s \leftarrow c'_0 / H'(c'_1 \cdot \frac{e(sk_{i'} \cdot \prod_{j \in S', j \neq i'} g_{n+1-j+i'}, c'_2)}{e(g_{i'}, c'_3)})$$

and output

$$m = \tilde{c}_1 / e(g^s, \tilde{c}_2).$$

Discussion. For the regular encryption, the scheme is just the same as the underlying BGW+BB HICBE scheme, hence we enjoy a constant-size ciphertext. For the re-encryption key under condition w, we encrypt g^s, one of the two elements of $sk_{i,w}$, under the key of the recipient set. Again, its size is independent of the number of delegatees. The same hold trues for re-encrypted ciphertext. The regular decryption procedure is proceeded in **ReEnc**, except for the cancellation of the term $e(g^s, f(w)^t)$. In **Decrypt-II**, the recipient can decrypt C' to get g^s first, and then cancel $e(g^s, f(w)^t)$ to get m.

Correctness. For any integer n, any sets S and S', any indices $i \in S$ and $i' \in S'$, any condition w and any message m, we can see that

– for **Decrypt-I**,

$$\begin{aligned}
& c_1 \cdot e(sk_i \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3) \\
= {} & c_1 \cdot e(g_i^{\gamma} \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t)/e(g_i, (v \cdot \textstyle\prod_{j\in S} g_{n+1-j})^t) \\
= {} & c_1 \cdot e(\textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, g^t)/e(g_i, \textstyle\prod_{j\in S} g^t_{n+1-j}) \\
= {} & c_1/e(g, g^t_{n+1}) = m \cdot e(g_1, g_n)^t/e(g, g^t_{n+1}) = m;
\end{aligned}$$

– for **Decrypt-II**,

$$\begin{aligned}
& \tilde{c}_1/e(g^s, \tilde{c}_2) \\
= {} & (c_1 \cdot e(d \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))/e(g^s, c_4) \\
= {} & (c_1 \cdot e(sk_i \cdot F(w)^s \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))/e(g^s, c_4) \\
= {} & (c_1 \cdot e(sk_i \cdot \textstyle\prod_{j\in S, j\neq i} g_{n+1-j+i}, c_2)/e(g_i, c_3))e(F(w)^s, c_2)/e(g^s, c_4) \\
= {} & m \cdot e(F(w)^s, g^t)/e(g^s, F(w)^t) \qquad \text{(via } \textbf{Decrypt-I}\text{)} \\
= {} & m.
\end{aligned}$$

3.2 Security

Theorem 2. *Suppose the decisional n-BDHE assumption holds and H' is a TCR hash function, the basic CPBRE scheme for n users described above is IND-sCond-sSet-CPA-secure.*

Proof. Suppose there is an adversary $\mathcal{A}$ breaking our basic CPBRE scheme with non-negligible advantage. Initially, $\mathcal{A}$ outputs a selected set S^* and a selected condition w^*. Then we construct another algorithm $\mathcal{B}$ breaking the underlying HICBE scheme as follows.

Given the public key PK of HICBE, $\mathcal{B}$ simulates the security game of basic CPBRE. Initially, $\mathcal{B}$ prepares two tables:

– EX with an index list: a track of Extract queries.
– RK with columns (i, S', w, d): d is the re-encryption key from index i to the set S' under the condition w.

Moreover, we use $*$ to denote the wildcard symbol.

1. Init. $\mathcal{B}$ outputs S^* and w^* as the target set and target identity of HICBE.
2. Setup. Choose a TCR hash function H'. $\mathcal{B}$ sends PK along with H' to $\mathcal{A}$.

3. Query phase 1. $\mathcal{B}$ answers the following queries issued by $\mathcal{A}$:
   (a) EXTRACT(i): if $i \in S^*$, or $(j, S', w^*, *)$ exists in the RK table, where $j \in S^*, i \in S'$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ forwards the query to the key extraction oracle of HICBE, and responds the received sk_i. $\mathcal{B}$ records i in the EX table.
   (b) RKEXTRACT(i, S', w): if there is a tuple $(i, S', w, d_{i \to S'|w})$ in the RK table, $\mathcal{B}$ responds $d_{i \to S'|w}$ to $\mathcal{A}$. Otherwise, we have the following cases:
   - $i \notin S^*$ or $w \neq w^*$: $\mathcal{B}$ queries i's secret key under identity w from the challenger of HICBE, and responds the re-encryption key as the real scheme (except that g^s is given by the challenger). $\mathcal{B}$ records the tuple $(i, S', w, d_{i \to S'|w})$ in the RK table.
   - $i \in S^*, w = w^*$ and $S' \cap EX \neq \emptyset$: $\mathcal{B}$ responds '$\perp$'.
   - $i \in S^*$ and $w = w^*$, but $S' \cap EX = \emptyset$: $\mathcal{B}$ picks $d, d' \in_R \mathbb{G}$ and responds a random re-encryption key $d_{i \to S'|w} = (d, \textbf{Encrypt'}(PK, S', d'))$. $\mathcal{B}$ records the tuple $(i, S', w, d_{i \to S'|w})$ in the RK table.
4. Challenge. $\mathcal{A}$ sends (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ forwards it to the challenger of HICBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ responds C^* to $\mathcal{A}$.
5. Query phase 2. $\mathcal{B}$ responds $\mathcal{A}$'s queries as in phase 1.
6. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

We can see that $\mathcal{B}$ successfully simulates $\mathcal{A}$'s view in the attack except for a case of re-encryption key queries (when $i \in S^*$ and $w = w^*$). For a randomly chosen key $d_{i \to S'|w} = (d, C')$, there must be a value $s' \in \mathbb{Z}_p$ such that $d = sk_i \cdot F(w)^{s'}$. Therefore the problem is equivalent to the indistinguishability of C' and the encryption of some value $g^{s'}$. This is implied by the CPA security of HICBE and the TCR hash function. So, $\mathcal{B}$ has non-negligible advantage in breaking the HICBE scheme. By Theorem 1, $\mathcal{B}$ has non-negligible advantage in breaking the decisional n-BDHE assumption. □

Note that our constructions can withstand the below attack mentioned in [16].

Chain collusion attack. Suppose Alice is the attack target in a proxy re-encryption scheme. Although the adversary $\mathcal{A}$ cannot get the re-encryption key from Alice to Bob and Bob's private key at the same time, $\mathcal{A}$ can get the re-encryption key from Bob to Carol and Carol's private key instead. If Bob's private key can be derived with these two keys, then Alice's private key can be derived as well by just asking for the re-encryption key from Alice to Bob.

In our constructions, we assume a condition w for each encryption. It means all re-encryption keys given to the proxy are coupled with a condition. The collusion of the proxy and the delegatee can recover the decryption key for some condition only. However, the ciphertext C' in the re-encryption key can be decrypted by the root private key only. So the above attack cannot be applied.

4 RCCA-Secure CPBRE

The RCCA-secure CPBRE scheme follows the structure of the basic CPBRE. However, it is challenging to design a RCCA-secure CPBRE scheme because it

involves two kinds of secret keys (regular decryption keys and re-encryption keys) and two kinds of ciphertexts (regular ciphertexts and re-encrypted ciphertexts). Again, we assume a condition is always specified for every encryption.

4.1 Construction

- **Setup**(n). Pick a prime p and generates groups $\mathbb{G}, \mathbb{G}_T$, bilinear map e and a generator g. Randomly choose $\alpha, \gamma \in_R \mathbb{Z}_p$ and compute $g_i = g^{\alpha^i} \in \mathbb{G}$ for $i = 1, \ldots, n, n+2, \ldots, 2n$. Define the functions $bit(i, x)$ be the i-th bit of a bit-string x, $F_1(x) = g_1^x h_1$ and $F_2(x) = u' \prod {u_i}^{bit(i,x)}$ where $h_1, u, u_1, \cdots, u_\eta \in_R \mathbb{G}$ Let $H : \{0,1\}^* \to \mathbb{Z}_p$ and $H' : \mathbb{G}_T \to \mathbb{G}$ be two TCR hash functions. Compute $v = g^\gamma$. Output the public key and the secret key:

$$PK = (v, g, g_1, \ldots, g_n, g_{n+2}, \ldots, g_{2n}, F_1, F_2, H, H'), \quad MK = \gamma.$$

- **KeyGen**(PK, MK, i). The private key for i is defined as

$$sk_i = g_i^\gamma.$$

- **Encrypt**(PK, S, w, m). For the set $S \subseteq \{1, 2, \ldots, n\}$ and the condition $w \in \mathbb{Z}_p$, the ciphertext for message $m \in \mathbb{G}_T$ can be output as

$$C = (c_1, c_2, c_3, c_4, c_5) = (m \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F_1(w)^t, F_2(h)^t),$$

  where $t \in_R \mathbb{Z}_p$ and $h = H(c_1, c_2, c_3, c_4)$.
- **RKGen**(PK, sk_i, S', w). Pick $s \in_R \mathbb{Z}_p$. Output the re-encryption key as

$$d_{i \to S'|w} = (sk_i \cdot F_1(w)^s, C'), \text{ where } C' \leftarrow \textbf{Encrypt'}(PK, S', g^s).$$

  The algorithm **Encrypt'**(PK, S, m) for $S \subseteq \{1, 2, \ldots, n\}$ and $m \in \mathbb{G}$ outputs

$$C' = (c'_0, c'_1, c'_2, c'_3, c'_4) = (m \cdot H'(\sigma), \sigma \cdot e(g_1, g_n)^t, g^t, (v \cdot \prod_{j \in S} g_{n+1-j})^t, F_2(h')^t),$$

  where $t \in_R \mathbb{Z}_p$, $\sigma \in_R \mathbb{G}_T$ and $h' = H(c'_0, c'_1, c'_2, c'_3)$.
- **ReEnc**$(PK, d_{i \to S'|w}, i, S, S', w, C)$. Let $C = (c_1, c_2, c_3, c_4, c_5)$ and $d_{i \to S'|w} = (d, C')$. Compute $h = H(c_1, c_2, c_3, c_4)$. Check that

$$e(c_2, v \cdot \textstyle\prod_{j \in S} g_{n+1-j}) \stackrel{?}{=} e(g, c_3), e(c_2, F_2(h)) \stackrel{?}{=} e(g, c_5), e(c_2, F_1(w)) \stackrel{?}{=} e(g, c_4).$$

  If any of the equations does not hold or $i \notin S$, output '$\perp$'. Otherwise, compute

$$d'_1 = d \cdot F_2(h)^{s'}, \quad \text{and} \quad d'_2 = g^{s'}.$$

  Output the re-encrypted ciphertext

$$C_R = (C, C', d'_1, d'_2).$$

– **Decrypt-I**(PK, sk_i, i, S, w, C). Let $C = (c_1, c_2, c_3, c_4, c_5)$. Compute $h = H(c_1, c_2, c_3, c_4)$ and check that

$$e(c_2, v \cdot \prod_{j \in S} g_{n+1-j}) \stackrel{?}{=} e(g, c_3), e(c_2, F_2(h)) \stackrel{?}{=} e(g, c_5), e(c_2, F_1(w)) \stackrel{?}{=} e(g, c_4).$$

If any of the equations does not hold or $i \notin S$, output '$\perp$'. Otherwise, output

$$m = c_1 \cdot \frac{e(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3)}.$$

– **Decrypt-II**$(PK, sk_{i'}, i, i', S, S', w, C_R)$. Let $C_R = (C, C', d_1', d_2')$ where $C = (c_1, c_2, c_3, c_4, c_5)$ and $C' = (c_0', c_1', c_2', c_3', c_4')$. Compute $h = H(c_1, c_2, c_3, c_4)$ and $h' = H(c_0', c_1', c_2', c_3')$. Check that

$$\begin{gathered} e(c_2, v \cdot \prod_{j \in S} g_{n+1-j}) \stackrel{?}{=} e(g, c_3), \quad e(c_2', v \cdot \prod_{j \in S'} g_{n+1-j}) \stackrel{?}{=} e(g, c_3') \\ e(c_2, F_1(w)) \stackrel{?}{=} e(g, c_4), \quad e(c_2, F_2(h)) \stackrel{?}{=} e(g, c_5), \\ e(c_2', F_2(h')) \stackrel{?}{=} e(g, c_4') \quad \text{and} \quad i \in S. \end{gathered} \tag{1}$$

If any of the equations does not hold, output '$\perp$'. Otherwise, perform

$$g^s = c_0'/H'(c_1' \cdot \frac{e(sk_{i'} \cdot \prod_{j \in S', j \neq i'} g_{n+1-j+i'}, c_2')}{e(g_{i'}, c_3')}).$$

and check that

$$e(d_1', g) \stackrel{?}{=} e(g_i, v) e(F_1(w), g^s) e(F_2(h), d_2'). \tag{2}$$

If the equation does not hold, output '$\perp$'. Otherwise output

$$m = c_1 \cdot \frac{e(d_1' \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3) e(g^s, c_4) e(d_2', c_5)}.$$

Correctness. The decryption of regular ciphertexts is just the same as our basic CPBRE. For the re-encrypted ciphertexts, we first check the decryption of C':

$$\begin{aligned} & c_0'/H'(c_1' \cdot \tfrac{e(sk_{i'} \cdot \prod_{j \in S', j \neq i'} g_{n+1-j+i'}, c_2')}{e(g_{i'}, c_3')}) \\ &= c_0'/H'(c_1' \cdot e(g_{i'}^{\gamma} \cdot \prod_{j \in S', j \neq i'} g_{n+1-j+i'}, g^t)/e(g_{i'}, (v \cdot \prod_{j \in S'} g_{n+1-j})^t)) \\ &= c_0'/H'(c_1' \cdot e(\prod_{j \in S', j \neq i'} g_{n+1-j+i'}, g^t)/e(g_{i'}, \prod_{j \in S'} g_{n+1-j}^t)) \\ &= c_0'/H'(c_1'/e(g, g_{n+1}^t)) = g^s \cdot H'(\sigma)/H'(\sigma \cdot e(g_1, g_n)^t/e(g, g_{n+1}^t)) = g^s. \end{aligned}$$

Once g^s is correctly computed, we can compute the message:

$$\begin{aligned} & c_1 \cdot \tfrac{e(d_1' \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{(e(g_i, c_3) e(g^s, c_4) e(d_2', c_5))} \\ &= c_1 \cdot \tfrac{e(g_i^{\gamma} F_1(w)^s F_2(h)^{s'} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)}{(e(g_i, c_3) e(g^s, F_1(w)^t) e(g^{s'}, F_2(h)^t))} \\ &= c_1 \cdot e(g_i^{\gamma} \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)/e(g_i, (v \cdot \prod_{j \in S} g_{n+1-j})^t) \\ &= c_1 \cdot e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)/e(g_i, \prod_{j \in S} g_{n+1-j}^t) \\ &= c_1/e(g, g_{n+1}^t) = m \cdot e(g_1, g_n)^t/e(g, g_{n+1}^t) = m; \end{aligned}$$

Moreover, it is easy to verify the checking equations are correct. Therefore the correctness of this construction holds.

4.2 Security

In this scheme, we get the RCCA security from the conversion technique of [7]. The scheme involves one more identity level for hash values. This ensures C and C' will not be modified. We did not check the integrity of the re-encrypted ciphertext as a whole. but we use Equation 1 and Equation 2 to check the validity of C, C', d'_1, d'_2 and their relationships to i, S, w. In Lemma 1, we show that if all of these equations hold, C will be decrypted to the original message by d'_1, d'_2 and g^s encrypted in C'. Therefore, the challenger is able to reject any re-randomization of the re-encrypted ciphertexts or any re-encryption of the challenge ciphertext.

Lemma 1. *If a re-encrypted ciphertext $C_R = (C, C', d'_1, d'_2)$ passes Equation 1 and 2, then C_R will be decrypted to the same message as the decryption of C.*

Proof. By Equation 1, we know that C is indeed an encryption under the set S coupling with identity w, C' is an encryption under the set S', and $i \in S$. Then Equation 2 implies that C_R will be decrypted to the same message as the decryption of C, since

$$
\begin{aligned}
& c_1 \cdot \frac{e(d'_1 \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3) e(g^s, c_4) e(d'_2, c_5)} \\
= {} & c_1 \cdot \frac{e(d'_1 \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t)}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} = c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(d'_1, g)^t}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} \\
= {} & c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(g_i, v)^t e(F_1(w), g^s)^t e(F_2(h), d'_2)^t}{e(g_i, c_3) e(g^s, F_1(w)^t) e(d'_2, F_2(h)^t)} \quad \text{(by Equation 2)} \\
= {} & c_1 \cdot \frac{e(\prod_{j \in S, j \neq i} g_{n+1-j+i}, g^t) e(g_i, v)^t}{e(g_i, c_3)} = c_1 \cdot \frac{e(sk_i \cdot \prod_{j \in S, j \neq i} g_{n+1-j+i}, c_2)}{e(g_i, c_3)}.
\end{aligned}
$$

□

Theorem 3. *Suppose the decisional n-BDHE assumption holds and H, H' are two TCR hash functions, the CPBRE scheme for n users described above is IND-sCond-sSet-RCCA-secure.*

Proof. Suppose there is an adversary $\mathcal{A}$ breaking our CPBRE scheme with non-negligible advantage. Initially, $\mathcal{A}$ outputs a selected set S^* and a selected condition w^*. Then we construct another algorithm $\mathcal{B}$ breaking the underlying CCA-secure HICBE scheme (CCA-HICBE) as follows.

Given the public key PK of CCA-HICBE, $\mathcal{B}$ simulates the RCCA game of CPBRE. Initially, $\mathcal{B}$ prepares the following tables:

- EX with (i): a track of EXTRACT queries.
- RK with columns $(i, S', w, d, C', g^s, b_r, b_q)$: the records of re-encryption key (d, C') returned by $\mathcal{B}$, where $C' \leftarrow$ **Encrypt'**(PK, S', g^s); b_r and b_q indicate whether the key is randomly chosen or output by RKEXTRACT oracle.
- RE with column (S'): a track of REENCRYPT(i, S, S', w^*, C^*) queries, where $i \in S^*$ and C^* is the challenge ciphertext.

We use $*$ to denote the wildcard symbol. Note that the intermediate ciphertext C' with a different form cannot be issued to DECRYPT oracle.

1. Init. $\mathcal{B}$ outputs S^* and w^* as the target set and identity of CCA-HICBE.
2. Setup. Pick a TCR hash function H'. $\mathcal{B}$ sends PK along with H' to $\mathcal{A}$.
3. Query phase 1. $\mathcal{B}$ answers the following queries issued by $\mathcal{A}$:
   (a) EXTRACT(i): if $i \in S^*$, or $(j, S', w^*, *, *, *, *, 1)$ exists in the RK table, where $j \in S^*, i \in S'$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ forwards the query to the key extraction oracle of CCA-HICBE, and responds the received sk_i. $\mathcal{B}$ records i in the EX table.
   (b) RKEXTRACT(i, S', w): if there is a tuple $(i, S', w, d, C', g^s, *, *, 1)$ in the RK table, $\mathcal{B}$ responds (d, C') to $\mathcal{A}$. Otherwise, $\mathcal{B}$ answers the re-encryption key for the following cases:
      – $i \notin S^*$ or $w \neq w^*$: $\mathcal{B}$ queries i's secret key under identity w from the challenger of CCA-HICBE, and computes and responds the re-encryption key (d, C') as the real scheme (except that g^s is given by the challenger). $\mathcal{B}$ records $(j, S', w^*, d, C', g^s, 0, 1)$ on RK.
      – $i \in S^*, w = w^*$ and $S' \cap EX \neq \emptyset$: $\mathcal{B}$ responds '$\perp$'.
      – $i \in S^*$ and $w = w^*$, but $S' \cap EX = \emptyset$: if the RK table contains $(i, S', w, d, C', g^s, 1, 0)$, $\mathcal{B}$ responds (d, C') to $\mathcal{A}$ and sets b_q of this tuple to 1. Otherwise, $\mathcal{B}$ responds a random re-encryption key (d, C'), where d is randomly chosen from $\mathbb{G}$ and $C' \leftarrow$ **Encrypt'**(PK, S', g^s) for some random $s \in \mathbb{Z}_p$. $\mathcal{B}$ records the tuple $(i, S', w, d, C', g^s, 1, 1)$ in the RK table. Note that in this case, any EXTRACT(i') query for $i' \in S'$ is forbidden, so the re-encryption key cannot be verified.
   (c) REENCRYPT(i, S, S', w, C): $\mathcal{B}$ proceeds depending on the following cases:
      – $i \notin S^*$ or $w \neq w^*$: if there is no tuple $(i, S', w, *, *, *, *, *)$ on RK, $\mathcal{B}$ performs RKEXTRACT(i, S', w) to get the re-encryption key (d, C') and records $(i, S', w^*, d, C', g^s, 0, 0)$ on RK. Then $\mathcal{B}$ re-encrypts C using the re-encryption key on RK as in the real scheme.
      – $i \in S^*, w = w^*$: if $(i, S', w, *, *, *, 1, 1)$ exists on RK, $\mathcal{B}$ re-encrypts C using the re-encryption key on RK as in the real scheme. Otherwise, $\mathcal{B}$ issues the key extraction query on $(i, (w, h))$ to the challenger of CCA-HICBE where $h = H(c_1, c_2, c_3, c_4)$, and gets back the private key $(d_1, d_2, d_3) = (sk_i F_1(w)^s F_2(h)^{s'}, g^s, g^{s'})$. Then $\mathcal{B}$ computes $C' =$ **Encrypt'**(PK, S', d_2) and responds (C, C', d_1, d_3). $\mathcal{B}$ records $(i, S', w^*, d, C', d_2, 1, 0)$ on RK, where $d \in_R \mathbb{G}$.
   (d) DECRYPT-I(i, S, w, C): $\mathcal{B}$ forwards (i, S, w, C) to the decryption oracle of CCA-HICBE and responds the result to $\mathcal{A}$.
   (e) DECRYPT-II(i, i', S, S', w, C_R): let $C_R = (C, C', d_1', d_2')$. $\mathcal{B}$ issues C' to the decryption oracle of CCA-HICBE scheme to obtain g^s first[1]. If the result is '$\perp$', $\mathcal{B}$ responds '$\perp$' as well. For (i, S', w), check whether there

[1] We cannot issue C' to the decryption oracle of CCA-HICBE directly. However, we can modify the CCA-HICBE scheme (removing the ID part and adding the TCR part) to get a new scheme which encrypts messages as in **Encrypt'**. The resulting scheme is like another way of making the BGW [5] scheme CCA-secure using the conversion technique of [7]. It is easy to show that the obtained scheme is IND-sSet-CCA secure assuming the decisional n-BDHE assumption and a TCR hash function.

exists a tuple $(i, S', w, \hat{d}, \hat{C}, g^s, 1, 1)$ on RK. If there does not exist such tuple, $\mathcal{B}$ decrypts the ciphertext as in the real scheme. Otherwise, check

$$e(d'_1, g) \stackrel{?}{=} e(\hat{d}, g)e(F_2(h), d'_2) \quad \text{and} \quad C' \stackrel{?}{=} \hat{C}.$$

If both equations hold, C_R is re-encrypted by the random re-encryption key given by $\mathcal{B}$. So $\mathcal{B}$ issues a query (i, S, w, C) to the decryption oracle of CCA-HICBE and responds the result to $\mathcal{A}$. Otherwise, $\mathcal{B}$ decrypts the ciphertext as in the real scheme.

4. Challenge. $\mathcal{A}$ sends (m_0, m_1) to $\mathcal{B}$. $\mathcal{B}$ forwards it to the challenger of CCA-HICBE. When the challenger returns ciphertext C^*, $\mathcal{B}$ responds C^* to $\mathcal{A}$.
5. Query phase 2. $\mathcal{A}$ continues making the following queries as in phase 1, except for the restrictions described in the definition.
   (a) EXTRACT(i): If $i \in S'$ for some S' on RE, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1.
   (b) RKEXTRACT(i, S', w): $\mathcal{B}$ responds as in Query phase 1.
   (c) REENCRYPT(i, S, S', w, C): If $C = C^*, S = S^*, w = w^*$, $i \in S$ and $S' \cap EX \neq \emptyset$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1. $\mathcal{B}$ records S' on RE if it did not respond '$\perp$' and $C = C^*$.
   (d) DECRYPT-I(i, S, w, C): If if $C = C^*, S = S^*, w = w^*$ and $i \in S$, $\mathcal{B}$ responds '$\perp$'. Otherwise, $\mathcal{B}$ responds the queries as in Query phase 1.
   (e) DECRYPT-II(i, i', S, S', w, C_R): let $C_R = (C, C', d'_1, d'_2)$. $\mathcal{B}$ responds the queries as in Query phase 1 except for the case $C = C^*, S = S^*, w = w^*$ and $i \in S$. For the latter case, $\mathcal{B}$ simply responds with '$\perp$'. We explain this by considering the following two cases for $i \in S$:
      – C_R does not pass Equation 1 or 2: $\mathcal{B}$ should return '$\perp$'.
      – C_R passes Equation 1 and 2: by Lemma 1, C_R must be decrypted into m_b. According to our security definition, this query is forbidden.
6. Guess. When $\mathcal{A}$ outputs the guess b', $\mathcal{B}$ outputs b'.

$\mathcal{A}$ is successfully simulated except for randomly chosen re-encryption keys. For a randomly chosen re-encryption key (d, C'), there must be a value $s' \in \mathbb{Z}_p$ such that $d = sk_i \cdot F(w)^{s'}$. Therefore the distinguishability of these keys is equivalent to the distinguishability of C' and the encryption of some value $g^{s'}$. By the CCA security of CCA-HICBE and the TCR hash function, this distinguishability is negligible. Moreover, if $\mathcal{A}$ uses a randomly chosen re-encryption key to re-encrypt a ciphertext and issues it to DECRYPT-II oracle, $\mathcal{B}$ can make a decryption query on the original ciphertext and respond with a correct message. So, $\mathcal{B}$ has non-negligible advantage in breaking the CCA-HICBE, and hence breaking the decisional n-BDHE assumption, by Theorem 1. □

5 RCCA-Secure Identity-Based Proxy Re-Encryption

Since CPBRE is a generalization of PRE, we can build RCCA-secure IB-PRE from our RCCA-secure CPBRE by letting the broadcast size be 1 and w be user ID. We briefly explain the changes here. Details are deferred to the full paper.

The key generation center executes **Setup**. For user key generation, each user gets a 1-level secret key $(g_1^{\gamma} F_1(id)^s, g^s)$ for his identity id. Everyone has to encrypt a message under a 2-level identity: recipient's identity and a dummy identity t. In **RKGen**, g^s is encrypted under the recipient's (1-level) identity. Then a proxy colluding with a third party can only get the decryption key of delegator's 2-level identity, which cannot be used to get g^s.

In short, a user Alice with the secret key $(g_1^{\gamma} F_1(\text{“Alice”})^s, g^s)$ delegates the decryption right to Bob by giving the re-encryption key $(g_1^{\gamma} F_1(\text{“Alice”})^s F_2(t)^{s'}$, **Encrypt'**$(PK, \text{“Bob”}, g^s), g^{s'})$ to a proxy. The proxy re-encrypts an Alice's ciphertext C in the form (C, C', d_1', d_2'). Then Bob can use his own secret key $(g_1^{\gamma} F_1(\text{“Bob”})^{s'}, g^{s'})$ to decrypt C' first, and get the decryption of C. The security proof is almost the same as that for the RCCA-secure CPBRE scheme.

We say that a PRE scheme is multi-use if the proxy can re-encrypt a ciphertext multiple times, e.g. re-encrypt from Alice to Bob, then re-encrypt the result from Bob to Carol. To do that in our construction, the proxy only needs to re-encrypt C' to further transform the re-encrypted ciphertext.

6 Conclusions

We introduce conditional proxy broadcast re-encryption, which allows a user to delegate the decryption rights of ciphertexts to a group of users, restricted to a certain condition, via the help of a proxy. Our final scheme is unidirectional and secure against replayable chosen-ciphertext attacks in the standard model, which also gives a unidirectional ID-based proxy re-encryption scheme.

Acknowledgement

Thanks to Junzuo Lai for the discussion that improves our initial construction.

References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. In: Proceedings of the Network and Distributed System Security Symposium (NDSS 2005). The Internet Society (2005)
2. Attrapadung, N., Furukawa, J., Imai, H.: Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 161–177. Springer, Heidelberg (2006)
3. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (1998)
4. Boneh, D., Boyen, X.: Efficient selective-id secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)

5. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005)
6. Boneh, D., Goh, E.-J., Matsuo, T.: Proposal for P1363.3 proxy re-encryption (2006), http://grouper.ieee.org/groups/1363/IBC/submissions
7. Boyen, X., Mei, Q., Waters, B.: Direct chosen ciphertext security from identity-based techniques. In: Proceedings of the 12th ACM Conference on Computer and Communications Security - CCS 2005, pp. 320–329. ACM Press, New York (2005)
8. Canetti, R., Hohenberger, S.: Chosen-ciphertext secure proxy re-encryption. In: Proceedings of ACM Conference on Computer and Communications Security (CCS 2007), pp. 185–194. ACM Press, New York (2007)
9. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (2003)
10. Chu, C.-K., Tzeng, W.-G.: Identity-based proxy re-encryption without random oracles. In: Garay, J.A., Lenstra, A.K., Mambo, M., Peralta, R. (eds.) ISC 2007. LNCS, vol. 4779, pp. 189–202. Springer, Heidelberg (2007)
11. Deng, R.H., Weng, J., Liu, S., Chen, K.: Chosen-ciphertext secure proxy re-encryption without pairings. In: Franklin, M.K., Hui, L.C.K., Wong, D.S. (eds.) CANS 2008. LNCS, vol. 5339, pp. 1–17. Springer, Heidelberg (2008)
12. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
13. Libert, B., Vergnaud, D.: Tracing malicious proxies in proxy re-encryption. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing 2008. LNCS, vol. 5209, pp. 332–353. Springer, Heidelberg (2008)
14. Libert, B., Vergnaud, D.: Unidirectional chosen-ciphertext secure proxy re-encryption. In: Cramer, R. (ed.) PKC 2008. LNCS, vol. 4939, pp. 360–379. Springer, Heidelberg (2008)
15. Matsuo, T.: Proxy re-encryption systems for identity-based encryption. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 247–267. Springer, Heidelberg (2007)
16. Shao, J., Cao, Z.: CCA-secure proxy re-encryption without pairings. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 357–376. Springer, Heidelberg (2009)
17. Tang, Q.: Type-based proxy re-encryption and its construction. In: Soomaruga, G. (ed.) Formal Theories of Information. LNCS, vol. 5363, pp. 130–134. Springer, Heidelberg (2008)
18. Weng, J., Deng, R.H., Ding, X., Chu, C.-K., Lai, J.: Conditional proxy re-encryption secure against chosen-ciphertext attack. In: Proceedings of ACM Symposium on Information, Computer & Communication Security (ASIACCS 2009). ACM Press, New York (to appear, 2009)
19. Weng, J., Deng, R.H., Liu, S., Chen, K., Lai, J., Wang, X.: Chosen-ciphertext secure proxy re-encryption schemes without pairings. Technical report, Cryptology ePrint Archive: Report 2008/509 (Version 3) (2008)

Security on Hybrid Encryption with the Tag-KEM/DEM Framework

Toshihide Matsuda[1], Ryo Nishimaki[2], Akira Numayama[1], and Keisuke Tanaka[1]

[1] Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{matsuda5,keisuke,numayam4}@is.titech.ac.jp
[2] NTT Laboratories,
3-9-11 Midori-cho Musashino-shi Tokyo 180-8585, Japan
nishimaki.ryo@lab.ntt.co.jp

Abstract. The tag-KEM/DEM framework has been proposed by Abe, Gennaro, Kurosawa, and Shoup to explain why the Kurosawa-Desmedt PKE is secure in the sense of IND-CCA2, yet the KEM part are not secure in the sense of IND-CCA2. They have concluded that the Kurosawa-Desmedt KEM satisfies the IND-CCA2 security for tag-KEM. They have shown that an IND-CCA2 secure PKE system can be constructed from an IND-CCA2 tag-KEM system and an IND-OT secure DEM system.

Herranz, Hofheinz and Kiltz have shown the necessary and sufficient conditions for the KEM/DEM framework. They also have studied implications and separations among the security notions of KEM.

In this paper, we study the necessary and sufficient conditions for the tag-KEM/DEM framework. Moreover, we study implications and separations among the security notions of tag-KEM. By these studies, we show gaps between KEM and tag-KEM about weak and strong non-malleability with respect to the necessary and sufficient conditions in order to obtain the same security levels.

1 Introduction

1.1 Background

An important task of cryptography is to transmit messages safely through a public channel. For this purpose, there exist two ways, i.e., *public-key encryption* (PKE) systems and *symmetric-key encryption* (SKE) systems. *Hybrid encryption* systems are constructed from the combination from both sides, which are employed to utilize the advantage of both of them. In these systems, a ciphertext consists of two parts: a ciphertext of a symmetric key with the public-key encryption system and a ciphertext of the message with the symmetric-key encryption system by using the symmetric key. Constructions of hybrid encryption systems have been improved and formalized as *key encapsulation mechanism* (KEM) and *data encapsulation mechanism* (DEM) by Cramer and Shoup [4,10].

In the KEM/DEM framework, if the KEM and DEM systems are secure in the sense of IND-CCA2, then the hybrid encryption system is also secure in the sense of IND-CCA2 [4]. It was believed that these conditions were necessary and sufficient. However, Kurosawa and Desmedt have constructed a hybrid encryption system such that the

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 343–359, 2009.

KEM system does not satisfy the IND-CCA2 security, but the hybrid encryption system constructed from it satisfies the IND-CCA2 security [7, 6].

Abe, Gennaro, Kurosawa, and Shoup have proposed the tag-KEM/DEM framework in order to formalize the security notions that the Kurosawa-Desmedt hybrid encryption system satisfies[1]. The tag-KEM/DEM framework applies the output of DEM to the input of tag-KEM as a tag. In this framework, if the tag-KEM system satisfies the IND-CCA2 security and the DEM system satisfies the IND-OT security, the hybrid encryption system is secure in the sense of IND-CCA2. They explained why the Kurosawa-Desmedt system satisfies the IND-CCA2 security within the tag-KEM/DEM framework.

Herranz, Hofheinz, and Kiltz [5] have classified the security notions of KEM into nine security notions, IND-{CPA, CCA1, CCA2}, wNM-{CPA, CCA1, CCA2}, sNM-{CPA,CCA1,CCA2}, and the security notions of DEM into ten security notions, IND-{OT, CPA, CCA1, OTCCA2, CCA2}, NM-{OT, CPA, CCA1, OTCCA2, CCA2}. They have analyzed implications and separations among these security notions, and shown the necessary and sufficient conditions for the security levels of the hybrid encryption systems.

1.2 Motivations

It is not clear the necessary and sufficient conditions for the tag-KEM/DEM framework, in contrast to the KEM/DEM framework [5]. There are differences between the KEM/DEM framework and the tag-KEM/DEM framework as illustrated by the Kurosawa-Desmedt system, for example. Therefore, it is important to study the necessary and sufficient conditions for the tag-KEM/DEM framework and to understand the differences between the KEM/DEM and the tag-KEM/DEM frameworks for several security notions. In the contrast to the KEM/DEM framework, we may not need strong security notions to construct a hybrid encryption system which satisfies a certain security notion with the tag-KEM/DEM framework.

1.3 Our Contributions

(1) We define the security notions on non-malleability for tag-KEM, i.e. weak non-malleability (wNM), comparison non-malleability (CNM), and simulation non-malleability (SNM), by following the definitions of KEM [5, 8]. We define, in addition to the above notions, the security notion related to non-malleability for tag-KEM, i.e. indistinguishability under parallel chosen ciphertext attack based on non-malleability (PNM), by following the definitions of KEM [8].

(2) We prove the equivalence among the CNM-ATK security, the SNM-ATK security, and the PNM-ATK security for tag-KEM, as in the case of KEM [8]. We call these security notions strong non-malleability(sNM). We note that the wNM-ATK security is not equal to the sNM-ATK security.

(3) We analyze implications and separations between each pair of the security notions on tag-KEM, as in the case of KEM [5]. This result is described in **Fig. 1**.

(4) We study the necessary and sufficient conditions for the security level of the hybrid encryption system with the tag-KEM/DEM framework, as in the case of KEM [5]. This result is described in **Fig. 2**.

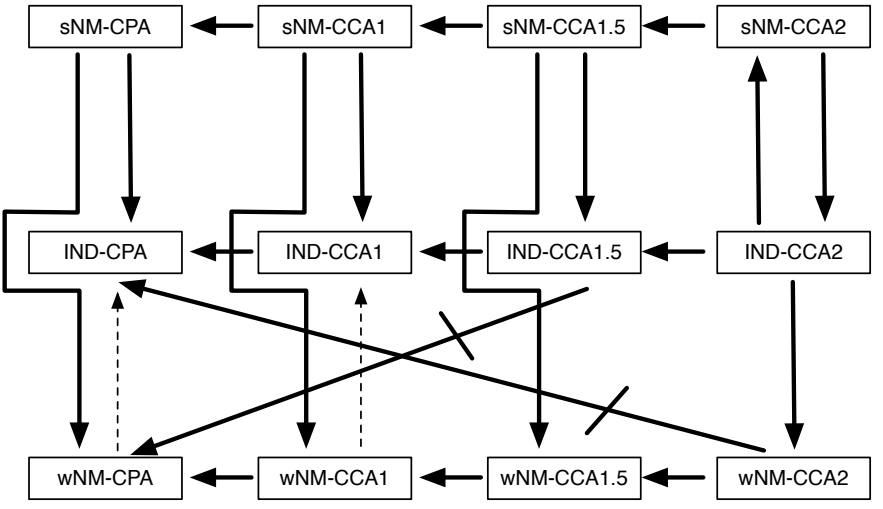


Fig. 1. Relations among the security notions of tag-KEMs

	IND-{OT,CCA2} DEM
wNM-{CPA, CCA1, CCA1.5, CCA2} tag-KEM	< IND-CPA PKE (Theorem 7) √
IND-CPA tag-KEM	≥ IND-CPA PKE (This is trivial from [1].) < NM-CPA, IND-CCA1 PKE
sNM-CPA tag-KEM	≥ NM-CPA PKE (Theorem 6) √ < IND-CCA1 PKE (the proof in the full paper)
IND-{CCA1,CCA1.5} tag-KEM	≥ IND-CCA1 PKE (This is trivial from [1].) < NM-CPA PKE (the proof in the full paper)
sNM-{CCA1,CCA1.5} tag-KEM	≥ NM-CCA1 PKE (Theorem 6) √ < IND-CCA2 PKE (the proof in the full paper)
IND-CCA2 tag-KEM	≥ IND-CCA2 PKE [1] ⋆

Fig. 2. Necessary and sufficient conditions for hybrid encryptions

In **Fig. 1**, solid arrows from X to Y mean that every tag-KEM system which satisfies the security notion X also satisfies the security notion Y. Dotted arrows from X to Y means that the security notion X always implies the security notion Y in the case of KEM system. "$X \nrightarrow Y$" means that the security notion X does not always imply the security notion Y. From **Fig. 1**, for each two notions, an implication or a separation can be derived.

In **Fig. 2**, the symbol $\geq$ means that, for any combination of tag-KEM and DEM with security notions in the cells, the hybrid encryption system from them satisfies the security notions in the cells. The symbol $<$ means that there exists a certain combination of tag-KEM and DEM with the security notions in the cells, such that PKE from them does not satisfy the security notions in the cells. The mark $\star$ indicates a known result. The mark $\surd$ indicates our contributions.

Definitions of the Security Notions on Tag-KEM. We define the security notions on non-malleability for tag-KEM, i.e. weak non-malleability (wNM), comparison non-malleability (CNM), and simulation non-malleability (SNM), by following the definitions of KEM [5, 8]. We define, in addition to the above notions, the security notion related to non-malleability for tag-KEM, i.e. indistinguishability under parallel chosen ciphertext attack based on non-malleability (PNM).

We only consider adversaries which are *non-copying* in experiments about non-malleability for tag-KEM (resp. PKE): In each experiment of non-malleability for tag-KEM (resp. PKE), adversary does not output the same encapsulation and tag (resp. ciphertext) as the challenge encapsulation and tag (resp. ciphertext). It was proved that three definitions, except for weak non-malleability, are equivalent in the case of public-key encryption systems and KEM for non-copying adversaries [2, 3, 8]. We prove these equivalence on tag-KEM for non-copying adversaries, and call these security notions strong non-malleability (sNM). We note that three definitions for copying adversaries are not equivalent in the case of PKE [9].

In contrast to the definitions of the security notions for KEM and PKE which are formalized in two stages, the definitions of the security notions for tag-KEM are formalized in three stages. So, we classify the attacks into four types, i.e. CPA, CCA1, CCA1.5, and CCA2. These notions of the security are classified by the adversary's access to the decryption oracle in each stage. Furthermore, we classify the goals into IND, sNM, and wNM as in the case of KEM [5]. Consequently, the notions of the security are classified into twelve types as in **Fig. 1**.

The security notion of CPA is that the adversary does not always have access to the decryption oracle, and the security notion of CCA2 is that the adversary always has access to the decryption oracle without sending to the challenge ciphertext. The security notions of CCA1 and CCA1.5 is limited with access to the decryption oracle. The former can access the decryption oracle only in the first stage, and cannot in the second and third stages. The latter can access the decryption oracle only in the first and second stages, and cannot in the third stage.

Implications and Separations among the Security Notions on the Tag-KEM. In the case of KEM, the wNM-{CPA, CCA1} security implies the IND-{CPA, CCA1} security and the inverse implication does not hold[1]. In other words, the wNM-{CPA, CCA1} security is a strictly stronger security notion than the IND-{CPA, CCA1} security.

In contrast, in the case of tag-KEM, these implications do not hold. That is, even the wNM-CCA2 security does not imply the IND-CPA security in the case of tag-KEM. The wNM-CCA2 security does not also imply the sNM-CPA security in the

[1] The wNM-CCA2 security does not always imply the IND-CCA2 security, even for KEM.

cases of both KEM and tag-KEM. The other implications are almost similar in the case of KEM [5].

Sufficient and Necessary Conditions for Hybrid Encryption Systems. Let KEM, TKEM, and DEM denote a KEM system, a tag-KEM system, and a DEM system, respectively. Let $\mathsf{PKE}^{\mathsf{KEM,DEM}}$ denote a hybrid encryption system which consists of KEM and DEM with the KEM/DEM framework. Let $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ denote a hybrid encryption system which consists of TKEM and DEM with the tag-KEM/DEM framework.

In the case of the KEM/DEM framework, it is known that the sufficient and necessary conditions for the security of the hybrid encryption systems [5]. However, in the case of the tag-KEM/DEM framework, it is only known that if TKEM satisfies the IND-CCA2 security and DEM satisfies the IND-OT security, $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ satisfies the IND-CCA2 security [1].

We make it clear the characterization of the other security notions for the tag-KEM/DEM framework with the twelve notions of tag-KEM and two notions of DEM. That is, when a tag-KEM system satisfies a certain security notion and a DEM system satisfies a certain security notion, we characterize the security notions that the hybrid encryption systems satisfy.

We consider only the two security notions IND-{OT, CCA2} on DEM. The reason is that the IND-OT security is considered as the weakest security notion and the IND-CCA2 security is considered as the strongest security notion. The IND-OT security gives a lower bound on the security level of the hybrid encryption system with tag-KEM/DEM framework, and the IND-CCA2 security gives an upper bound on the security level of the hybrid encryption system. Another reason is that the security level of hybrid encryption systems from the tag-KEM/DEM framework is hardly affected by the security level of DEM as shown in **Fig. 2**.

Discussion on the Gap between KEM/DEM and Tag-KEM/DEM An example was shown that there exists $\mathsf{PKE}^{\mathsf{KEM,DEM}}$ which does not always satisfy the IND-CCA2 security even if KEM satisfies the IND-CCA2 security and DEM satisfies the IND-OT security [5]. As far as we know, this is the first result on the gap between the KEM/DEM and the tag-KEM/DEM frameworks. We describe this gap as indicated by the mark $\star$ in **Fig. 2**.

Nagao, Manabe, and Okamoto have proposed the security notions of non-malleability for KEM [8]. Herranz et al. called those notions *strong non-malleability* (sNM) [5]. They showed that $\mathsf{PKE}^{\mathsf{KEM,DEM}}$ may not be NM-{CPA, CCA1} secure in the KEM/DEM framework, even if KEM is sNM-{CPA, CCA1} secure and DEM is IND-OT secure. In contrast, we show that $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is NM-{CPA, CCA1} secure in the tag-KEM/DEM framework, if TKEM is sNM-{CPA, CCA1} secure and DEM is IND-OT secure.

We prove that there exists TKEM and DEM such that $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is not secure in the sense of the IND-CPA security even if TKEM satisfies the wNM-CCA2 security and DEM satisfies the IND-CCA2 security. This gap between KEM and tag-KEM depends on their different structures for a tag.

We simply define the wNM security on tag-KEM by following the definition in the case of KEM from [5] and obtain the negative fact on the wNM security (i.e. wNM-CCA2 tag-KEM + IND-CCA2 DEM $\nRightarrow$ IND-CPA PKE). This security notion does not seem to grasp suitably the notion of non-malleability in the case of the tag-KEM/DEM framework. It may be interesting to consider other security notions which have weaker properties than the properties of the sNM security and adequately grasp the notion of non-malleability.

The most noticeable result in this paper is the fact that, if a tag-KEM system satisfies the sNM-{CPA,CCA1} security and a DEM system satisfies the IND-OT security then the PKE system with the tag-KEM/DEM framework satisfies the NM-{CPA,CCA1} security. The reason is that this fact shows the essential differences between the tag-KEM/DEM framework and the KEM/DEM framework. The other sufficient and necessary conditions for the hybrid encryption system are almost similar to [5], and their proofs are almost similar to [5].

1.4 Organization

In Section 2, we review some definitions of tag-KEM, DEM, and PKE. Then, we recall the tag-KEM/DEM framework. In Section 3, we define the security notions on tag-KEM, i.e. the wNM security, the CNM security, the SNM security, and the PNM security. In Section 4, we present three theorems. One of them means the equivalence on the CNM security, the SNM security, and the PNM security. The others mean the difference between the KEM/DEM and the tag-KEM/DEM frameworks. One theorem is positive and the other is negative. In Section 5, we give their proof sketches.

2 Preliminaries

Notations If $k \in \mathbb{N}$ then 1^k denote the string of k ones. If S is a set then $s \leftarrow S$ represents an operation of picking an element s in S at uniformly random. We write l-dimensional vectors as $\langle a_i \rangle_{i=1}^{l} = (a_1, a_2, \ldots, a_l)$ and similarly $\langle a_i, b_i \rangle_{i=1}^{l} = ((a_1, b_1), (a_2, b_2), \ldots, (a_l, b_l))$. We write $y \leftarrow A(x_1, x_2, \cdots, x_n)$ to indicate that A is a probabilistic algorithm which receives n inputs $x_1, x_2, \cdots, x_n$ and then outputs y. In the case that an algorithm A has access to oracles $O_1(\cdot), O_2(\cdot), \cdots, O_m(\cdot)$, we write it as $A^{O_1(\cdot), O_2(\cdot), \cdots, O_m(\cdot)}(x_1, x_2, \cdots, x_n)$.

Requirement of Adversaries in each Experiment We only consider adversaries which are *legitimate* in each experiment for tag-KEM (resp. DEM and PKE): in every CCA2 experiment for tag-KEM (resp. DEM and PKE), adversaries does not query to its decryption oracle with the same challenge encapsulation and the tag (resp. the challenge ciphertext) in the CCA2 experiment.

2.1 Key Encapsulation Mechanism with Tag

A *key encapsulation mechanism with tag* (tag-KEM) system $\mathsf{TKEM} = (\mathsf{TKEM.Gen}, \mathsf{TKEM.Key}, \mathsf{TKEM.Enc}, \mathsf{TKEM.Dec})$ consists of four algorithms. Formally, $(pk, sk) \leftarrow$

$\mathsf{TKEM.Gen}(1^k)$ is a probabilistic algorithm that generates a public key pk and a private key sk. The public key pk is used to encapsulate a session key, and the secret key sk is used to decapsulate an encapsulation. $(w, K) \leftarrow \mathsf{TKEM.Key}(pk)$ is a probabilistic algorithm that generates a session key $K \in \mathcal{K}$ and internal state information w. The session key K is used for encryption of DEMs. $C \leftarrow \mathsf{TKEM.Enc}(w, \tau)$ is a probabilistic algorithm that encrypts K into C by using τ, where τ is called a tag. We describe a tag space as $\mathcal{T}$. $K \leftarrow \mathsf{TKEM.Dem}(sk, C, \tau)$ is a deterministic algorithm that recovers K from C and τ. For every sk, K, C, and τ associated with the above three functions, we claim $\mathsf{TKEM.Dec}(sk, C, \tau) = K$.

Here, we only consider tag-KEM systems that produce uniformly distributed keys. In the other words, we require that for every public key pk, the second element of an output of $\mathsf{TKEM.Enc}(pk)$ has the uniform distribution on key space $\mathcal{K}$. We keep generality on our discussion since most KEM systems and tag-KEM systems have this property. Furthermore, key space $\mathcal{K}$ is restricted to $l(k)$ bits space $\{0,1\}^{l(k)}$, where k is the security parameter and l is some polynomial. We only consider this setting, since tag space $\mathcal{T}$ is equal to DEM's ciphertext space in the case of the tag-KEM/DEM framework.

2.2 Data Encapsulation Mechanism

A *data encapsulation mechanism* (DEM) system $\mathsf{DEM} = (\mathsf{DEM.Enc}, \mathsf{DEM.Dec})$ consists of two algorithms and key space $\mathcal{K}$. Formally, $\tau \leftarrow \mathsf{DEM.Enc}(K, m)$ is a probabilistic algorithm that encrypts m into τ by using K, where a key K is chosen uniform randomly in $\mathcal{K}$. $m \leftarrow \mathsf{DEM.Dem}(K, \tau)$ is a deterministic algorithm that recovers m from K and τ. For every K, m, and τ associated with the above $\mathsf{DEM.Enc}$, we claim $\mathsf{DEM.Dec}(K, \tau) = m$. We require key space $\mathcal{K}$ is restricted to $l(k)$ bits space $\{0,1\}^{l(k)}$, where k is the security parameter and l is some polynomial.

2.3 Public Key Encryption

A *public-key encryption* (PKE) $\mathsf{PKE} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ consists of three algorithms. Formally, $(pk, sk) \leftarrow \mathsf{PKE.Gen}(1^k)$ is a probabilistic algorithm that generates a public key pk and a secret key sk. The public key pk is published and used to encrypt messages. The secret key sk is unpublished and used to decrypt the ciphertext into the message. $\gamma \leftarrow \mathsf{PKE.Enc}(pk, m)$ is a probabilistic algorithm that encrypts m into γ by using pk. γ is called a ciphertext. $m \leftarrow \mathsf{PKE.Dec}(sk, \gamma)$ is a deterministic algorithm that recovers m from γ by using sk. For every pk, m, and γ associated with the above two algorithm, we claim $\mathsf{PKE.Dec}(sk, \gamma) = m$.

2.4 PKE Constructed from the Tag-KEM/DEM Framework

Abe et al. proposed the tag-KEM/DEM framework to capture the formulation for the Kurosawa-Desmedt KEM/DEM [1,7,6]. In this paper, we only consider public-key encryption systems constructed from the tag-KEM/DEM framework. Let $\mathsf{TKEM} = (\mathsf{TKEM.Gen}, \mathsf{TKEM.Key}, \mathsf{TKEM.Enc}, \mathsf{TKEM.Dec})$ be a tag-KEM system and $\mathsf{DEM} = (\mathsf{DEM.Enc}, \mathsf{DEM.Dec})$ a DEM system. We assume that the tag-KEM's key space is equal to the DEM's key space. Then we can construct PKE system $\mathsf{PKE}^{\mathsf{TKEM.DEM}} = (\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ as follows.

$\mathsf{PKE.Gen}(1^k)$:	$\mathsf{PKE.Enc}(pk, m)$:	$\mathsf{PKE.Dec}(sk, \gamma)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$	$(w, K) \leftarrow \mathsf{TKEM.Key}(pk)$	parse $(C, \tau) \leftarrow \gamma$
return (pk, sk)	$\tau \leftarrow \mathsf{DEM.Enc}(K, m)$	$K \leftarrow \mathsf{TKEM.Dec}(sk, C, \tau)$
	$C \leftarrow \mathsf{TKEM.Enc}(w, \tau)$	$m \leftarrow \mathsf{DEM.Dec}(K, \tau)$
	$\gamma \leftarrow (C, \tau)$	return m
	return γ	

3 The Security Notions of Tag-KEM

In this section, we define the security notions on tag-KEM, i.e. wNM-ATK, CNM-ATK, SNM-ATK, and PNM-ATK. In order to simplify the description, we use unified definitions on ATK $\in \{$CPA, CCA1, CCA1.5, CCA2$\}$ in each experiment as follows.

If ATK = CPA, then $O_1 = O_2 = O_3 = \perp$.
If ATK = CCA1, then $O_1 = \mathsf{TKEM.Dec}(sk, \cdot, \cdot)$ and $O_2 = O_3 = \perp$.
If ATK = CCA1.5, then $O_1 = O_2 = \mathsf{TKEM.Dec}(sk, \cdot, \cdot)$ and $O_3 = \perp$.
If ATK = CCA2, then $O_1 = O_2 = O_3 = \mathsf{TKEM.Dec}(sk, \cdot, \cdot)$.

We denote by n a polynomial in the security parameter k in the following definitions.

3.1 wNM-ATK Tag-KEM

We define *weak non-malleability* for tag-KEM by following in [5]. Roughly speaking, the wNM experiment means that the adversary A cannot modify the challenge encapsulation C^* and the tag τ^* into other encapsulations and tags related to (C^*, τ^*) without receiving challenge keys (K_c^*, K_{1-c}^*). The most importance in this definition is that the adversary does not receive the challenge keys (K_c^*, K_{1-c}^*). As explained in Section 1.3, one of the differences between KEM and tag-KEM depends on the fact that the adversary which does not know the real key K_0^* must make a tag τ^*.

Definition 1 (wNM-ATK tag-KEM). Let TKEM be a tag-KEM system. We define the advantage $\mathsf{Adv}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k) \stackrel{\text{def}}{=} |\Pr[\mathsf{Expt}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k,0) = 1] - \Pr[\mathsf{Expt}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k,1) = 1]|$ for a legitimate and non-copying adversary A as follows. If for every legitimate and non-copying adversary $A = (A_1, A_2, A_3)$, $\mathsf{Adv}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k)$ is negligible in the security parameter k, then we say that TKEM satisfies the wNM-ATK security, where ATK $\in \{$CPA, CCA1, CCA1.5, CCA2$\}$.

$\mathsf{Expt}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k, b)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$; $st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$
$(w, K_0^*) \leftarrow \mathsf{TKEM.Key}(pk)$; $K_1^* \leftarrow \mathcal{K}$
$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1)$; $C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$
$(Rel, \langle C_i, \tau_i \rangle_{i=1}^n) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$
for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$
if $Rel(K_b^*, \langle K_i \rangle_{i=1}^n) = 1$, then return 1
else return 0

We can define the wNM-ATK experiment which consists of only two stages by merging the first and the second stages. However, we choose the definition which consists of three stages in order to integrate the definitions of the IND-ATK and the sNM-ATK experiments.

3.2 sNM-ATK Tag-KEM

We define *strong non-malleability* (sNM) for tag-KEM as *comparison non-malleability*, *simulation non-malleability*, and *indistinguishability under parallel chosen ciphertext attack based on non-malleability*. We show three security notions are equivalent in Theorem 6. We only give a proof sketch in this paper, and a detailed proof can be found full version of this paper. In contrast, the wNM-ATK security is strictly weaker property than the sNM-ATK security.

CNM-ATK Tag-KEM. We define *comparison non-malleability* (CNM) for tag-KEM by following [8]. Roughly speaking, the CNM experiment expresses whether the adversary A can modify the challenge encapsulation C^* and the tag τ^* into other encapsulations and tags related to (C^*, τ^*).

Definition 2 (CNM-ATK tag-KEM). Let TKEM be a tag-KEM system. We define the advantage $\mathsf{Adv}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k) \stackrel{\text{def}}{=} |\Pr[\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k) = 1] - \Pr[\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k) = 1]|$ for a legitimate and non-copying adversary A as follows. If for every legitimate and non-copying adversary adversary $A = (A_1, A_2, A_3)$, $\mathsf{Adv}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k)$ is negligible in the security parameter k, then we say that TKEM satisfies the CNM-ATK security, where $\text{ATK} \in \{\text{CPA}, \text{CCA1}, \text{CCA1.5}, \text{CCA2}\}$.

$\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k)$:	$\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},A}(k)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$
$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$	$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$
$(w, K_0^*) \leftarrow \mathsf{TKEM.Key}(pk)$; $K_1^* \leftarrow \mathcal{K}$	$(w, K) \leftarrow \mathsf{TKEM.Key}(pk)$; $K_0^*, K_1^* \leftarrow \mathcal{K}$
$c \leftarrow \{0,1\}$; $X \leftarrow (K_c^*, K_{1-c}^*)$	$c \leftarrow \{0,1\}$; $X \leftarrow (K_c^*, K_{1-c}^*)$
$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$	$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$
$C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$	$C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$
$(Rel, \langle C_i, \tau_i\rangle_{i=1}^n) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$	$(Rel, \langle C_i, \tau_i\rangle_{i=1}^n) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$
for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$	for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$
if $Rel(K_0^*, \langle K_i\rangle_{i=1}^n) = 1$, then return 1	if $Rel(K_1^*, \langle K_i\rangle_{i=1}^n) = 1$, then return 1
else return 0	else return 0

From the definitions of the wNM-ATK experiment and the CNM-ATK experiment, the CNM-ATK security implies the wNM-ATK security. Because, for any wNM-ATK adversary A, we can construct the adversary A' against the CNM-ATK security with the same advantage as the advantage of the adversary A. The adversary A' does the same thing of the adversary A except that the adversary A' ignores $X = (K_c^*, K_{1-c}^*)$. We have that the experiment $\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},A'}(k)$ is equal to the experiment $\mathsf{Expt}^{\text{wNM-ATK}}_{\mathsf{TKEM},A}(k, 0)$, and

$\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A'}(k)$ is equal to the experiment $\mathsf{Expt}^{\text{wNM-ATK}}_{\mathsf{TKEM},\,A}(k,1)$. As a conclusion, we have $\mathsf{Adv}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A'}(k) = \mathsf{Adv}^{\text{wNM-ATK}}_{\mathsf{TKEM},\,A}(k)$.

SNM-ATK Tag-KEM. We define *simulation non-malleability* (SNM) for tag-KEM by following in [8]. Roughly speaking, there exists a simulator S which output encapsulations without keys, where these encapsulations are the same as those the adversary with keys outputs.

Definition 3 (SNM-ATK tag-KEM). Let TKEM be a tag-KEM system. We define the advantage $\mathsf{Adv}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,A,\,S}(k,Rel) \stackrel{\text{def}}{=} |\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,A}(k,Rel) = 1] - \Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,S}(k,Rel) = 1]|$ for a legitimate and non-copying adversary A, a relation Rel, and a simulator S as follows. If, for every legitimate and non-copying adversary $A = (A_1, A_2, A_3)$ and relation Rel, there exists some simulator S such that $\mathsf{Adv}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,A,\,S}(k,Rel)$ is negligible in the security parameter k, then we say that TKEM satisfies the SNM-ATK security, where ATK $\in \{$CPA, CCA1, CCA1.5, CCA2$\}$.

$\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,A}(k,Rel)$:	$\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,S}(k,Rel)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$
$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$	$st_1 \leftarrow S_1^{O_1(\cdot)}(pk)$
$(w, K_0^*) \leftarrow \mathsf{TKEM.Key}(pk)$	
$K_1^* \leftarrow \mathcal{K}$; $c \leftarrow \{0,1\}$; $X \leftarrow (K_c^*, K_{1-c}^*)$	$K_0^*, K_1^* \leftarrow \mathcal{K}$; $c \leftarrow \{0,1\}$; $X \leftarrow (K_c^*, K_{1-c}^*)$
$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$	$st_2 \leftarrow S_2^{O_2(\cdot)}(st_1, X)$
$C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$	
$(\langle C_i, \tau_i\rangle_{i=1}^n, \sigma) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$	$(\langle C_i, \tau_i\rangle_{i=1}^n, \sigma) \leftarrow S_3^{O_3(\cdot)}(st_2)$
for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$	for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$
if $Rel(K_0^*, \langle K_i\rangle_{i=1}^n, \sigma) = 1$, then return 1	if $Rel(K_0^*, \langle K_i\rangle_{i=1}^n, \sigma) = 1$, then return 1
else return 0	else return 0

PNM-ATK Tag-KEM. We define *indistinguishability under parallel chosen ciphertext attack based on non-malleability* (PNM) for tag-KEM by following [8]. This definition is proposed in order to prove the equivalence between the IND-CCA2 and the NM-CCA2 security for PKE and KEM[2,3,8]. We also use the PNM-ATK security in order to prove the equivalence between the IND-CCA2 and the sNM-CCA2 security for tag-KEM.

Definition 4 (PNM-ATK tag-KEM). Let TKEM be a tag-KEM system. We define the advantage $\mathsf{Adv}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,A}(k) \stackrel{\text{def}}{=} |\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1] - \frac{1}{2}|$ for a legitimate and non-copying adversary A as follows. If for every legitimate and non-copying adversary $A = (A_1, A_2, A_3, A_4)$, $\mathsf{Adv}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,A}(k)$ is negligible in the security parameter k, then we say that TKEM satisfies the PNM-ATK security, where ATK $\in \{$CPA, CCA1, CCA1.5, CCA2$\}$.

$\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},A}(k)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$; $st_1 \leftarrow A_1^{\mathcal{O}_1(\cdot)}(pk)$
$(w, K_0^*) \leftarrow \mathsf{TKEM.Key}(pk)$; $K_1^* \leftarrow \mathcal{K}$; $b \leftarrow \{0,1\}$
$(st_2, \tau^*) \leftarrow A_2^{\mathcal{O}_2(\cdot)}(st_1, K_b^*)$; $C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$
$(st_3, \langle C_i, \tau_i \rangle_{i=1}^n) \leftarrow A_3^{\mathcal{O}_3(\cdot)}(st_2, C^*)$
for every i, $K_i \leftarrow \mathsf{TKEM.Dec}(sk, C_i, \tau_i)$
$b' \leftarrow A_4(st_3, \langle K_i \rangle_{i=1}^n)$
if $b' = b$, return 1
else return 0

We remark that the PNM-ATK security implies the IND-ATK security by the definitions of the experiments, since the IND-ATK experiment is equal to the PNM-ATK experiment except for the one-time parallel decryption operation between the third stage and the forth stage. Moreover, the inverse implication holds in the case of CCA2 since the adversary with access to the decryption oracle does not need the one-time parallel decryption operation.

4 Theorems

In this section, we present main theorems in this paper. We omit the other theorems and their detailed proofs. They can be found in the full version of this paper. In Section 5, we only present the proof sketches of the main theorems.

Theorem 5 (Equivalence among three definitions for sNM-ATK). Let TKEM be a tag-KEM system and ATK $\in$ {CPA, CCA1, CCA1.5, CCA2}. TKEM satisfies the CNM-ATK security, if and only if TKEM satisfies the SNM-ATK security. Similarly, TKEM satisfies the CNM-ATK security, if and only if TKEM satisfies the PNM-ATK security.

It has been already proved that the CNM-ATK, the SNM-ATK, and the PNM-ATK security are equivalent for any non-copying adversary in the case of PKE and KEM [2,3,8], but not yet proved in the case of tag-KEM. We prove the equivalence among these definitions.

Theorem 6 (sNM-{CPA, CCA1} tag-KEM + IND-OT DEM $\Rightarrow$ NM-{CPA,CCA1} PKE). If TKEM is an sNM-{CPA, CCA1} secure tag-KEM system and DEM is an IND-OT secure DEM system, then $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is NM-{CPA,CCA1} secure .

This theorem means that $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is NM-ATK secure, whenever TKEM is sNM-ATK secure and DEM is IND-OT secure. In contrast, it is proved by Herranz that this implication for the KEM/DEM framework does not hold [5].

Theorem 7 (wNM-CCA2 tag-KEM + IND-CCA2 DEM $\not\Rightarrow$ IND-CPA PKE). There exists TKEM which is a wNM-CCA2 secure tag-KEM system and DEM which is an IND-CCA2 secure DEM system, such that $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is not IND-CPA secure.

In the case of KEM, if a KEM system satisfies the wNM-{CPA, CCA1} security and a DEM system satisfies the IND-OT security, a PKE system from them satisfies the

IND-{CPA, CCA1} security [5]. The reason is that the wNM-{CPA, CCA1} security implies the IND-{CPA, CCA1} security. However, in the case of tag-KEM, even if a tag-KEM system satisfies the wNM-CCA2 security and a DEM system satisfies the IND-CCA2 security, a PKE system from them satisfies the IND-CPA security. The wNM security for tag-KEM is insufficient to construct hybrid encryption systems with the weakest security, i.e. the IND-CPA security.

5 Proof Sketches of Theorem 5, Theorem 6, and Theorem 7

In this section, we present the proof sketches of Theorem 5, Theorem 6, and Theorem 7. The detailed proofs can be found in the full version of this paper.

5.1 Proof Sketch of Theorem 5

The proof of this theorem is clearly proved by Lemma 8, Lemma 9 and Lemma 10. In order to prove, we modify the proofs of these lemmas in [8].

Lemma 8 (CNM-ATK ⇒ SNM-ATK). Let TKEM be a tag-KEM system. If TKEM satisfies the CNM-ATK security, then TKEM satisfies the SNM-ATK security.

Lemma 9 (SNM-ATK ⇒ PNM-ATK). Let TKEM be a tag-KEM system. If TKEM satisfies the SNM-ATK security, then TKEM satisfies the PNM-ATK security.

Lemma 10 (PNM-ATK ⇒ CNM-ATK). Let TKEM be a tag-KEM system. If TKEM satisfies the PNM-ATK security, then TKEM satisfies the CNM-ATK security.

First, we present the proof sketch of Lemma 8. We prove that TKEM is not secure in the sense of CNM-ATK if TKEM is not secure in the sense of SNM-ATK. Let $A = (A_1, A_2, A_3)$ be the adversary against the SNM-ATK security. Using the adversary A, we construct the adversary $B = (B_1, B_2, B_3)$ against the CNM-ATK security and the simulator $\tilde{S} = (\tilde{S}_1, \tilde{S}_2, \tilde{S}_3)$ in the SNM-ATK experiment as follows.

$B_1(pk)^{O_1(\cdot)}$:	$B_2^{O_2(\cdot)}(st_1, X)$:	$B_3^{O_3(\cdot)}(st_2, C^*)$:
$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$	$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$	$(\langle C_i, \tau_i \rangle_{i=1}^n, \sigma) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$
return st_1	return (st_2, τ^*)	define Rel',
		as bind σ to the last input of Rel
		return $(Rel', \langle C_i, \tau_i \rangle_{i=1}^n)$

$\tilde{S}_1(pk)^{O_1(\cdot)}$:	$\tilde{S}_2^{O_2(\cdot)}(st_1 \| pk, X)$:	$\tilde{S}_3^{O_3(\cdot)}(st_2 \| pk \| \tau^*)$:
$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$	$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$	$(K, w) \leftarrow \mathsf{TKEM.Key}(pk)$
return $st_1 \| pk$	return $(st_2 \| pk \| \tau^*, \tau^*)$	$C \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$
		$(\langle C_i, \tau_i \rangle_{i=1}^n, \sigma) \leftarrow A_3^{O_3(\cdot)}(st_2, C)$
		if $(C_i, \tau_i) \neq (C, \tau^*)$ for every i,
		then return $(\langle C_i, \tau_i \rangle_{i=1}^n, \sigma)$
		else abort

The adversary B in the experiment $\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},B}$ perfectly simulates the SNM-ATK experiment for A, when B received X which is a pair of real and random keys. The simulator $\tilde{S}$ in the experiment $\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\tilde{S}}$ can perfectly simulate the act of B in the experiment $\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},B}$. Hence, we have

$$\begin{aligned}\mathsf{Adv}^{\text{CNM-ATK}}_{\mathsf{TKEM},B}(k) &= |\Pr[\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},B}(k)=1] - \Pr[\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},B}(k)=1]| \\ &= |\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},A}(k)=1] - \Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\tilde{S}}(k)=1]| \\ &= \mathsf{Adv}^{\text{SNM-ATK}}_{\mathsf{TKEM},A,\tilde{S}}(k, Rel).\end{aligned}$$

Second, we present the proof sketch of Lemma 9. We prove that TKEM is not secure in the sense of SNM-ATK if TKEM is not secure in the sense of PNM-ATK. Let A be the adversary against the PNM-ATK security. Using the adversary $A = (A_1, A_2, A_3, A_4)$, we construct the adversary $B = (B_1, B_2, B_3)$ against the SNM-ATK security and a relation Rel as follows.

$B_1(pk)^{O_1(\cdot)}$:	$B_2^{O_2(\cdot)}(st_1, X)$:	$B_3^{O_3(\cdot)}(st_2 \| X, C^*)$:
$st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$	parse $X = (K, K')$	$(st_3, \langle C_i, \tau_i\rangle_{i=1}^n) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$
return st_1	$(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, K)$	$r \leftarrow \{0,1\}^k$
	return $(st_2 \| X, \tau^*)$	$\sigma' \leftarrow (st_3, r, X)$
		return $(\langle C_i, \tau_i\rangle_{i=1}^n, \sigma')$

$Rel(Y, \langle K_i\rangle_{i=1}^n, \sigma')$:
parse $\sigma' = (st_3, r, X)$, $X = (K, K')$
if $Y = K$, then $g = 0$
else if $Y = K'$, then $g = 1$
otherwise, return 0
$g' \leftarrow A_4(st_3, \langle K_i\rangle_{i=1}^n; r)$
if $g = g'$, then return 1
else return 0

We note that r chosen by B_3 is used to A_4's random input. We define the events Real that $Y = K$ in Rel and Random that $Y = K'$ in Rel. From the construction of B and Rel, we have

$$\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},B}(k, Rel) = 1 | \mathsf{Real}] = \Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},A}(k) = 1 | b = 0]$$

$$\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},B}(k, Rel) = 1 | \mathsf{Random}] = \Pr[\mathsf{Expt}^{PNM-ATK}_{\mathsf{TKEM},A}(k) = 1 | b = 1].$$

Hence, we have

$$\begin{aligned}&\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},B}(k, Rel) = 1] \\ &= \frac{1}{2}(\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},B}(k, Rel) = 1 | \mathsf{Real}] + \Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},B}(k, Rel) = 1 | \mathsf{Random}]) \\ &= \frac{1}{2}(\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},A}(k) = 1 | b = 0] + \Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},A}(k) = 1 | b = 1]) \\ &= \Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},A}(k) = 1].\end{aligned}$$

Next, let S be a simulator in the SNM-ATK security. We define the event Bad that $Y \neq K \wedge Y \neq K'$ in the SNM-ATK experiment for simulators. Then, for every simulator S, we have

$$\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,S}(k, Rel) = 1] = \Pr[g = g' \wedge \neg\mathsf{Bad}] \leq \frac{1}{2}.$$

Hence, we conclude

$$\begin{aligned}\mathsf{Adv}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,B,\,S}(k, Rel) &= |\Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,B}(k, Rel) = 1] - \Pr[\mathsf{Expt}^{\text{SNM-ATK}}_{\mathsf{TKEM},\,S}(k, Rel) = 1]| \\ &\geq |\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1] - \frac{1}{2}| = \mathsf{Adv}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,A}(k).\end{aligned}$$

Finally, we present the proof sketch of Lemma 10. We prove that TKEM is not secure in the sense of PNM-ATK if TKEM is not secure in the sense of CNM-ATK. Let $A = (A_1, A_2, A_3)$ be the adversary against the CNM-ATK security. Using the adversary A, we construct the adversary $B = (B_1, B_2, B_3, B_4)$ against the PNM-ATK security as follows.

$B_1(pk)^{O_1(\cdot)}$: $st_1 \leftarrow A_1^{O_1(\cdot)}(pk)$ return st_1	$B_2^{O_2(\cdot)}(st_1, X_0)$: $X_1 \leftarrow \mathcal{K}$; $c \leftarrow \{0,1\}$ $X \leftarrow (X_c, X_{1-c})$ $(st_2, \tau^*) \leftarrow A_2^{O_2(\cdot)}(st_1, X)$ return $(st_2 \| X_0, \tau^*)$
$B_3^{O_3(\cdot)}(st_2 \| X_0, C^*)$: $(Rel, \langle C_i, \tau_i\rangle_{i=1}^n) \leftarrow A_3^{O_3(\cdot)}(st_2, C^*)$ $st_3 \leftarrow (Rel, X_0)$ return $(st_3, \langle C_i, \tau_i\rangle_{i=1}^n)$	$B_4(st_3, \langle K_i\rangle_{i=1}^n)$: if $Rel(X_0, \langle K_i\rangle_{i=1}^n) = 1$, then $b' \leftarrow 0$ else $b' \leftarrow 1$ return b'

From the construction of B, we have

$$\begin{aligned}\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 1 | b = 0] &= \Pr[Rel(X_0, \langle K_i\rangle_{i=1}^n) = 1 | X_0 \text{ is a real key}] \\ &= \Pr[\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1],\end{aligned}$$

$$\begin{aligned}\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 0 | b = 1] &= \Pr[Rel(X_0, \langle K_i\rangle_{i=1}^n) = 1 | X_0 \text{ is a random key}] \\ &= \Pr[\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1].\end{aligned}$$

Therefore, we conclude

$$\begin{aligned}\mathsf{Adv}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) &= |\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 1] - \frac{1}{2}| \\ &= \frac{1}{2}|\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 1 | b = 0] + \Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 1 | b = 1] - 1| \\ &= \frac{1}{2}|\Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 1 | b = 0] - \Pr[\mathsf{Expt}^{\text{PNM-ATK}}_{\mathsf{TKEM},\,B}(k) = 0 | b = 1]| \\ &= |\Pr[\mathsf{Expt}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1] - \Pr[\widetilde{\mathsf{Expt}}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A}(k) = 1]| \\ &= \mathsf{Adv}^{\text{CNM-ATK}}_{\mathsf{TKEM},\,A}(k).\end{aligned}$$

5.2 Proof Sketch of Theorem 6

First, we only consider the case of CCA1. Let TKEM be a tag-KEM system satisfies the PNM-CCA1 security and DEM be a DEM system satisfies the IND-OT security. Then, $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ is a hybrid encryption system from them with the tag-KEM/DEM framework. We show $\mathsf{PKE}^{\mathsf{TKEM,DEM}}$ satisfies the NM-CCA1 security by modifying the NM-CCA1 experiment into the games, $\mathsf{Game}_0(b)$, $\mathsf{Game}_1(b)$, $\mathsf{Game}_2(b)$. The game $\mathsf{Game}_0(b)$ is equal to the experiment $\mathsf{Expt}^{\text{NM-CCA1}}_{\mathsf{PKE}^{\mathsf{TKEM,DEM}},A}(k,b)$.

$\mathsf{Game}_0(b)$:	$\mathsf{Game}_1(b)$:	$\mathsf{Game}_2(b)$:
$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$	$(pk, sk) \leftarrow \mathsf{TKEM.Gen}(1^k)$
$(st, \mathcal{M}) \leftarrow A_1^{O_1(\cdot)}(pk)$	$(st, \mathcal{M}) \leftarrow A_1^{O_1(\cdot)}(pk)$	$(st, \mathcal{M}) \leftarrow A_1^{O_1(\cdot)}(pk)$
$m_0^*, m_1^* \leftarrow \mathcal{M}$	$m_0^*, m_1^* \leftarrow \mathcal{M}$	$m_0^*, m_1^* \leftarrow \mathcal{M}$
$(w, K^*) \leftarrow \mathsf{TKEM.Key}(pk)$	$(w, K^*) \leftarrow \mathsf{TKEM.Key}(pk)$	$(w, K^*) \leftarrow \mathsf{TKEM.Key}(pk)$
	$\underline{K' \leftarrow \mathcal{K}}$	$K' \leftarrow \mathcal{K}$
$\tau^* \leftarrow \mathsf{DEM.Enc}(K^*, m_0^*)$	$\tau^* \leftarrow \mathsf{DEM.Enc}(\underline{K'}, m_0^*)$	$\tau_b^* \leftarrow \mathsf{DEM.Enc}(K', m_b^*)$
$C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$	$C^* \leftarrow \mathsf{TKEM.Enc}(w, \tau^*)$	$\underline{C_b^* \leftarrow \mathsf{TKEM.Enc}(w, \tau_b^*)}$
$\gamma^* \leftarrow (C^*, \tau^*)$	$\gamma^* \leftarrow (C^*, \tau^*)$	$\underline{\gamma_b^* \leftarrow (C_b^*, \tau_b^*)}$
$(Rel, \langle\gamma_i\rangle_{i=1}^n) \leftarrow A_2(st, \gamma^*)$	$(Rel, \langle\gamma_i\rangle_{i=1}^n) \leftarrow A_2(st, \gamma^*)$	$(Rel, \langle\gamma_i\rangle_{i=1}^n) \leftarrow A_2(st, \underline{\gamma_b^*})$
for every i,	for every i,	for every i,
$m_i \leftarrow \mathsf{PKE.Dec}(sk, \gamma_i)$	$m_i \leftarrow \mathsf{PKE.Dec}(sk, \gamma_i)$	$m_i \leftarrow \mathsf{PKE.Dec}(sk, \gamma_i)$
if $Rel(m_b^*, \langle m_i\rangle_{i=1}^n) = 1$,	if $Rel(m_b^*, \langle m_i\rangle_{i=1}^n) = 1$,	if $Rel(\underline{m_0^*}, \langle m_i\rangle_{i=1}^n) = 1$,
return 1	return 1	return $\underline{1}$
else return 0	else return 0	else return 0

We denote modified parts by underlines. On this modification, we state the following claims.

Claim 11. For every $b \in \{0,1\}$ and every adversary $B = (B_1, B_2, B_3, B_4)$,

$$|\Pr[\mathsf{Game}_0(b) = 1] - \Pr[\mathsf{Game}_1(b) = 1]| \leq \mathsf{Adv}^{\text{PNM-CCA1}}_{\mathsf{TKEM},B}(k).$$

Claim 12. For every adversary $C = (C_1, C_2)$,

$$|\Pr[\mathsf{Game}_2(0) = 1] - \Pr[\mathsf{Game}_2(1) = 1]| \leq \mathsf{Adv}^{\text{IND-OT}}_{\mathsf{DEM},C}(k).$$

We can prove Claim 11 and Claim 12 with reduction to the PNM-CCA1 security for TKEM and the IND-OT security for DEM, respectively. Since changes between Game_1 and Game_2 are purely conceptual, we have $\Pr[\mathsf{Game}_2(b) = 1] = \Pr[\mathsf{Game}_1(b) = 1]$ for every b. From Claim 11 and Claim 12, we have

$$\begin{aligned}
&\mathsf{Adv}^{\text{NM-CCA1}}_{\mathsf{PKE}^{\mathsf{TKEM,DEM}},A} \\
&= |\Pr[\mathsf{Expt}^{\text{NM-CCA1}}_{\mathsf{PKE}^{\mathsf{TKEM,DEM}},A}(k,0) = 1] - \Pr[\mathsf{Expt}^{\text{NM-CCA1}}_{\mathsf{PKE}^{\mathsf{TKEM,DEM}},A}(k,1) = 1]| \\
&= |\Pr[\mathsf{Game}_0(0) = 1] - \Pr[\mathsf{Game}_0(1) = 1]| \\
&\leq 2\mathsf{Adv}^{\text{PNM-CCA1}}_{\mathsf{TKEM},B}(k) + \mathsf{Adv}^{\text{IND-OT}}_{\mathsf{DEM},C}(k).
\end{aligned}$$

Therefore, we can prove Theorem 6 in the case of CCA1. Similarly, we can prove it in the case of CPA, since the only difference between CCA1 and CPA is whether A_1 has access to $O_A(\cdot)$ or not.

Incidentally, we can consider this proof to be another proof of the well-known theorem in [1] that $\mathsf{PKE}^{\mathsf{TKEM},\mathsf{DEM}}$ is IND-CCA2 secure, where TKEM is an IND-CCA2 secure tag-KEM system and DEM is an IND-OT secure DEM system.

5.3 Proof Sketch of Theorem 7

Assume there exists a wNM-CCA2 secure tag-KEM $\mathsf{TKEM} = (\mathsf{TKEM.Gen}, \mathsf{TKEM.Key}, \mathsf{TKEM.Enc}, \mathsf{TKEM.Dec})$ and an IND-CCA2 secure DEM $\mathsf{DEM} = (\mathsf{DEM.Enc}, \mathsf{DEM.Dec})$. We modify DEM and TKEM into a new DEM′ and TKEM′, respectively.

$\mathsf{DEM}'.\mathsf{Enc}(K, m)$:	$\mathsf{DEM}'.\mathsf{Dec}(K, \tau')$:
parse $K = K_1 \Vert K_2$	parse $K = K_1 \Vert K_2$ and $\tau' = \tau'_1 \Vert \tau'_2$
$\tau'_1 \leftarrow \mathsf{DEM.Enc}(K_1, m)$	if $K_2 = \tau'_2$, then $m' \leftarrow \mathsf{DEM.Dec}(K_1, \tau'_1)$
$\tau'_2 \leftarrow K_2$	else $m' \leftarrow \perp$.
return $\tau' = \tau'_1 \Vert \tau'_2$	return m'

Claim 13. DEM′ is secure in the sense of IND-CCA2.

The adversary A against the IND-CCA2 security of DEM can simulate the view of the adversary B against the IND-CCA2 security of DEM′. The adversary A only picks K_2 and checks that least significant bits of a ciphertext τ' is equal to K_2.

Next, we also modify TKEM to use DEM with key space $\{0,1\}^{2k}$ into a new $\mathsf{TKEM}' = (\mathsf{TKEM}'.\mathsf{Gen}, \mathsf{TKEM}'.\mathsf{Key}, \mathsf{TKEM}'.\mathsf{Enc}, \mathsf{TKEM}'.\mathsf{Dec})$ with the same key space, which still satisfies the wNM-CCA2 security. We set $\mathsf{TKEM}'.\mathsf{Gen} = \mathsf{TKEM.Gen}$ and the others as follows.

$\mathsf{TKEM}'.\mathsf{Key}(pk)$:	$\mathsf{TKEM}'.\mathsf{Enc}(w', \tau')$:
$(w, K) \leftarrow \mathsf{TKEM.Key}(pk)$	parse $w' = w \Vert K_1 \Vert K_2$ and $\tau' = \tau'_1 \Vert \tau'_2$
divide $K = K_1 \Vert K_2$	if $K_2 = \tau'_2$,
$w' \leftarrow w \Vert K_1 \Vert K_2$	then $C'_1 \leftarrow \mathsf{DEM.Dec}(K_1, \tau'_1)$
return $(w', K_1 \Vert K_2)$	and $C'_2 \leftarrow K_1 \Vert K_2$
	else $C'_1 \leftarrow \mathsf{TKEM.Enc}(w, \tau'_1)$
	and $C'_2 \leftarrow \perp$
	return $C'_1 \Vert C'_2$

$\mathsf{TKEM}'.\mathsf{Dec}(sk, C'_1 \Vert C'_2, \tau')$:
parse $\tau' = \tau'_1 \Vert \tau'_2$
if $C'_2 = \perp$, then return $\mathsf{TKEM.Dec}(sk, C'_1, \tau'_1)$
else
parse $C'_2 = K_1 \Vert K_2$
if $K_2 = \tau'_2$ and $C'_1 = \mathsf{DEM.Dec}(K_1, \tau'_1)$,
then return $K_1 \Vert K_2$
else return $\perp$

Claim 14. TKEM' is secure in the sense of wNM-CCA2.

The adversary A against the wNM-CCA2 security of TKEM can simulate the view of the adversary B against the wNM-CCA2 security of TKEM'. If $K_2 \neq \tau'_2$, then A can perfectly simulate the view of the adversary B against the wNM-CCA2 security of TKEM'. The probability that $K_2 = \tau'_2$ is negligible since K_2 is perfectly secret and uniformly random. Therefore, we can ignore this event.

Claim 15. $\mathsf{PKE}^{\mathsf{TKEM}',\mathsf{DEM}'}$ is not secure in the sense of IND-CPA.

We consider the adversary $A = (A_1, A_2)$ against the IND-CPA security of $\mathsf{PKE}^{\mathsf{TKEM}',\mathsf{DEM}'}$. In the first stage, A_1 outputs $(0^k, 1^k)$ as challenge messages. By the definitions of TKEM' and DEM', if 0^k is encrypted in the IND-CPA experiment, then A_2 receives $(0^k \| K_1^* \| K_2^*, \mathsf{DEM.Enc}(K_1^*, 0^k))$ as the challenge ciphertext. Similarly, if 1^k is encrypted in the IND-CPA experiment, then A_2 receives $(1^k \| K_1^* \| K_2^*, \mathsf{DEM.Enc}(K_1^*, 1^k))$. Since these ciphertexts are distinguishable, $\mathsf{PKE}^{\mathsf{TKEM}',\mathsf{DEM}'}$ is not secure in the sense of IND-CPA.

References

1. Abe, M., Gennaro, R., Kurosawa, K., Shoup, V.: Tag-KEM/DEM: A New Framework for Hybrid Encryption and A New Analysis of Kurosawa-Desmedt KEM. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 128–146. Springer, Heidelberg (2005)
2. Bellare, M., Desai, A., Pointcheval, D., Rogaway, P.: Relations Among Notions of Security for Public-Key Encryption Schemes. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 26–45. Springer, Heidelberg (1998)
3. Bellare, M., Sahai, A.: Non-malleable Encryption: Equivalence between Two Notions, and an Indistinguishability-Based Characterization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 519–536. Springer, Heidelberg (1999)
4. Cramer, R., Shoup, V.: Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM Journal on Computing 33 (2003)
5. Herranz, J., Hofheinz, D., Kiltz, E.: KEM/DEM: Necessary and Sufficient Conditions for Secure Hybrid Encryption (2006), http://eprint.iacr.org/2006/265
6. Herranz, J., Hofheinz, D., Kiltz, E.: The Kurosawa-Desmedt Key Encapsulation is not Chosen-Ciphertext Secure (2006), http://eprint.iacr.org/2006/207
7. Kurosawa, K., Desmedt, Y.: A New Paradigm of Hybrid Encryption Scheme. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 426–442. Springer, Heidelberg (2004)
8. Nagao, W., Manabe, Y., Okamoto, T.: On the Equivalence of Several Security Notions of Key Encapsulation Mechanism (2006), http://eprint.iacr.org/2006/268
9. Pass, R., Shelat, A., Vaikuntanathan, V.: Relations Among Notions of Non-malleability for Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 519–535. Springer, Heidelberg (2007)
10. Shoup, V.: A proposal for an ISO standard for public key encryption (version 2.1) (manuscript, 2001), http://www.shoup.net/papers/

A Highly Scalable RFID Authentication Protocol

Jiang Wu and Douglas R. Stinson*

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1, Canada
{j32wu,dstinson}@uwaterloo.ca

Abstract. In previous RFID protocols, a hash-chain is used to achieve good privacy. Each tag is associated with a chain of Q hash values. To identify one tag out of a total of N tags, a server searches a table of size NQ. A naive search takes either $\Theta(NQ)$ time or $\Theta(NQ)$ memory, and therefore it does not scale well. A time-space tradeoff technique can mitigate the scalability problem. However, with the time-memory tradeoff, either time or space is still at least $\Theta((NQ)^{2/3})$.

In this paper, we propose a novel RFID protocol to solve the scalability problem. The server "solves", instead of "searches", for a tag ID. The protocol is based on polynomial operations, and its security and privacy is based on the difficulty of reconstructing a polynomial with noisy data. The protocol supports very large values of the product NQ. In our demo implementation where $N = 2^{32}$ and $Q = 13700$, the server takes 0.1 seconds and 10K bytes memory to identify a tag. As a comparison, a hash-chain based protocol enhanced with a time-memory tradeoff will require about 67 seconds and a 1G bytes memory.

1 Introduction

Radio Frequency Identification (RFID) is an automated object identification technology. RFID systems consist of two main components: tags and readers. Tags are small radio transponders. They contain the identification information of objects to which they are attached. Readers query these tags for the identifying information about the objects. Readers often have secure access to a back-end database. For simplicity, a reader and a back-end database can be treated as a single entity.

While being promising in a wide range of applications such as supply chain, anti-counterfeiting, and libraries, RFID also raises privacy and security concerns. Since RFID tags respond to radio interrogation automatically, malicious scanning of tags is a plausible threat. Even if the information emitted by a tag is encrypted, the information may be used to track the tag, thus causing privacy issues. An equally significant problem is authentication. One purpose of RFID

* Research supported by NSERC discovery grant 203114-06.

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 360–376, 2009.

tags is to prove the authenticity of objects. If an RFID tag can be scanned and replicated, then a counterfeit tag can be made to impersonate the authentic one.

RFID protocols must provide privacy and security under those possible attacks. A common attack model is as follows. There is an adversary who is able to eavesdrop the communications between the tags and readers, interrogate the tags, and compromise some tags. The goal of the adversary is to impersonate or track uncompromised tags. Correspondingly, an RFID protocol needs to meet two requirements: be secure against impersonation and be untraceable.

In addition to the security and privacy requirements, an RFID protocol needs to be scalable on the reader side and efficient on the tag side. An RFID system may have a large number, say, millions to billions, of tags. The RFID protocol must be scalable to allow the reader to deal with such a large number of tags. On the other hand, a tag has very limited resources in computation and memory, so the protocol must be efficient on the tag side. For cryptographic tools, symmetric key techniques are usually considered feasible for some RFID tags while public key techniques are considered unsuitable for any of them.

To design an RFID protocol satisfying all the desired privacy, security, scalability, and efficiency properties is challenging. Privacy makes RFID authentication different from conventional cryptographic authentication. Using symmetric key techniques, secure authentication relies on a symmetric key shared between a tag and reader. For privacy, a tag cannot identify itself to a reader before an authentication interaction, thus the reader does not know which key to use in the interaction. A straightforward solution is to try every key. This is prohibitively costly when the number of tags becomes large. This is known as the *key search problem*. Literature in this area has sought to reduce the cost of key search. Every such protocol proposed so far involves some kind of tradeoff among the desired properties [13].

Our Contributions. In this paper, we propose a novel RFID protocol to solve the key search problem. The server "solves", instead of "searches", for a tag ID in an identification process. The operation is based on polynomial computations and its security and privacy is based on the difficulty of reconstructing polynomials with noisy data. The protocol supports a very large number of tags as well as a very large number of queries on one tag, which is impractical for previous protocols. At the same time, the protocol is secure and privacy-preserving.

Organization. The remainder of the paper is organized as follows. In Section 2, we review RFID protocols addressing the scalability issue. In Section 3, we present our scheme. Section 4 gives the security analysis. Section 5 gives the performance analysis, and Section 6 concludes the paper.

2 Related Work

In this section, we review previous approaches and solutions to the key search problem in the literature. In [13], Juels gave a comprehensive survey of RFID

security and privacy, including the key search problem. Here we briefly review the existing solutions.

Tree approach. In [16], Molnar and Wagner proposed a scheme to reduce the key search cost to $\Theta(\log n)$ where n is the total number of keys. The scheme uses d sets of keys $K_1, \ldots, K_d$. Each set contains b keys. Each tag is assigned with d keys $k_1 \in K_1, \ldots, k_d \in K_d$. The key assignment can be represented as a tree of depth d and each node has d children. The scheme can accommodate up to b^d tags in total. In an identification session, the tag runs d rounds of interaction with the reader. In the i^{th} round, the tag uses the key k_i, and the reader searches K_i. In a session, the reader needs to search through db keys.

Since a key will be used in more than one tag, compromise of one tag results in compromise of keys in other tags; hence this leads to privacy infringements [2].

Synchronization approach. The basic idea in the synchronization approach is for the reader and tags to maintain a synchronized state. For example, every tag T_i maintains a counter c_i. On interrogation, the tag outputs $E = f_{k_i}(c_i)$ where f is a keyed hash function and k_i is a secret key shared between the reader and T_i, then increases c_i. The reader can compute all possible outputs of all tags and store the results in a searchable table. In each interrogation, the reader searches the response from the tag in the table.

There are several variants of the above approach in the literature, e.g., Ohkubo, Suzuki, and Kinoshita [19], Henrici and Müller [11], Juels [12], and Dimitriou [8].

Time-space tradeoff approach. In [3] and [2], Avoine, Dysli, and Oechslin used a time-space trade-off to achieve $\Theta(n^{\frac{2}{3}})$ in both memory and time complexity for key search. Time-space tradeoff is used as an enhancement to the synchronization approach. The basic idea is to organize all future response values of all tags in chains. Each chain consists of a sequence of response values as follows. The reader chooses a random function f to map a response value x to a pair (i, j), then the response of tag i at the j^{th} query will be the successor of x in the chain. Only the head and the tail of a chain is stored. When receiving a response x, the reader can apply f to get its successors. The reader searches x and its successors in all the tails of the chains to locate the chain which contains x. Then it starts from the head of the chain, applies f repeatedly, and locates (i, j) of x.

3 Scheme Description

In this section, we present our RFID protocol. We assume that there are N tags and one reader (here the reader refers to an entity consisting of the actual reader, the database that stores the tag information, and perhaps an application server which processes the data). We also assume that a tag is capable of generating random bits, computing cryptographic hash functions, and modular multiplication over a field F_{2^l}. Such tags would fall into the category of symmetric-key tags in [13].

3.1 Initial Setup

The initial setup configures the reader and the tags as follows.

- Choose an l-bit block cipher and a secret key. Let E be the encryption function and D be the decryption function.
- Generate m random bivariate polynomials $f_1(x, y), \ldots, f_m(x, y)$ over a finite field F_{2^l}. The degree of x and y in each f_i is at most k.
- Assign the m polynomials to the reader.
- For tag $i, 1 \leq i \leq N$, compute $x_i = E(i)$ as its meta ID, then
  - store m univariate polynomials $f_{1,i}(y) = f_1(x_i, y), \ldots, f_{m,i}(y) = f_m(x_i, y)$ in the tag in a randomly permuted order.
  - Set a counter c to 0.
  - Set an interrogation threshold Q_{max}, which is the maximum number of queries that the tag will answer correctly.
  - Set a number b, which will be used in the protocol.
- Choose a secure hash function $h : \{0,1\}^* \mapsto \{0,1\}^l$ which will be used by readers and tags.

Note that the meta ID x_i is not stored in the tag or the reader. We will discuss the parameter selection for l, m, b, k, and Q_{max} in the analysis and performance sections.

3.2 Authentication

The authentication process between a reader and a tag i is as follows.

1. The reader generates $r \in F_{2^l}$ uniformly at random, then sends r to the tag.
2. If $c > Q_{max}$, then the tag responds with b random points in $F_{2^l} \times F_{2^l}$. Otherwise, the tag does the following:
   (a) Generate $r' \in F_{2^l}$ uniformly at random, and compute $y' = h(r||r')$.
   (b) Choose g from $\{f_{1,i}, \ldots, f_{m,i}\}$ randomly, and compute $z' = g(y')$.
   (c) Generate $b-1$ pairs of points $(r_1, z_1) \ldots, (r_{b-1}, z_{b-1}) \in F_{2^l} \times F_{2^l}$ uniformly at random.
   (d) Send the b points $(r', z'), (r_1, z_1), \ldots, (r_{b-1}, z_{b-1})$ to the reader in a random order.
   (e) Set $c = c + 1$.
3. After receiving the b points, for each point (r', z') and each polynomial $f \in \{f_1, \ldots, f_m\}$, the reader solves the equation $z' = f(x, h(r||r'))$. The reader needs to solve mb such equations, and each equation generates up to k solutions. If any solution x is a valid meta ID, i.e., $1 \leq D(x) \leq N$, then the reader identifies the tag ID as $D(x)$.

Remark. In Step 2 (b), to pick a polynomial, the tag can use two different approaches. One is *random choice* and the other is *random permutation*. In random choice, each time, the tag chooses one polynomial from the m polynomials uniformly at random. In random permutation, the choice of g in m consecutive authentication sessions is a random permutation of the m polynomials assigned to the tag. In the following parts, we assume the random permutation approach.

Correctness. If the reader receives b points from a valid tag with ID i, then one of the points (r', z') will satisfy $z' = f(E(i), h(r||r'))$ for a polynomial $f \in \{f_1, \ldots, f_m\}$. At the reader side, $E(i)$ will be the solution of x to the equation $z' = f_j(x, h(r||r'))$. So the valid tag will be correctly identified.

The server will also get up to $mbk - 1$ other solutions in one authentication session. The distribution of the meta IDs can be considered to be uniformly at random in F_{2^l}, so the probability that one or more of these solutions happen to be valid meta IDs is estimated to be

$$1 - \left(1 - \frac{N}{2^l}\right)^{mbk-1}. \tag{1}$$

4 Security Analysis

In this section, we analyze the security and privacy properties of our protocol. Recall that in the attack model, the adversary can eavesdrop and query any tag, and it can compromise some tags. The goal of the adversary is to impersonate or trace the uncompromised tags.

4.1 Query and Recovery

First, we consider the query-and-recovery attack. In this attack, the adversary repeatedly queries a tag, collects the responses, and then tries to recover the polynomials assigned to the tag. This is an intermediate step toward impersonation and tracing.

Our security analysis for the query-and-recover attack is based on the hardness of the *noisy polynomial interpolation problem*, which is related to the well-known *polynomial reconstruction problem*. Next we present existing results about these problems in the literature, followed by our analysis.

Preliminary. First we review the polynomial reconstruction (PR) problem [15] and the noisy polynomial interpolation (NPI) problem [18].

Definition 1. *(Polynomial reconstruction)*
Input: Integers k and t, and T points $\{(x_i, y_i) : i \in [1, T]\}$ where $x_i, y_i \in F_{2^l}$.
Output: All univariate polynomials P of degree at most k such that $P(x_i) = y_i$ for at least t values $i \in [1, T]$.

The best known algorithm to solve this problem is the Guruswami-Sudan algorithm [10]. It solves the problem when $t > \sqrt{Tk}$ in time polynomial in T. When $t \leq \sqrt{Tk}$, the current state of knowledge suggests that the problem may be difficult even in the light of recent extensions of list or average case decoding for related families of codes [15].

A variant of the PR problem, denoted as the noisy polynomial interpolation (NPI) problem, is as follows:

Definition 2. *(Noisy Polynomial Interpolation)*

Input: S sets of points generated as follows: Pick a random polynomial P over F_{2^l} of degree at most k. Generate $S \geq k+1$ sets, each containing B points. The x coordinate of each point is randomly chosen from F_{2^l} subject to the condition that all x values are distinct and different from 0. In each set there is exactly one point (x, y) which satisfies $y = P(x)$. For the other $B-1$ points, the y coordinate is chosen randomly.

Output: the polynomial P.

The NPI problem is presented in [18]. A previous version of this problem was presented in [17] in which the points in the same set all have the same x coordinate. In [5], Bleichenbacher and Nguyen show that having the same x coordinate allows meet-in-the-middle attacks and lattice attacks. However, it is unknown how to employ these attacks against NPI. It appears that, although there is more information given in NPI than in the PR problem, NPI may be as hard as the PR problem.

Parameter Selection for NPI. In [18], Noar and Pinkas also proposed a cryptographic assumption based on the NPI problem. They assumed that, if S, B and k are polynomial in the security parameter and $S < \sqrt{SBk}$, then the problem is hard. Here we further analyze the security level under concrete parameter settings.

First we review the analysis of parameter choices for the PR problem in [14]. In [14], the best approach to solve the PR problem is assumed to be one of the two choices: (1) choose $k+1$ points to recover a polynomial and then test all $\binom{T}{k+1}$ polynomials, or (2) delete d points, where $t > \sqrt{(T-d)k}$, and use the GS algorithm on the remaining $T-d$ points. We call the first approach *exhaustive search*; its idea is clear. The idea of the second approach is to delete d points. If all the d points are not on the polynomial P, then it happens that $t > \sqrt{(T-d)k}$ so that GS will output the proper P. We call this approach *exhaustive deletion*.

We note that the idea of exhaustive search and exhaustive deletion can be generalized to selecting n points where $k+1 \leq n \leq T$. We use this generalized idea for the NPI problem and consider the following new probabilistic algorithm A as shown in Algorithm 1 to solve NPI.

Algorithm 1. A

choose n sets of points
choose β points from each set
use the $n\beta$ points as input to the GS algorithm, and output all polynomials of degree at most k that fit at least $k+1$ points.

Note that A may output more than one polynomial. If P is among them, we consider A successful in solving the problem.

Let $P_{NPI}(S, B, k, n, \beta)$ be the probability that P will be outputted. Let t be the number of points in the set of $n\beta$ points that satisfy $y = P(x)$. It is clear

that $0 \leq t \leq n$ and t follows a binomial distribution with parameters $(\frac{\beta}{B}, n)$. If $t > \sqrt{n\beta k}$, then P will be outputted. It holds that

$$\begin{aligned} P_{NPI}(S, B, k, n, \beta) &= \Pr[t > \sqrt{n\beta k}] \\ &= \sum_{i=\lfloor\sqrt{n\beta k}\rfloor+1}^{n} \Pr[t = i] \\ &= \sum_{i=\lfloor\sqrt{n\beta k}\rfloor+1}^{n} \binom{n}{i} \left(\frac{\beta}{B}\right)^{i} \left(1 - \frac{\beta}{B}\right)^{n-i}. \end{aligned} \tag{2}$$

Remark. Note that, when $\beta = 1$ and $n = k + 1$, the algorithm becomes an exhaustive search and

$$P_{NPI}(S, B, k, k+1, 1) = \left(\frac{1}{B}\right)^{k+1}.$$

When $\beta = B$, the algorithm is deterministic for given n. In this situation, if $n \leq Bk$, then $\lfloor\sqrt{nBk}\rfloor + 1 > n$ so that $P_{NPI}(S, B, k, n, B) = 0$. If $n > Bk$, then

$$P_{NPI}(S, B, k, n, B) = \sum_{i=\lfloor\sqrt{nBk}\rfloor+1}^{n} \binom{n}{i} 0^{n-i} = 1.$$

Regarding the hardness of NPI, we make the following assumption to estimate the concrete security level.

Assumption 1. *Let*

$$Adv_{S,B,k}^{NPI} = \max\{P_{NPI}(S, B, k, n, \beta) : 1 \leq \beta \leq B, k+1 \leq n \leq S\}. \tag{3}$$

Let n_0, β_0 be the optimal choices of n, β for A to achieve $Adv_{S,B,k}^{NPI}$. Let τ be the running time of the A using n_0, β_0. We assume that no algorithm can solve NPI with probability more than $Adv_{S,B,k}^{NPI}$ using time at most τ.

$Adv_{S,B,k}^{NPI}$ can be estimated as

$$Adv_{S,B,k}^{NPI} = \max\left\{\left(\frac{1}{B}\right)^{k+1}, P_{NPI}\left(S, B, k, S, \left\lceil\frac{S}{k}\right\rceil - 1\right)\right\} \tag{4}$$

(See Appendix A for a detailed analysis).

Security Under Query-and-Recovery. Now we relate the difficulty of the query-and-recover attack to the difficulty of solving the NPI problem.

Recall that in the protocol, to answer a challenge r, the tag computes $z' = g(h(r||r'))$ where r' is a random number generated by the tag. $h(r||r')$ can be considered as random to the adversary. This point (z', r') is sent along with other $b-1$ random points. In a query-and-recovery attack, the adversary queries a tag Q times. In every consecutive m queries, the tag uses each of its m polynomials once and in a random order. The problem for the adversary is to recover these polynomials after Q queries. We have the following result.

Theorem 2. *Suppose Assumption 1 holds. If the adversary queries a tag Q times (where we assume that $m|Q$), then the probability that it can recover any polynomial of the tag within time τ is at most*

$$Adv^{NPI}_{Q/m,mb-m+1,k}. \tag{5}$$

Proof. We reduce an NPI problem with parameters $(S = Q/m, B = mb-m+1)$ to a query-and-recover problem. We choose $m-1$ random polynomials. For each polynomial, we generate Q/m points from Q/m random x values, and put the Q/m points in the Q/m sets. This becomes a query-and-recover problem of Q queries, m polynomials of degree k, and b points in each answer.

Given ϵ, m, b, and k, there is a Q_{max} such that for $Q \leq Q_{max}$, it holds that

$$Adv^{NPI}_{Q/m,mb-m+1,k} \leq \epsilon.$$

Therefore, Q_{max} is the maximum number of queries allowed for one tag. Q_{max} can be estimated as

$$Q_{max} \approx ((b-1)m^2 + m)k \tag{6}$$

(See Appendix B for a detailed analysis).

Equation (6) indicates that Q_{max} increases linearly with m^2 and k. mk indicates the memory overhead to store the m polynomials of degree k in a tag. Q_{max} also increases linearly with b, which indicates the communication overhead and the number of random points the tag needs to generate.

4.2 Compromise and Recovery

In this attack, the adversary compromises some tags, obtains the polynomials assigned to these tags, and tries to recover the bivariate polynomials used in the server side. This is an intermediate step toward impersonation and tracing.

First we express the polynomial assignment in the form of a matrix computation. Let

$$X = \begin{pmatrix} 1 & x_1 & \dots & {x_1}^k \\ & \ddots & & \\ & & \ddots & \\ 1 & x_n & \dots & {x_n}^k \end{pmatrix},$$

where $x_1, \dots, x_n$ are the meta IDs assigned to the n tags. Let $M_1, \dots, M_m$ be $(k+1) \times (k+1)$ matrices, which are matrix representations of the bivariate polynomials $f_1, \dots, f_m$. Let $Y_i = XM_i$. Then Y_i is an $n \times (k+1)$ matrix, and row j of M_i corresponds to the univariate polynomial generated using f_i and assigned to tag j.

We use $M[r]$ to denote the r^{th} row of a matrix M, and $M[r][c]$ to denote the entry at the r^{th} row and c^{th} column of M. It is clear that $Y_j[i] = X[i]M_j$, and the univariate polynomials assigned to tag i are $Y_1[i], \dots, Y_m[i]$.

For the adversary to recover an M_i, it is necessary to know Y_i. Suppose the adversary obtains Y_i. He then needs to solve the following system of $n(k+1)$ equations with $n+(k+1)^2$ unknown variables:

$$Y_i[r][c] = \sum_{j=0}^{k} x_r{}^j M_i[j][c], \tag{7}$$

where $0 \le r \le n-1, 0 \le c \le k$.

A necessary (but maybe not sufficient) condition to solve (7) is $n \ge \frac{(k+1)^2}{k}$. We assume that when $n \ge \frac{(k+1)^2}{k}$, the adversary can solve (7) in a practical time period τ.[1]

However, when $Y_1[i], \ldots, Y_m[i]$ are assigned to the tag x_i, their order is randomly permuted. So by compromising n tags, the adversary does not know which n polynomials in the n tags are generated from the same M. We assume that the best he can do is to draw one polynomial from each tag as a row of Y_i, and solve $x_1, \ldots, x_n$ and M_i in (7). With probability $1/m^{n-1}$, the polynomials in Y_i are generated from the same bivariate polynomial f. Therefore, we estimate that the probability that the adversary can compute one M_i in time τ is

$$\begin{aligned} Adv_{CR} &= \frac{1}{m^{n-1}} \\ &\le \frac{1}{m^{\left\lceil \frac{(k+1)^2}{k} \right\rceil - 1}} \\ &= m^{-k-2}. \end{aligned} \tag{8}$$

Remark. After the adversary has compromised n tags, if he also knows the meta IDs $x_1, \ldots, x_n$ of the tags, then he can use the GS algorithm to recover $M_1, \ldots, M_m$. However, when $x_1, \ldots, x_n$ are unknown, the GS algorithm is no longer applicable, and we assume that there is no efficient algorithm to solve the problem. It is worthwhile to further investigate the validity of this assumption.

4.3 Impersonation

In the impersonation attack, the adversary tries to impersonate a uncompromised tag to a reader. In the RFID literature, this is often named counterfeiting or cloning. Since cloning implies replicating a tag, we think that impersonation may describe the goal of the adversary better. For example, the adversary may

[1] (7) is a system of nonlinear polynomial equations. For general systems of nonlinear polynomial equations where number of unknown variables u equals the number of equations v, there is no efficient algorithm. If the system is overdefined, i.e., $v > u$, then the linearization technique may sometimes be used to solve the problem efficiently [7]. But for (7), the linearization technique does not work. On the other hand, we cannot rule out the possibility that the special form of (7) may make some efficient algorithms possible. Here we assume that (7) can be solved efficiently. This assumption may underestimate the security of the protocol.

combine the information gathered from a tag to generate new messages to impersonate a tag, or use the information gather from one tag to impersonate another tag. In both cases, the adversary tries to cheat the reader in a way other than by replicating.

The impersonation attack model for RFID should be much weaker than that for conventional cryptographic entity authentication. For example, concurrent attacks (where the adversary concurrently executes multiple sessions with the prover (tag)) and intruder-in-the-middle attacks (where the adversary intercepts and manipulate the messages between the prover (tag) and the verifier (reader)) do not seem to apply to RFID. These attacks originated in the Internet communication model and have not been considered for RFID in the literature. The main concern for RFID is that, in a authentication session, the adversary might be able to use the information collected before to make the reader accept it as a valid and uncompromised tag. This is similar to the smartcard communication model. We analyze the protocol in this scenario.

After receiving a challenge r, to make the reader accept it as a valid tag, the adversary A needs to generate a point (r', z') such that $z' = g(h(r||r'))$ where g is a polynomial assigned to the tag. As we have analyzed, the adversary cannot learn the polynomials assigned to a tag by querying a tag or compromising other tags. Since A does not know g, he cannot compute z' by evaluating $g()$. Then A may choose a z' previously generated by the tag. A can do this in a probabilistic way: after observing a query and response from the tag, A randomly picks one point from the response. With probability $1/b$, A gets r_0, r'_0, and z'_0 where $z'_0 = f(h(r_0||r'_0))$. Then A needs to find r' such that $h(r||r') = h(r_0||r'_0)$. When r is a random challenge, and the hash function $h()$ is modelled as a random oracle, then the probability that A chooses r such that $h(r||r') = h(r_0||r'_0)$ is

$$\begin{aligned} & \Pr[h(r \parallel r') = h(r_0 \parallel r'_0)] \\ &= \Pr[h(r \parallel r') = h(r_0 \parallel r'_0) \wedge r = r_0] + \Pr[h(r \parallel r') = h(r_0 \parallel r'_0) \wedge r \neq r_0] \\ &\leq \Pr[r = r_0] + \Pr[h(r \parallel r') = h(r_0 \parallel r'_0) | r \neq r_0] \\ &= \frac{1}{2^l} + \frac{1}{2^l} \\ &= \frac{1}{2^{l-1}}. \end{aligned} \tag{9}$$

A may just send random points and hope that at least one of them satisfies $z' = f(x', h(r||r'))$ for a polynomial $f \in \{f_1, \ldots, f_m\}$ and a valid meta ID x'. The probability that this happens is similar to (1).

4.4 Tracing

Now we consider the tracing problem. Suppose that the adversary observes or participates in two authentication sessions as a reader. The problem for the adversary is to tell if the two sessions involve the same tag.

The output of a tag is b points. $b - 1$ of them are random points, and only one point $(r', g(h(r||r')))$ contains the information related to the tag. As we have

analyzed, the adversary cannot learn the polynomials of an uncompromised tag. Without knowing g, only when $y_1 = h(r||r'_1) = y_2 = h(r||r'_2)$, the adversary can tell that two points $(y_1, g(y_1))$ and $(y_2, g(y_2))$ are generated by the same polynomial. When r'_1 and r'_2 are generated by the tag at random, and the hash function $h()$ is modelled as a random oracle, the probability that the adversary can trace a tag by its response is

$$\begin{aligned} \Pr[h(r||r'_1) = h(r||r'_2)] &\leq \Pr[r'_1 = r'_2] + \Pr[h(r||r'_1) = h(r||r'_2) | r'_1 \neq r'_2] \quad (10) \\ &= \frac{1}{2^l} + \frac{1}{2^l} \\ &= \frac{1}{2^{l-1}}. \end{aligned}$$

Note that if a tag has been queried more than Q_{max} times, and the adversary can get more information than the tag's response (i.e., if an authentic reader accepts the tag), then the adversary may trace the tag.

5 Performance

We discuss the performance of the protocol under a concrete parameter setting. We set $l = 64, m = 16, k = 8, b = 8$, and $Q_{max} = 13700$.

5.1 Security and Privacy

The security and privacy provided by the protocol is as follows.

- The adversary queries a tag and tries to recover the secret polynomials held by the tag. In each trial, the probability that the adversary can succeed is at most 2^{-60} (by (5)).
- The adversary compromises several tags and tries to determine the secret bivariate polynomials held by the server. In each trial, the probability that the adversary can succeed is at most 2^{-40} (by (8)).
  Note that, with probability 2^{-40}, the adversary only obtains a correct system of nonlinear equations. The system consists of $(k+1)^2 + k + 3$ unknowns and $(k+1)^2 + k + 3$ equations. It is not clear whether there is any efficient algorithm to solve the system.
- The adversary tries to compute a valid response using messages previously generated by a tag. In each trial, the probability that the adversary can succeed is 2^{-63} (by (9)).
- The adversary answers a query with random responses. In each interaction with the reader, the probability that the adversary is identified as a valid tag is at most 2^{-22} when $N \leq 2^{32}$ (by (1)).
- The adversary tries to determine if two responses are from the same tag. For each pair of responses from one tag, the probability that the adversary succeeds is at most 2^{-63} (by 10).

– The adversary launches a Denial of Service (DOS) attack against a tag by repeatedly querying a tag. After being queried for 13700 times, a tag cannot be accepted by an authentic reader. In addition, an adversary may be able to trace the tag using information in addition to the tag's response, e.g., if the tag is accepted by an authentic reader.

5.2 Tag

In each session, a tag needs to generate $2b-1$ random numbers in F_{2^l}, evaluate a polynomial of degree k over F_{2^l}, and compute a hash function. Random number generation and hash computation are considered suitable for symmetric-key tags and have been used in most previous RFID protocols. The tag needs to store m polynomials over F_{2^l}, each of degree k. So it needs to store $m(k+1)$ items of size l. For $m = 16, k = 8, l = 64$, it takes 9216 bit ROM, corresponding to 9216 gates in hardware. Using Horner's rule, it takes k modular multiplication over F_{2^l} to evaluate a polynomial of degree k. A 64 bit modular multiplier takes several hundred gates in hardware. Therefore, in our protocol, a tag needs about 10000 more gates in hardware than a regular tag capable of hashing computation. As a comparison, if pubic key techniques are used to solve the scalability problem while preserving privacy, then about 20000 gates are required to implement an ECC processor for RFID tags to perform public key computations [9].

5.3 Server

The server stores m bivariate polynomials of degree k. This requires $m(k+1)^2 l/8 \approx 10$K bytes memory.

In each session, for each bivariate polynomial $f(x, y)$ and each received point (z', r'), the reader needs to solve an equation $z' = f(x, h(r||r'))$. Then it checks if the roots are valid meta IDs. There are efficient algorithms to solve polynomials over finite fileds, e.g., Berlekamp's algorithm [4] and the Cantor-Zassenhaus algorithm [6]. After a meta ID is computed, it takes constant time to check if it is valid. Solving the polynomials dominates the total time in an authentication process.

We implemented the reader algorithm based on NTL [1]. On a P4 3.2G PC with 1G RAM running Linux, when $m = 16, k = 8$, $b = 8$, and $l = 64$, the running time using the Cantor-Zassenhaus algorithm in average case (solving $mb/2$ polynomials of degree k on F_{2^l} where $l = 64$) is about 0.1 seconds for one query.

5.4 Scalability

N (the maximum number of tags) is at most l bits. N is also limited by the false positive probability given in (1). Given the fixed parameters $m = 16, k = 8$, $b = 8$, and $l = 64$, when the false positive probability is less than 2^{-22}, N can be as large as 2^{32}, which is sufficient for almost all conceivable applications. Therefore the protocol is highly scalable.

5.5 Comparison

We compare our protocol with OSK/AO [3]. Both protocols are secure and untraceable, and both are designed to solve the key search problem to provide good scalability.

Time and Space. OSK/AO has a query threshold Q_m, which is the length of its hash chain. OSK/AO needs to store a precomputed table of size M. The time for each query follows the rule

$$T = \mu \frac{(NQ_m)^2}{M^2}$$

where μ is a constant. For $N = 2^{20}$, $Q_m = 128$, and $M = 1$GByte, OSK/AO takes 0.004 milliseconds for one query. However, when the number of tags increases, without increasing the table size, the time increases fast. Let M, Q_m be fixed, and let T_1, T_2 be the time for tag number N_1, N_2 respectively. It holds that

$$T_2 = \left(\frac{N_2}{N_1}\right)^2 T_1.$$

If we consider a system of $N = 2^{32}$ tags, then, OSK/AO will take 67 seconds for one query, using an 1G bytes precomputed table. As a comparison, our protocol takes 0.1 seconds and 10K bytes memory in the server.

Query Threshold. In OSK/AO, if a tag is queried more than $Q_m = 128$ times by an adversary between two queries by an authentic server, then the tag cannot be recognized by an authentic reader. In our protocol, if a tag is queried a total $Q_m = 13700$ times, then it cannot be recognized by an authentic reader. In OSK/AO, the time for one query increases in ${Q_m}^3$ (instead of ${Q_m}^2$). In our protocol, the time increases in $\sqrt{Q_m}$. Therefore, the query threshold Q_m in our protocol is much higher. However, OSK/AO does not have a limit on the total number of queries for a tag, although it is more susceptible to tag disable attacks where an adversary repeatedly queries a tag in a short time. On the contrary, our scheme is more resilient to tag disable attacks, but there is a limit on the total number of times that a tag can be queried. Which approach is better depends on how frequently a tag is queried. For example, in a library RFID system where a tag may be queried several times a day, the OSK/AO scheme may be desirable. In some other applications such as e-passport where a tag is queried every several weeks or months, our protocol may be preferable.

Tag Hardware. The main drawback of our protocol compared to OSK/AO may be the hardware cost on the tag side, which is about 10000 more gates in hardware.

6 Conclusion

In this paper, we proposed a novel RFID protocol to solve the key search problem in RFID identification protocols. In previous RFID protocols, a hash-chain

is used to achieve good privacy. In such protocols, to identify a tag, a sever needs to search a table of size NQ, where N is the number of tags and Q is the length of the hash chain. The search takes either $\Theta(NQ)$ time or $\Theta(NQ)$ memory, and therefore it does not scale well. A time-memory tradeoff technique can mitigate the scalability problem. However, with the time-memory tradeoff, either the time or the space is still at least $\Theta((NQ)^{2/3})$. In our protocol, the server "solves", instead of "searches", for a tag ID. The protocol is based on polynomial operation, and its security and privacy is based on the difficulty of reconstructing a polynomial with noisy data. The protocol supports very large NQ values. In our demo implementation where $N = 2^{32}, Q = 13700$, the server takes 0.1 seconds and 10K bytes memory to identify a tag. As a comparison, a hash-chain protocol enhanced with time-memory tradeoff will take 67 seconds with the support of a 1G bytes pre-computed table. At the same time, our protocol preserves security and privacy.

References

1. NTL: A library for doing number theory, http://www.shoup.net/ntl/
2. Avoine, G., Dysli, E., Oechslin, P.: Reducing time complexity in RFID systems. In: Preneel, B., Tavares, S.E. (eds.) SAC 2005. LNCS, vol. 3897, pp. 291–306. Springer, Heidelberg (2006)
3. Avoine, G., Oechslin, P.: A scalable and provably secure hash-based RFID protocol. In: PerCom Workshops, pp. 110–114. IEEE Computer Society, Los Alamitos (2005)
4. Berlekamp, E.R.: Factoring polynomials over finite fields. Bell Systems Technical Journal (46), 1853–1859 (1967)
5. Bleichenbacher, D., Nguyên, P.Q.: Noisy Polynomial Interpolation and Noisy Chinese Remaindering. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 53–69. Springer, Heidelberg (2000)
6. Cantor, D.G., Zassenhaus, H.: A new algorithm for factoring polynomials over finite fields. Math. Comp. 36(154), 587–592 (1981)
7. Courtois, N., Klimov, A., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations (2000)
8. Dimitriou, T.: A lightweight RFID protocol to protect against traceability and cloning attacks. In: SECURECOMM 2005: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, Washington, DC, USA, pp. 59–66. IEEE Computer Society, Los Alamitos (2005)
9. Fürbass, F., Wolkerstorfer, J.: ECC processor with low die size for RFID applications. In: ISCAS, pp. 1835–1838. IEEE Computer Society Press, Los Alamitos (2007)
10. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Transactions on Information Theory 45(6), 1757–1767 (1999)
11. Henrici, D., Müller, P.: Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers. In: PerCom Workshops, pp. 149–153. IEEE Computer Society Press, Los Alamitos (2004)
12. Juels, A.: Minimalist cryptography for low-cost RFID Tags. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 149–164. Springer, Heidelberg (2005)

13. Juels, A.: RFID security and privacy: a research survey. IEEE Journal on Selected Areas in Communications 24(2), 381–394 (2006)
14. Kkiayias, A., Yung, M.: Directions in polynomial reconstruction based cryptography. IEICE Transactions on Fundamentals of Electronics, Communications and Computer E87-A(5), 978–985 (2004)
15. Kkiayias, A., Yung, M.: Cryptographic hardness based on the decoding of Reed-Solomon codes. IEEE Transactions on Information Theory 54(6) (2008)
16. Molnar, D., Wagner, D.: Privacy and security in library RFID: issues, practices, and architectures. In: CCS 2004: Proceedings of the 11th ACM conference on Computer and communications security, pp. 210–219. ACM Press, New York (2004)
17. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: STOC 1999: Proceedings of the thirty-first annual ACM symposium on Theory of computing, pp. 245–254. ACM Press, New York (1999)
18. Naor, M., Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. 35(5), 1254–1281 (2006)
19. Ohkubo, M., Suzuki, K., Kinoshita, S.: Efficient hash-chain based RFID privacy protection scheme. In: International Conference on Ubiquitous Computing Ubicomp, Workshop Privacy: Current Status and Future Directions (2004)

Appendix

A Discussion on NPI and Exhaustive Search and Deletion

We discuss the optimal choices of n and β for A to achieve $Adv^{NPI}_{S,B,k}$. It is difficult to give an analytic result, so here we discuss this question based on some simulations.

Figure 1 shows $P_{NPI}(S, B, k, n, \beta)$ as a function of n and β for $1 \leq n \leq Bk + 1, 1 \leq \beta \leq B$, $B = 5$ and $k = 5$. We see that when n approaches $Bk + 1$ and β approaches B, P_{NPI} increases sharply to 1. That indicates that if there are enough sets of points ($s \geq Bk+1$), then the adversary can easily recover the polynomial.

Figure 2 shows $P_{NPI}(S, B, k, n, \beta)$ as a function of n and β for $1 \leq n \leq S, 1 \leq \beta \leq B$, $B = 5$, $k = 5$ and $S = 83 \leq Bk$. There are two points that have peak values among their vicinities in the graph. One is at $n = k + 2$ and $\beta = 1$, the other is at $n = 83 = S$ and $\beta = 13$.

We note that, if we use the exhaustive search approach, we will choose $n = k + 1$ and $\beta = 1$, and then we have $P_{NPI}(S, B, k, k + 1, 1) = 0.1 \times 10^{-7}$ which is lower than the peak value where $P_{NPI}(S, B, k, k + 2, 1) = 0.8 \times 10^{-7}$, but the difference is small in view of their magnitudes. Also, the point $(n = k+1, \beta = 1)$ for exhaustive search is close to that for the peak value at $(n = k + 2, b = 1)$.

If we take the exhaustive deletion approach, then we will choose $n = S = 83, \beta = \lceil \frac{S}{k} \rceil - 1 = 16$, and have $P_{NPI}(S, B, k, 83, 16) = 0.4 \times 10^{-8}$ which is lower than the other peak value $P_{NPI}(S, B, k, 83, 15) = 0.4 \times 10^{-7}$, but difference is small in view of their magnitudes. Also the point for exhaustive search $(n = 83, \beta = 16)$ is close to that for the peak value at $(n = 83, \beta = 15)$.

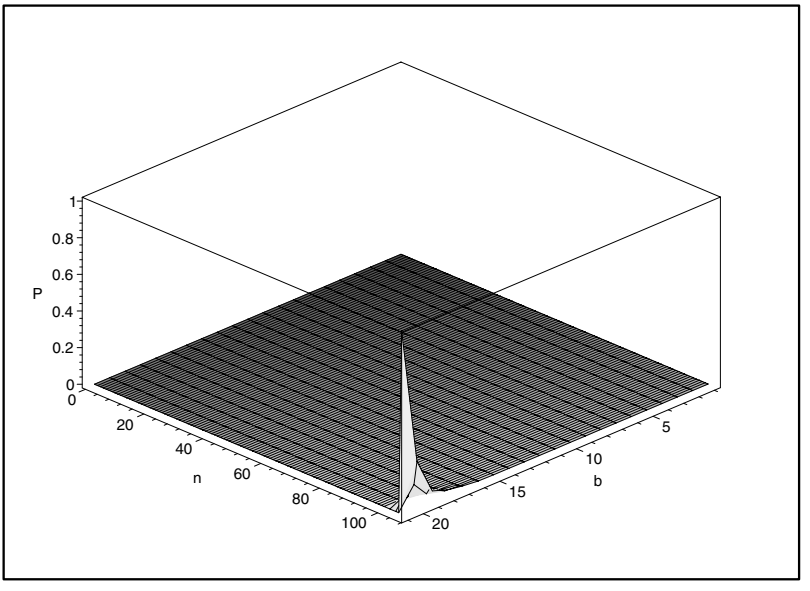


Fig. 1. P_{NPI} when $S \geq Bk + 1$

.

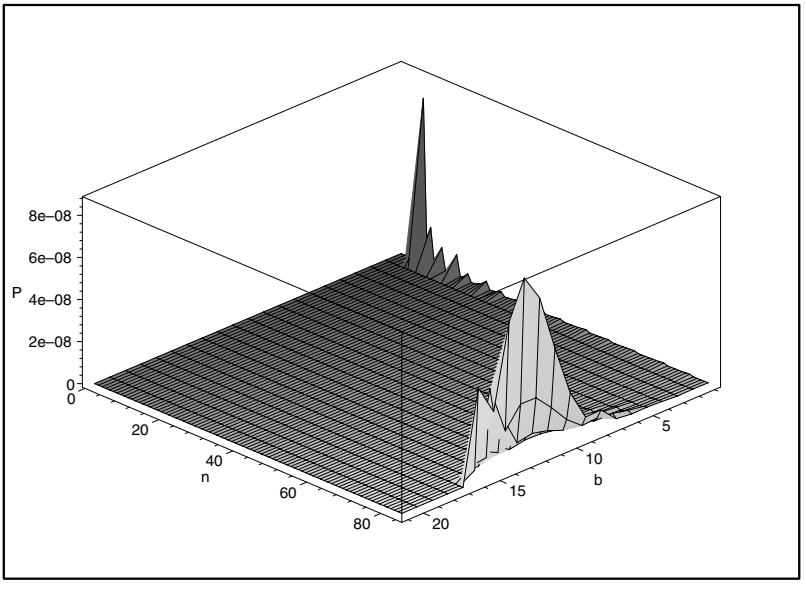


Fig. 2. P_{NPI} when $S < Bk$

.

The above observation leads to the conjecture that, for $S < Bk+1$, the results of exhaustive search and exhaustive deletion are close to the best strategy for the adversary. Then the advantage of the adversary may be estimated as

$$Adv^{NPI}_{S,B,k} = \max\left\{\left(\frac{1}{B}\right)^{k+1}, P_{NPI}\left(S, B, k, S, \left\lceil\frac{S}{k}\right\rceil - 1\right)\right\}.$$

B Discussion on Interrogation Threshold

For the protocol to be secure, we require that the advantage of the adversary is no more than a given ϵ. It is not clear how to directly compute Q_{max} (or equivalently $n_{max} = Q_{max}/m$) for given ϵ. We give the following estimate. In Figure 1, we see that $P(S, B, k, n, k) = 1$ when $n = Bk + 1$, and $P(S, B, k, n, k)$ decreases sharply to 0 when n decreases. This implies that n_{max} is less than Bk and may be close to Bk. Correspondingly, in the query-and-recovery attack, Q_{max} is close to $((b-1)m^2 + m)k$. We compare Q_{max} for given ϵ and different sets of m, b, k values and $((b-1)m^2 + m)k$ in Table 1.

Table 1. Estimation of Q_{max}

m	Q_{max}	$((b-1)m^2+m)k$	$\frac{Q_{max}}{((b-1)m^2+m)k}$
5	420	525	0.80
10	1860	2050	0.90
15	4290	4575	0.94
20	7720	8100	0.95
25	12150	12625	0.96
30	17580	18150	0.97
35	24010	24675	0.97
40	31440	32200	0.98
45	39870	40725	0.98
50	49300	50250	0.98

The results show that Q_{max} and $((b-1)m^2 + m)k$ are close. Therefore, we can estimate that

$$Q_{max} \approx ((b-1)m^2 + m)k.$$

Strengthening the Security of Distributed Oblivious Transfer

K.Y. Cheong, Takeshi Koshiba, and Shohei Nishiyama

Division of Mathematics, Electronics and Informatics,
Graduate School of Science and Engineering, Saitama University
255 Shimo-Okubo, Sakura, Saitama 338-8570, Japan
{kaiyuen,koshiba,shohei}@tcs.ics.saitama-u.ac.jp

Abstract. We study the distributed oblivious transfer first proposed by Naor and Pinkas in ASIACRYPT 2000, and generalized by Blundo et al. originally in SAC 2002 and Nikov et al. in INDOCRYPT 2002. One major objective of distributed oblivious transfer is to achieve information theoretic security under specified conditions through the distribution of the functions of traditional oblivious transfer to a set of neutral parties. In this paper we revise the definition of distributed oblivious transfer in order to deal with stronger adversaries and clarify possible ambiguities. Under the new definition, we observe some impossibility results and derive the upper bounds for the system parameters (with respect to the size of coalition). The weak points of previously proposed schemes based on threshold secret sharing schemes using polynomial interpolation are reviewed and resolved. We generalize the results and prove that, by adjusting some technical details, a previous scheme proposed by Nikov et al. is unconditionally secure. This protocol is efficient and achieves the parameter bounds at the same time.

Keywords: oblivious transfer, secret sharing scheme, information theoretic security.

1 Introduction

Oblivious Transfer (OT) is a two-party cryptographic protocol. In the first OT system introduced by Rabin [12], a message is received with probability 1/2 and the sender does not know whether his message reaches the other side. Later, Even et al. defined the 1-out-of-2 OT [6], where the sender has two secrets ω_0 and ω_1 and the receiver can choose one and only one of them in an oblivious manner. That is, the sender cannot know the receiver's choice $\sigma \in \{0, 1\}$ and the receiver cannot know any information on $\omega_{1-\sigma}$ as he gets ω_σ. The more general 1-out-of-N OT (where the sender has N secrets), the more specific 1-out-of-2 *bit* OT (where the secrets are one bit long), are similarly defined and the reductions among them have been discussed [2,3,5]. Also, Rabin's OT and the 1-out-of-2 OT are equivalent, as shown by Crépeau in [4]. In this paper we focus on the 1-out-of-2 OT. OT protocols are important building blocks of modern cryptography. Most notably, any secure multi-party computation can be based on OT [8,9,14].

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 377–388, 2009.

By simple arguments it can be shown that OT can only be computationally secure for either the sender or receiver, without involving extra parties or special communication channels. If information-theoretic security is desired for both sides, some more structures are needed. Distributed OT (DOT), proposed by Naor and Pinkas [10], achieves this. In their DOT, the sender interacts with M servers to distribute the secrets. After that, the receiver communicates with any m servers to get one secret. The scheme makes use of properties of polynomials as in Shamir's classic secret sharing scheme [13], which gives the fundamental framework of most other DOT schemes proposed later. For example, Blundo et al. [1] extended the work of [10] to the case of 1-out-of-N OT. Also in [1], the first DOT scheme not using polynomials was proposed. Taking the idea of [10], another DOT scheme has been suggested by Nikov et al. in [11], using only a different polynomial. In their work, the 1-out-of-2 OT can also be extended to 1-out-of-N OT. Moreover, they gave a general access structure protocol implementing the 1-out-of-N OT. Also, some general lower bounds for computational resources and system parameters (with respect to the size of coalition) in DOT have been derived in [1] and [11] respectively.

Recently, Ghodosi [7] revealed some weak points of the DOT scheme in [10] and showed some possible attacks. As the protocol in [10] uses a restricted form of Shamir's threshold secret sharing scheme based on polynomial interpolation, Ghodosi pointed out that such restriction causes the weak points of the DOT scheme even though the original secret sharing scheme is perfectly secure. These attacks may apply to similar schemes in [1] too. The protocol in [11] is mostly immune to the attacks but still not perfectly secure.

In this paper, we work on polynomial-based DOT, but the essential properties of the general DOT are first discussed in Section 2. We try to define DOT in simpler terms and to cope with stronger adversaries. Under the new definition, in Section 3 we derive some upper bounds for system parameters measuring the strength of adversaries, partially similar to the results in [11]. The framework by [10] for all DOT schemes based on polynomials is then introduced in Section 4.1. Problems related to Ghodosi's attacks are discussed in details in Section 4.2. Based on the results of [11], we then construct a protocol in Section 5.1. We use it to develop a scheme with provable security in information theoretic sense.

2 Defining DOT

In this paper, we try to define DOT generally. The desired properties of a DOT system are listed, though some of them may be optional. As in traditional OT, a 1-out-of-2 DOT protocol involves a sender with secrets (ω_0, ω_1) and a receiver with a choice $\sigma \in \{0, 1\}$. In addition, in DOT there is a set of m neutral servers. The sender interacts with the servers to dedicate the task of OT to them. The receiver follows the protocol to get ω_σ from the servers. In our definition of DOT, the interactions between all parties have the following properties:

1. Each single party has a secure and private communication channel with any other party. This is the basic assumption even if some channels are not used.

2. The sender and the receiver do not communicate with each other.
3. If all parties are honest, the interactions between the sender and the servers are finished before the interactions between the receiver and the servers begin. In practice, the servers may have to notify the receiver when it is possible for him to start his part of the protocol.

With the properties of DOT described above, when the sender has finished his part of protocol, it is possible for the receiver to halt the protocol for some time before he starts it again, if he ever does. The presence of the sender is not required during that time. This gives an advantage in various situations, for example, if it is uncertain when the receiver will request the OT, or if the sender is not supposed to know (or predict) the identity of the receiver. Next, the DOT protocol must guarantee the following functional properties:

1. Correctness: If all parties are honest, when the protocol is finished the receiver can always compute ω_σ.
2. Receiver's privacy: The (malicious) sender cannot get any information about σ of an honest receiver, even if he is in coalition with λ_1 corrupted servers.
3. Sender's privacy against receiver: For any coalition of the receiver and λ_2 servers, if they obtain any information about one secret of the honest sender, then they get no information about the other secret.
4. Sender's privacy against servers: A coalition of λ_3 corrupted servers gains no information about any of the honest sender's secrets.

Our DOT has three system parameters as shown above. The larger they are, the more secure the protocol is. If $\lambda_1 = \lambda_2 = \lambda_3 = 0$ then this DOT is a trivial OT achieved through a trusted third party. When giving the parameters, we assume a corrupted party is corrupted the whole time the protocol is run. In our definition of DOT, the security requirements are made as simple as possible. This is not only for simplicity but also for the largest freedom of the adversaries. We have three parameters due to three possible types of malicious parties, each going after a different objective:

1. A party not including the receiver may try to obtain information on σ.
2. A party including the receiver but not the sender may try to obtain information on both ω_0 and ω_1, while they are entitled to only one of them.
3. A party including neither the sender nor the receiver may try to obtain information on ω_0, ω_1, or both.

By simple arguments it is known that traditional OT is automatically secure against eavesdropping. Therefore the last type of adversary above is related to DOT only.

Note that our definition of DOT is slightly different from the previous schemes [1,10,11]. First, we do not assume the existence of a measure to limit receiver's access to the servers, so the receiver will contact all servers rather than a subset of them. Next, in all previous DOT schemes, the definition of receiver's privacy is against the servers only. The sender is assumed to be honest and not discussed. In our study the sender may actively cheat.

Also, in the DOT definitions in [10] and [11], the sender can only send one message to each server. After that, only one round (two messages) of communication is allowed between the receiver and each server. This one-round DOT assumption suffices for polynomial-based schemes, but is not true for a general DOT. For example, in [1] a two-round DOT was proposed. In this paper we do not limit the number of rounds of communication between any parties, such that a larger family of possible DOT protocols will be included. We believe our definition of DOT is simpler than others as a result. It also copes with stronger adversaries due to larger freedom on the possible malicious behaviors.

3 The Upper Bounds

We now show a simple upper bound for the system parameters λ_1 and λ_2 for a fixed m. We show that $m > \lambda_1 + \lambda_2$. For contradiction, consider $m \leq \lambda_1 + \lambda_2$. As a party can always corrupt fewer servers than allowed, it suffices to show that even the case $m = \lambda_1 + \lambda_2$ leads to insecurity of the DOT.

We now proceed to the proof of insecurity. Assume in one case, due to the absence of neutral servers, the sender agree to simulate by himself the first λ_1 servers and the receiver the remaining λ_2 servers, in order to achieve the function of a traditional OT. If both sides are fully honest, this clearly will give the desired result in terms of correctness. In traditional OT, information-theoretic security cannot be achieved for both sides, even for the semi-honest model. Therefore, one side (Alice) is at most only protected computationally. Replace the other side (Bob) with a semi-honest but computationally unlimited machine, it is clear that Bob will violate Alice's privacy without being detected. Next, replace all servers with neutral ones. But the malicious Bob can corrupt the servers which have been simulated by him earlier. He can then act semi-honest with the corrupted servers. Alice and the honest servers will not detect anything wrong while security is actually broken.

As a result, DOT can only be achieved if $m > \lambda_1 + \lambda_2$. This is the same as the bound shown in [11], but in [11] it is only shown for one-round DOT. Actually this is true in general. Note that this bound is tight in two ways. First, DOT can be achieved with $m = \lambda_1 + \lambda_2 + 1$. The actual protocols are discussed in this paper. Also, our new definition of DOT is closely related to this new bound, as DOT schemes overcoming this bound in some ways have been constructed in [1] by assuming that the sender is honest.

On the other hand, for λ_3, it is clear that $m > \lambda_3$ because all OT functions are dedicated to the m servers when the sender finishes his part of protocol and goes offline. A coalition of m servers can get all of the sender's secrets. In this paper we show that DOT can be achieved for $m = \lambda_3 + 1$.

4 DOT Based on Polynomials

4.1 The Original Scheme

First proposed by [10], a DOT scheme based on polynomials has the following structure:

1. Sender starts with two secret numbers (ω_0, ω_1) smaller than security parameter p and generates in finite field $\mathbb{F}_p$ (modulus p) a random polynomial $Q(x, y)$ such that $Q(0, 0) = \omega_0$ and $Q(0, 1) = \omega_1$.
2. There are m servers, named by integer 1 to m. To each server i the sender sends $Q(i, y)$ as a polynomial in y.
3. The receiver generates $S(x)$ such that $S(0) = \sigma$, his choice of secret. He sends $S(i)$ to each server i.
4. Define $R(x) = Q(x, S(x))$. Server i calculates $R(i) = Q(i, S(i))$ and sends it to the receiver.
5. The receiver now has m points of $R(x)$ to interpolate the polynomial. Then he calculates $R(0) = Q(0, S(0)) = \omega_\sigma$.

There are a number of different ways of choosing $Q(x, y)$ and $S(x)$ that differentiate the few schemes proposed previously. For correctness, we have to make sure $R(x)$ is of degree $m - 1$ or lower. Apparently, for receiver's privacy $S(x)$ should be of degree λ_1 or above. The structure of the polynomials has to be specified but all coefficients should be randomly selected to provide the best security.

In [10], two actual schemes were proposed. In both schemes, $S(x)$ is random with degree d_s. The only difference is $Q(x, y)$. In the first scheme, we have:

$$Q(x, y) = \sum_{j=1}^{m-1} a_j x^j + b_1 y + b_0. \tag{1}$$

In [10] it is known that this choice of $Q(x, y)$ is insecure when one server is corrupted by the receiver. This is because the receiver can run the DOT honestly to obtain $\omega_0 = b_0$ and then the corrupted server can get $\omega_1 = b_1 + b_0$ because b_1 is known by all servers. To avoid this problem, the second scheme in [10] is proposed. The choice of $Q(x, y)$ is a full polynomial:

$$Q(x, y) = \sum_{l=0}^{d_y} \sum_{j=0}^{d_x} a_{j,l} x^j y^l \tag{2}$$

with $d_x + d_y d_s = m - 1$. However, in [7], this scheme is also shown to have some security problems.

4.2 Security Concerns

First of all, as shown in the first attack in [7], if $S(x)$ is restricted to be of degree $d_s = \lambda_1$, a problem for receiver's privacy will occur. A coalition of d_s servers has a small advantage of guessing $\sigma = S(0)$. With probability $\frac{1}{p-1}$ the value of σ is exposed with certainty. Therefore, with this attack, the overall probability of guessing σ is $\frac{1}{2} + \frac{1}{2p-2}$ rather than the neutral $\frac{1}{2}$, if σ is uniformly distributed in the first place.

Actually, we can generalize this observation to show that, with a coalition of even fewer corrupted servers, a slight advantage for guessing $S(0)$ always exists.

Using the same method in [7], we define:

$$S(x) = \sum_{j=0}^{d_s} s_j x^j$$
$$S'(x) = \sum_{j=0}^{d_c} s'_j x^j \tag{3}$$

where $S(x)$ is the secret polynomial and $S'(x)$ is constructed by the d_c+1 points of $S(x)$ obtained by the cheating servers such that $S(x) = S'(x)$ at these points. If $d_s = d_c$ then $S(x)$ can be fully determined. Otherwise $d_c < d_s$ and the equation $S(x) - S'(x) = 0$ is expressed as

$$(s_0 - s'_0) + (s_1 - s'_1)x + \ldots + (s_{d_c} - s'_{d_c})x^{d_c} + s_{d_c+1}x^{d_c+1} + \ldots + s_{d_s}x^{d_s} = 0. \tag{4}$$

For the sake of convenience we define $s'_j = 0$ for $d_s \geq j > d_c$. Now, as a hypothesis concerning the unknown $S(x)$, consider the possible case that $(s_0, s_1, \ldots s_{d_s-d_c-1}) = (s'_0, s'_1, \ldots s'_{d_s-d_c-1})$. Then the equation of (4) may be reduced to

$$(s_{d_s-d_c} - s'_{d_s-d_c})x^{d_s-d_c} + (s_{d_s-d_c+1} - s'_{d_s-d_c+1})x^{d_s-d_c+1} + \ldots + (s_{d_s} - s'_{d_s})x^{d_s} = 0. \tag{5}$$

Denote by β_i the identity of the ith cheating server, this gives

$$\begin{pmatrix} \beta_1^{d_s-d_c} & \beta_1^{d_s-d_c+1} & \ldots & \beta_1^{d_s} \\ \beta_2^{d_s-d_c} & \beta_2^{d_s-d_c+1} & \ldots & \beta_2^{d_s} \\ \vdots & \ddots & & \vdots \\ \beta_{d_c+1}^{d_s-d_c} & \beta_{d_c+1}^{d_s-d_c+1} & \ldots & \beta_{d_c+1}^{d_s} \end{pmatrix} \begin{pmatrix} s_{d_s-d_c} - s'_{d_s-d_c} \\ s_{d_s-d_c+1} - s'_{d_s-d_c+1} \\ \vdots \\ s_{d_s} - s'_{d_s} \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \tag{6}$$

leading to the conclusion that $s_{d_s} = 0$ by solving this system of equations with nonzero Vandermonde determinant. This is a contradiction because the degree of $S(x)$ is d_s. Therefore the hypothesis is refuted and we know for certain that $(s_0, s_1, \ldots s_{d_s-d_c-1}) \neq (s'_0, s'_1, \ldots s'_{d_s-d_c-1})$. This eliminates one possibility on the value of $(s_0, s_1, \ldots s_{d_s-d_c-1})$. As $\sigma = s_0$, if it happens that s'_0 equals 0 or 1, then we can conclude that $s'_0 = 1 - \sigma$ is slightly more likely than $s'_0 = \sigma$. For the case where $d_c = d_s - 1$ as shown in [7], we can simply know that $s'_0 = 1 - \sigma$. Therefore, the receiver's privacy is not perfectly protected. Furthermore, even in the case of a coalition of d_s servers, the coalition can simulate a coalition of fewer servers. Combining the results, the adversary can actually guess σ with a probability larger than $\frac{1}{2} + \frac{1}{2p-2}$ in this new attack.

Next, in the second scheme of [10] as shown in (2) above, the degree of $R(x) = Q(x, S(x))$ is dependent on $S(x)$. This is a security problem for the honest sender if the receiver is cheating. As illustrated in the second attack in [7], the minimum degree of $R(x)$ is d_x when $S(x)$ has zero degree. The receiver can use $S(x) = 0$ to communicate with the first $d_x + 1$ servers to obtain ω_0, and use $S(x) = 1$ to interact with the next $d_x + 1$ servers to get ω_1. This attack works when

$m \geq 2d_x + 2$, even if no server is corrupted. If λ_2 servers are corrupted by the receiver the attack can work for $m \geq 2d_x + 2 - \lambda_2$. Also, this attack seems to apply to one of the schemes in [1] having the same type of structure.

4.3 Ideas on Security Fix

The first problem is for receiver's security. To resolve it we simply allow more freedom on the choice of $S(x)$. We can set $S(x)$ to be random with degree no larger than λ_1. The actual degree is only known by the receiver. This is enough to ensure correctness as the degree of $R(x)$ will not be increased. The only restriction is $S(0) = \sigma$. That is:

$$S(x) = \sum_{j=1}^{\lambda_1} s_j x^j + \sigma \tag{7}$$

where each s_j is uniformly distributed in $\mathbb{F}_p$. In this case the value of $S(0)$ is uniformly distributed, thus perfectly secret, in the view of any party knowing fewer than $\lambda_1 + 1$ other points of $S(x)$.

The second security issue threatens sender's privacy. One solution is to ensure that the degree of $R(x)$ is not dependent on $S(x)$. This can be done by the following general construction. The structure of $Q(x, y)$ is slightly modified to

$$Q(x, y) = \sum_{j,l \in \mathbb{N}} a_{j,l} x^j y^l \tag{8}$$

where j, l are nonnegative integers and $a_{j,l}$ is random if $j + l\lambda_1 \leq m - 1$, otherwise $a_{j,l} = 0$. This is a more general form of $Q(x, y)$ compared with (2) and the number of coefficients $a_{j,l}$ is increased. With this $Q(x, y)$, it is clear that the degree of $R(x)$ does not exceed $m - 1$ as long as the degree of $S(x)$ does not exceed λ_1. Also, the degree of $R(x)$ is independent of $S(x)$ such that the attack in [7] against the sender no longer works. This general structure is provably secure. In the next section we provide a specific scheme for it.

5 The Secure DOT Scheme

5.1 A Sub-protocol

In this paper we introduce a DOT protocol and prove its security in information theoretic sense, leaving no doubt on how secure it is. Also, $m = \lambda_1 + \lambda_2 + 1 = \lambda_3 + 1$, reaching the upper bounds of the parameters. First, by choosing a special case of the general case in (8), we give the following much simplified $Q(x, y)$ by setting $a_{j,l} = 0$ when $l > 1$. Practically this limits the degree of y in $Q(x, y)$ to one, and it becomes

$$Q(x, y) = B_0(x) + B_1(x)y \tag{9}$$

where $B_0(x)$ has degree at most $m - 1$ and $B_1(x)$ is of degree at most λ_2. These polynomials are the same as those in the scheme of [11] except the specifications

on their degrees. Only $B_0(0)$ and $B_1(0)$ are relevant to sender's secrets and all other coefficients are random in $\mathbb{F}_p$.

Next, we use this structure as a sub-protocol, as it is still not fully secure for the sender, because the receiver can actually get a linear combination of the sender's secrets. In particular, if $S(0) = \alpha$ then the receiver can get $B_0(0) + \alpha B_1(0)$ for any value of α, without being detected that α may not be zero or one. In the full DOT scheme this will be handled. But for the sub-protocol we allow this to happen, and the receiver will get $B_0(0) + \alpha B_1(0)$ for any α of his choice but nothing more. To summarize we have the following structure:

1. The sender begins with secret numbers a_0 and b_0, the receiver begins with secret α, all in $\mathbb{F}_p$. The receiver is allowed to obtain $a_0 + \alpha b_0$ while the sender should get nothing.
2. The sender generates randomly $(a_1, \ldots a_{m-1})$ and $(b_1, \ldots b_{\lambda_2})$ all in $\mathbb{F}_p$ and constructs:

$$\begin{aligned} B_0(x) &= \sum_{j=1}^{m-1} a_j x^j + a_0 \\ B_1(x) &= \sum_{j=1}^{\lambda_2} b_j x^j + b_0 \\ Q(x, y) &= B_0(x) + B_1(x) y \end{aligned} \tag{10}$$

3. The receiver generates randomly $(s_1, \ldots s_{\lambda_1})$ in $\mathbb{F}_p$ to construct:

$$S(x) = \sum_{j=1}^{\lambda_1} s_j x^j + \alpha \tag{11}$$

4. There are m servers named 1 to m. The sender sends each server i the values of $B_0(i)$ and $B_1(i)$. The receiver sends each server i the value of $S(i)$.
5. Let $R(x) = Q(x, S(x))$. Each server i calculates $R(i) = B_0(i) + B_1(i)S(i)$ and sends it to receiver.
6. The receiver interpolates m points of $R(x)$ to solve it fully. After that the value of $R(0) = a_0 + b_0\alpha$ can be calculated.

5.2 Correctness and Security

If the protocol above is followed, correctness is straightforward as the degree of $R(x)$ is at most $m - 1$. Next, it is clear that the receiver's privacy is ensured. As $S(x)$ is a random polynomial of degree no larger than λ_1, the view for $\alpha = S(0)$ is uniformly distributed given λ_1 or fewer other values of $S(x)$.

For sender's privacy against receiver, we show that the receiver can obtain at most a linear combination of the sender's two secret numbers, fulfilling the objective of the sub-protocol. A receiver corrupting server i can do no better than obtaining $B_0(i)$ and $B_1(i)$ directly. On the other hand, if server i is not corrupted, the receiver can get $B_0(i) + B_1(i)y_i$ for any y_i selected. As there are

$m - \lambda_2$ honest servers, the collection of all these y_i implicitly corresponds to some $S(x)$ of degree at most $m - \lambda_2 - 1 = \lambda_1$ such that $S(i) = y_i$. Therefore, the choice of y_i of any malicious receiver will actually be the same as some honest receiver choosing $S(x)$. The value of $S(x)$ is known by the receiver:

$$S(x) = \sum_{j=0}^{\lambda_1} s_j x^j. \tag{12}$$

So the malicious receiver will at least know the value of $a_0 + b_0 s_0$. If we can show that the receiver can only get a linear combination of a_0 and b_0, this is exactly what he gets.

The receiver gets $2\lambda_2$ equations from the corrupted servers, and $m - \lambda_2$ equations in the form of $B_0(i) + B_1(i)S(i) = v_i$ from honest server i. All the coefficients of $B_0(x)$ and $B_1(x)$ are the unknowns and therefore there are $m + \lambda_2 + 1$ of them. There are more unknowns than equations so it is impossible to solve them fully. In particular, with all the information on $B_1(x)$ from the cheating servers alone, the receiver gets the vector

$$\begin{pmatrix} 1 & \beta_1 & \beta_1^2 & \dots & \beta_1^{\lambda_2} \\ \vdots & & \ddots & & \vdots \\ 1 & \beta_{\lambda_2} & \beta_{\lambda_2}^2 & \dots & \beta_{\lambda_2}^{\lambda_2} \end{pmatrix} \begin{pmatrix} b_0 \\ \vdots \\ b_{\lambda_2} \end{pmatrix} \tag{13}$$

where β_i is the identity of the ith cheating server. Using the property of Vandermonde matrices, the receiver can now express each of $(b_1, b_2, \dots b_{\lambda_2})$ as a linear function of b_0, as there is one more unknowns than equations. From this, for any i the value of $B_1(i)S(i)$ is now also a linear function of b_0. An honest server i would give the receiver the value of $v_i = B_0(i) + B_1(i)S(i)$. In such a case $B_0(i) = v_i - B_1(i)S(i)$ can also be expressed as a linear function of b_0. On the other hand, if i is a cheating server then $B_0(i)$ is known directly as a number. Collectively all information about $B_0(x)$ can now be expressed:

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 2^2 & \dots & 2^{m-1} \\ \vdots & & \ddots & & \vdots \\ 1 & m & m^2 & \dots & m^{m-1} \end{pmatrix} \begin{pmatrix} a_0 \\ \vdots \\ a_{m-1} \end{pmatrix} = \begin{pmatrix} u_1 \\ \vdots \\ u_m \end{pmatrix} \tag{14}$$

where every u_i is a known linear function of b_0 (or a constant, which is still a linear function of b_0). Using Vandermonde matrices again, it is clear that the receiver can solve a_0 as a linear function of b_0. This implies that the value of $a_0 + \alpha b_0$ can be known for some α, while the exact value of (a_0, b_0) is unknown. To conclude, in every possible case, the view of the receiver allows him to know $a_0 + s_0 b_0$ only. Other than that, the freedom of a_0 and b_0 remains. Therefore, $a_0 + s_0 b_0$ is all the receiver can get, fulfilling the security requirement of the sub-protocol.

For sender's privacy against servers, it is clear that a coalition of λ_3 servers gets no information about a_0 due to missing information on $B_0(x)$. But they can

get b_0 (as $\lambda_3 > \lambda_2$), which means they also seem to get a linear combination of secrets. The important difference is that they do not have the freedom to choose the combination. Due to this, in the next section we can show the security of the full protocol.

5.3 Full Protocol Avoiding Linear Combination

We note that it is important to prevent the receiver from getting a linear combination of the secrets. In particular, OT security should not depend on what is already known about the secrets when the protocol is started. For example, if ω_0 is known to be a multiple of a number K, and ω_1 is smaller than K, then the knowledge of $\omega_0 + \omega_1$ can reveal both secrets.

To ensure the receiver really only gets one secret, the method suggested in [10] is used. The sub-protocol is run twice in parallel. For the first time, two random values of γ_0 and γ_1 are generated by the sender and the sub-protocol is run with $a_0 = \gamma_0$ and $b_0 = \gamma_1 - \gamma_0$. For the second time, the values of $\gamma_0\omega_0$ and $\gamma_1\omega_1$ are used, where (ω_0, ω_1) are the real secrets. In this run $a_0 = \gamma_0\omega_0$ and $b_0 = \gamma_1\omega_1 - \gamma_0\omega_0$. The receiver then starts his part of protocol but only one $S(x)$ is used, with each honest server i receiving $y_i = S(i)$ only once. This effectively allows the receiver to get $A\gamma_0 + B\gamma_1$ and $A\gamma_0\omega_0 + B\gamma_1\omega_1$ for any A and B with $A + B \neq 0 \bmod p$. The receiver is honest whenever $A = 0$ or $B = 0$. In this case, the required secret ω_σ can be obtained directly. Otherwise, the receiver gets the value of the vector:

$$\begin{pmatrix} A & B \\ A\omega_0 & B\omega_1 \end{pmatrix} \begin{pmatrix} \gamma_0 \\ \gamma_1 \end{pmatrix} \tag{15}$$

such that each possible pair of (ω_0, ω_1) with $\omega_0 \neq \omega_1$ corresponds to a unique pair of (γ_0, γ_1), giving no information on (ω_0, ω_1). To make sure $\omega_0 \neq \omega_1$ we need to add some special rules to the DOT. For example, it can be that all sender's secrets are odd numbers, but the sender can randomly choose to add one to any of them before putting them as the input of the DOT. In the case that the two secrets are equal, one and only one of them is changed to an even number. Although the message space is reduced by half, it can be handled by choosing a larger p.

On the other hand, for sender's privacy against λ_3 servers, it is now obvious that all the malicious servers get is the vector in (15) with $A = -1$ and $B = 1$. Therefore they receive no information about the sender's secrets in the DOT scheme.

This protocol achieves full security of DOT, and reaches the upper bounds for all parameters. The protocol is highly efficient for the servers, as each of them only receives five numbers and returns two simple functions of them. With a fixed value of m, parameters λ_1 and λ_2 can be adjusted freely before the protocol begins. This gives an advantage when one of the sender or receiver is more trustworthy than the other.

6 Extension and Concluding Remarks

Our scheme can be extended to 1-out-of-N DOT. The result is basically same as [11]. The only difference is that all polynomials should be allowed to have any degree smaller than their corresponding maximum. Techniques to keep the receiver from getting a linear combination of secrets are required too.

This scheme can also be extended to the case where M servers are available, but the receiver is only allowed to communicate with any m of them, including the servers corrupted by him. In this case, exactly the same protocol can be used, with the sender communicating with all servers and the receiver only m of them.

To conclude, in this paper we revise the definition of DOT with stronger adversaries and fewer other assumptions. Then we generalize the concept of [10] in order to construct a secure DOT protocol based on polynomials. After considering some known attacks, we show a secure example based on [11]. For the first time, we prove the DOT security explicitly. We also show the general bounds of DOT, which are reached by the system parameters of our scheme.

References

1. Blundo, C., D'Arco, P., De Santis, A., Stinson, D.: On unconditionally secure distributed oblivious transfer. Journal of Cryptology 20(3), 323–373 (2007); A preliminary version appeared in Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 291–309. Springer, Heidelberg (2003)
2. Brassard, G., Crépeau, C., Santha, M.: Oblivious transfers and intersecting codes. IEEE Transactions on Information Theory 42(6), 1769–1780 (1996)
3. Brassard, G., Crépeau, C., Wolf, S.: Oblivious transfers and privacy amplification. Journal of Cryptology 16(4), 219–237 (2003)
4. Crépeau, C.: Equivalence between two flavours of oblivious transfer. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 350–354. Springer, Heidelberg (1988)
5. Crépeau, C., Savvides, G.: Optimal reductions between oblivious transfers using interactive hashing. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 201–221. Springer, Heidelberg (2006)
6. Even, S., Goldreich, O., Lempel, A.: A randomized protocol for signing contracts. Communications of the ACM 28(6), 637–647 (1985)
7. Ghodosi, H.: On insecurity of Naor-Pinkas' distributed oblivious transfer. Information Processing Letters 104(5), 179–182 (2007)
8. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Proc. 19th ACM Symposium on Theory of Computing, pp. 218–229 (1987)
9. Kilian, J.: Founding cryptography on oblivious transfer. In: Proc. 20th ACM Symposium on Theory of Computing, pp. 20–31 (1988)
10. Naor, M., Pinkas, B.: Distributed oblivious transfer. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 205–219. Springer, Heidelberg (2000)

11. Nikov, V., Nikova, S., Preneel, B., Vandewalle, J.: On unconditionally secure distributed oblivious transfer. In: Menezes, A., Sarkar, P. (eds.) INDOCRYPT 2002. LNCS, vol. 2551, pp. 395–408. Springer, Heidelberg (2002)
12. Rabin, M.: How to exchange secrets by oblivious transfer, Technical Report TR-81, Aiken Computation Laboratory, Harvard University (1981)
13. Shamir, A.: How to share a secret. Communications of the ACM 22(11), 612–613 (1979)
14. Yao, A.C.-C.: Protocols for secure computations. In: Proc. 23rd IEEE Symposium on Foundations of Computer Science, pp. 160–164 (1982)

Towards Denial-of-Service-Resilient Key Agreement Protocols

Douglas Stebila[1] and Berkant Ustaoglu[2]

[1] Information Security Institute, Queensland University of Technology, Brisbane, Australia
douglas@stebila.ca
[2] NTT Information Sharing Platform Laboratories, Tokyo, Japan
bustaoglu@cryptolounge.net

Abstract. Denial of service resilience is an important practical consideration for key agreement protocols in any hostile environment such as the Internet. There are well-known models that consider the security of key agreement protocols, but denial of service resilience is not considered as part of these models. Many protocols have been argued to be denial-of-service-resilient, only to be subsequently broken or shown ineffective.

In this work we propose a formal definition of denial of service resilience, a model for secure authenticated key agreement, and show how security and denial of service resilience can be considered in a common framework, with a particular focus on client puzzles. The model accommodates a variety of techniques for achieving denial of service resilience, and we describe one such technique by exhibiting a denial-of-service-resilient secure authenticated key agreement protocol. Our approach addresses the correct integration of denial of service countermeasures with the key agreement protocol to prevent hijacking attacks that would otherwise render the countermeasures irrelevant.

1 Motivation

Reliable, fast, and secure communication is essential for commercial success on today's Internet. Slow web pages could motivate clients to switch to an alternative business, leading to a loss in customer base for the service provider. However, maintaining sufficiently powerful servers can be an expensive venture. Servers have limited resources in terms of the amount of traffic that can be handled, the time required to establish a connection, and the number of active concurrent connections. This effectively bounds the number of connections a company's server can honour in a given period in time.

Malicious parties have recognized that they can benefit by depleting the limited resources of others' servers. *Denial of service attacks* aim to disrupt, destroy, or render services unavailable. A typical denial of service attack exhausts the target's resources. The server is rendered unavailable for honest clients, who then proceed to request similar services from competitors. To prevent malicious requests, a server needs to filter out bogus connection requests and honour those from legitimate clients. Recognizing legitimate clients is difficult. We will view

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 389–406, 2009.

a client as having *legitimate intentions* if it is willing to perform an expensive computation; it still could be malicious, but we may have no further way of distinguishing legitimate from malicious connections with a priori authentication.

Security is an important aspect of online services. Many connections between a client and a server need to be secured against third parties; financial transactions are the most common case. Key agreement is used to produce a shared secret that can be used to encrypt subsequent communication. Key exchange involves computationally expensive algorithms, and hence may dominate server-side run time, limiting the number of clients serviced. This makes key agreement an enticing target for denial of service attacks since a malicious party can easily issue many key agreement requests. Hence, it is advantageous to try to reject as many bogus connections as possible during key agreement. In this paper we are concerned with deterring malicious parties from initiating denial of service attacks based on key agreement, without compromising on security.

Practitioners and standardization bodies have recognized the importance of denial of service resilience, but researchers have been slow to respond with a formal treatment of the subject. In some sense, addressing denial of service resembles the initial approach to key agreement: rather than constructing an overall model, a list of ad hoc goals is selected and then it is shown that a protocol meets those goals. As a result, it is difficult to evaluate the strength and usefulness of denial of service countermeasures when integrated into protocols.

Our Contributions. We give a formal definition of denial of service resilience for key agreement protocols in the context of the extended Canetti-Krawczyk (eCK) model for secure key agreement. Our definition for denial of service resilience is sufficiently strong that it prevents known attacks that arose against protocols once thought to be denial of service resilient. Puzzles have been previously used as a denial of service countermeasure, but in an ad hoc manner. Compared to previous work, our contribution models the careful integration of two orthogonal issues: key agreement security and denial of service resilience.

It is well established that improper use of cryptographic algorithms can render them useless. For example, even the most secure password-based system is of no use if weak passwords are used. Similar reasoning applies for DoS countermeasures: even good solutions are useless if they are not used properly. For example, if a proof-of-work in the form of a puzzle solution does not indicate who is the intended recipient, when the solution was created, or who created the solution, then the door is open for misuse. Without careful integration into the overall protocol, DoS countermeasures may not achieve their goal.

Additionally, we present the DoS-CMQV protocol which is a secure key agreement protocol and uses client puzzles to offer denial of service resilience in our model, showing how to achieve security and denial of service resilience together.

2 Previous Work

Key Agreement. Key agreement is an important cryptographic primitive used for building confidential channels. Designing and analyzing key agreement protocols

is a non-trivial task. Formal models, which allow complexity-theoretic security arguments for authenticated key agreement, were first proposed by Bellare and Rogaway [1] and Blake-Wilson, Johnson, and Menezes [2]. The work of Canetti and Krawczyk [3] is one of the most influential extensions to the original models. Their work was later augmented by Krawzcyk [4] and LaMacchia, Lauter and Mityagin [5] to capture a wider range of desirable security properties; we refer to this as the *extended Canetti-Krawczyk* (eCK) model.

In all of the above models the adversary controls *all* communication links. It is not immediately clear how denial of service can be considered alongside key establishment when the adversary may not deliver messages to destinations. However, even in this setting there are meaningful DoS-related goals that can be incorporated into the model; we discuss our extension in Sect. 3.1.

Denial of Service. There are two main types of denial of service attacks (see [6, §1.6.6], for example): *resource depletion* attacks and *connection depletion* attacks. In resource depletion attacks a malicious party attempts to drain a server's computational or memory resources. By contrast, connection depletion attacks aim to exhaust the number of allowed connections to the server. A DoS countermeasure can aim to defend against either or both of these types of attacks.

Distributed denial of service (DDoS) attacks, in which many distributed client computers attack a single server, are of significant concern on the Internet today. These types of attacks are very difficult to defend against. One known technique, which we use in this paper, is to allow a server to adjust its denial of service countermeasure based on the load it experiences. Puzzle auctions [7] are one such implementation of tunable puzzles.

Aura and Nikander [8] introduced the notion of *stateless connections*, in which stateful connections are transformed into stateless ones by attaching the state information to the message and using a message authentication code for integrity. This gives some protection against denial of service by saving the server from having to store session information until later in the exchange when more assurance is possible.

Meadows [9] offered the first formal framework for denial-of-service-resilient protocols, based on the causal sequencing language of fail-stop protocols of Gong and Syverson [10]. To avoid connection depletion, Meadows suggests that each message be authenticated with increasingly complex levels of authentication. Meadows then applies this framework to the Station-to-Station protocol [11] to identify potential DoS attacks but does not provide a denial-of-service-resilient protocol. An application of Meadow's cost-based framework to the JFK protocol revealed a potential DoS attack, and a solution to this problem was proposed using client puzzles [12]. This underscores the ability of formal models to reveal flaws and the need for the formalization of denial of service resilience.

Cookies. One of the first techniques used to defend protocols against denial of service attacks was cookies. Introduced in the Photuris protocol (published in 1999 as [13] but introduced earlier), *cookies* are small authentication tokens returned by a server upon initial connection by the client. In order for the client to

be allowed to continue with the connection, the client must echo the cookie back to the server. The server does not store the cookie, instead using the stateless connection technique to check the authenticity of the cookie which the client includes in subsequent messages. Cookies can be applied in Meadows' framework as an early level of authentication.

Krawczyk's SIGMA protocol [14] was proposed as a successor of the Internet Key Exchange (IKE) protocol used in IPsec, and was adapted to have denial of service resilience in the form of cookies in its implementation in IKEv2 [15]. Cookies are also used in the Just Fast Keying protocol (JFK) proposed by Aiello et al. [16]. JFK allows the server to reuse its ephemeral private-public key pair across multiple sessions to reduce the server's computational load, at the expense of increasing the potential damage should an ephemeral private key be leaked.

Cookies are a valuable first-order denial of service countermeasure and have been used extensively as described above. However, they are a weak form of denial of service resilience because they do not require an attacker to do anything other than faithfully relay a previously received cookie.

Protocols using cookies can also be susceptible to other types of attacks. Mao and Paterson [17, §2.2] described a denial of service attack against IKEv2. In their attack, a malicious party who controls a popular server $\hat{M}$ can redirect legitimate traffic from $\hat{M}$ towards another target server $\hat{M}'$, thereby effecting a denial of service attack against $\hat{M}'$. The attack only costs $\hat{M}$ bandwidth, not computation or memory, and is resilient to cookie-based DoS countermeasures. This attack is possible because there is no strong binding between the DoS countermeasure and the identity of the server to which the client wishes to connect: we codify this notion in our security criterion DoS-2 in Sect. 3. Despite key agreement and denial of service being orthogonal issues, combining them is no trivial task, as demonstrated by this attack.

Puzzles. Dwork and Naor [18] introduced the notion of *client puzzles* to defend against denial of service attacks. A server under a denial of service attack can require clients to find the solution to a puzzle before the server allocates resources: the puzzle should be hard to solve but the solution should be easy to verify. Back [19] and Juels and Brainard [20] suggested using a hash function so that a client must perform a large number of operations to find the solution; this is a *computation-bound* puzzle. We build on their approach by specifying how puzzles should be integrated with key agreement. Puzzles where computing the solution is more dependent on memory access time, called *memory-bound puzzles*, have also been suggested for use [21]; these offer less varied running times across different hardware platforms because memory access times vary less than processor speed. Waters et al. [22] described how puzzles can be distributed across multiple servers for coordinated access.

Aura, Nikander, and Leiwo [23] gave a framework for using hash function preimages as a denial of service in authentication protocols, and lay out the basic principle that "the client should always commit its resources to the authentication protocol first and the server should be able to verify the client commitment before allocating its own resources". We use this principle to develop a model for

denial-of-service-resilient key agreement. The technique of [23] is not sufficient to defend against the attack of Mao and Paterson [17]; our approach is.

This principle of clients committing resources before the server does applies well to preemptive DoS countermeasures. The server obtains assurance that the client committed resources, but this is no guarantee that the client will *complete* the request. If a client does not finish a request then what should the actions of the server be? What should a server do if a client takes too long to respond after presenting its proof-of-work? Even though such open connections have important practical significance, preemptive measures do not completely cover the problem of open connections, such as the half-open connections of TCP SYN flood attacks [24]; a common countermeasure is to discard old uncompleted open connections.

3 Modelling Denial of Service Resilience and Security

We begin with an informal description of the goals of a denial-of-service-resilient protocol and then proceed to outline a formal model for integrating denial of service resilience and secure key agreement protocols. While the goals of denial of service resilience and secure key agreement are, as others such as Krawczyk [14, §2.3] have noted, orthogonal issues, it is useful to be able to discuss them in a common framework. We must be careful to integrate the two issues sufficiently well to avoid the types of attacks proposed by Mao and Paterson [17].

Denial of Service Resilience Intuition. We are concerned about the situation in which a malicious party on the network can cause a server to perform many expensive operations (and key agreement is one such expensive operation) for no good reason, eventually consuming all of the server's available resources. But since the server is willing to place itself on the network for the use of all users, how can the server know if it is doing work for a good reason or not? Distinguishing legitimate requests from malicious requests is an essential element of denial of service resilience.

While one can never be certain about the good intentions of another party on the network, it is plausible to believe that a client is making a legitimate request if the client is willing to commit some expensive resources – computation, memory, etc. – to the connection request. However, if a client does do something expensive to prove its good faith, then a good protocol should protect the client from being exploited by a malicious party aiming to steal the client's work. Last but not least, a server should be able to adapt its procedures to resist being flooded by many "honest" requests that may be coordinated for maximum impact.

These ideas lead us to the following five informal criteria for a denial-of-service-resilient protocol:

DoS-1. An uncompromised honest server $\hat{B}$ does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.

DoS-2. Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.

DoS-3. A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.

DoS-4. A malicious party[1] must use a very large amount of resources if it wishes to prepare sufficiently many connection requests and "flood" a server with many valid connection requests.

DoS-5. A server can adjust the amount of work a client has to do in times of higher or lower load.

In Sect. 3.1, we give a formal definition of denial of service resilience and describe in Sect. 3.2 how it achieves each of the goals DoS-1 through DoS-4; goal DoS-5 is a property of a particular countermeasure and not of the formal model.

The first two goals aim to protect the server from performing unnecessary expensive operations. While what qualifies as an expensive operation can vary depending on the setting, we identify three main classes of expensive operations for the purposes of denial of service: *memory denial of service* attacks, in which the server is forced to perform slow, expensive memory reads or writes or use large amounts of memory; *computational denial of service* attacks, in which the server is forced to perform operations requiring significant computational time (such as exponentiation, elliptic curve point multiplication, or many simpler operations such as hash function or MAC evaluations); and *transmission denial of service* attacks, in which the server is forced to expend resources available to send and receive communications. In various situations, different notions of expensive can apply; for example, in mobile environments, transmitting and receiving take a lot of time and power.

To achieve denial of service resilience, we require that a client answer a puzzle that takes significant computational or memory resources to solve, but is easy for a server to prepare and verify. The key idea is to tightly bind the puzzles with the identities of the parties involved to prevent attacks in which work can be stolen or redirected. This allows a server to be more convinced of a client's legitimate intention to engage in a key agreement protocol.

Secure Key Agreement Intuition. As noted in Sect. 2, secure key agreement has been extensively studied. The goal of an adversary in the eCK model is to learn any information about the session key established by a pair of uncompromised participants. If an adversary cannot distinguish such a session key from a randomly selected string with non-negligible probability, then the session key is viewed as secure and suitable for use in bulk encryption. The model is formally described in Sect. 3.1 and the definitions of security are given in Sect. 3.1.

3.1 Formal Model Description

Our model is based on the extended Canetti-Krawczyk (eCK) model proposed in [5]. However, instead of all exchanged messages we use a fingerprint of exchanged messages to identify session.

[1] A malicious party can be a single entity or a collection of entities working together.

A protocol takes place among n parties $\mathsf{Parties} = \{\hat{A}, \hat{B}, \ldots\}$, where a *party* is a probabilistic polynomial-time Turing machine. In addition to the certified and validated static key pair, each party also possesses static information that is not certified. If non-empty, the non-certified information may be either private or public. Parties are activated via incoming messages, which are then processed within the party. As a result a party either returns its outgoing response or indicates if the processing resulted in failure or success.

Pre-session and Session Creation. An execution of the protocol is called a *session*. A party may receive an incoming *request* to initiate a session via a message (i) $(\hat{A}, \hat{B})$ or (ii) $(\hat{A}, \hat{B}, \text{“hello”})$. In the former case $\hat{A}$ is called the *client* or *initiator*, and reacts by creating a separate *session request* $(\hat{B}, \hat{A}, \text{“hello”})$ designated for $\hat{B}$. In the latter case, $\hat{A}$ is called the *server* or *responder*, and creates a separate *session request* $(\hat{B}, \hat{A}, \text{“hello”})$ designated for $\hat{B}$. Server $\hat{A}$ selects a fresh (unique within $\hat{A}$) challenge ch and sends $(\hat{B}, \hat{A}, \text{“hello”}, \mathsf{ch})$ to $\hat{B}$. Within a party the execution of the subroutines between the session request and either accepting or rejecting the request to initiate a session is called a *pre-session*. The model does not prevent a protocol from accepting multiple distinct responses to the same challenge.

The motivation for the pre-session is to include the denial of service countermeasure in the pre-session and leave expensive operations and resources commitment by a server for the session. A session can only be reached after a successful pre-session: in other words, expensive resources will only be committed by the server once the denial of service countermeasure are passed.

A party $\hat{A}$ can be activated to *create* a session with a message of the form (i) $(\hat{A}, \hat{B}, \text{“hello”}, \mathsf{ch})$ or (ii) $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$. If the activation is of type (i) then $\hat{A}$, who is the initiator, prepares re that passes all protocol conditions and creates an *active* session. The string re must be unique within $\hat{A}$; the outgoing message is $(\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re})$. If the activation is of type (ii) then $\hat{A}$, who is the responder in this case, verifies that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ satisfies the protocol requirements; if so a new active session is created, otherwise the message is ignored. If a responder $\hat{A}$ creates a new session, then the outgoing message is (Ψ, mesg), where mesg is prepared by $\hat{A}$ in accordance with the protocol and Ψ is the *session string identifier*, a string used to identify sessions within $\hat{A}$ and $\hat{B}$, which is derived from $(\mathsf{ch}, \mathsf{re})$ and possibly other publicly known parameters. The conditions imposed on ch and re allow for the derivation of a string unique within both $\hat{A}$ and $\hat{B}$.

Session State. Upon creating a session, $\hat{A}$ also creates a separate *session state* that contains both private and public session-specific information. The private information is needed to derive a secret session key. The public information is $(\hat{A}, \hat{B}, \Psi, \mathsf{role}, \mathsf{otherinfo})$, where $\hat{B}$ is the purported *session peer*; role is either “initiator” or “responder” and $\mathsf{otherinfo}$ is any other public information required by the protocol. Globally the session is identified via $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ and Ψ identifies the session within $\hat{A}$. For $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ we call $\hat{A}$ the owner and together $\hat{A}$ and $\hat{B}$ are the *communicating partners* of sid. Sessions $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$ and $\mathsf{sid}^* = [\hat{C}, \hat{D}, \Psi']$ are *matching* if $\Psi = \Psi'$, $\hat{A} = \hat{D}$, and $\hat{C} = \hat{B}$.

As in the eCK model, $\hat{A}$ can be activated to update a session via a message of the form (Ψ, mesg). Upon receipt of such a message, $\hat{A}$ performs validation procedures similar to the eCK model and updates its state. At any stage a session is in exactly one of the following states: *active*, *completed* or *aborted*.

Adversary. The adversary $\mathcal{M}$ is a probabilistic Turing machine that controls *all* communications and party activation via the query $\mathsf{Send}(\cdot)$. Parties present $\mathcal{M}$ with their outgoing messages. Leakage of private information to $\mathcal{M}$ is modelled via the following adversary queries:

- $\mathsf{StaticKeyReveal}(\hat{A})$: $\mathcal{M}$ obtains $\hat{A}$'s static private key.
- $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$: $\mathcal{M}$ obtains the ephemeral private key of $\hat{A}$ in session $\mathsf{sid} = [\hat{A}, \hat{B}, \Psi]$.
- $\mathsf{SessionKeyReveal}(\mathsf{sid})$: If sid has completed then $\mathcal{M}$ obtains the session key in sid.
- $\mathsf{Establish}(\hat{M}, M)$: $\mathcal{M}$ registers an *adversary-controlled* party $\hat{M}$ with static public key $M \in \mathcal{G}$. If a party is not adversary-controlled it is said to be *honest*.
- $\mathsf{DoSExpose}(\hat{A})$: $\mathcal{M}$ obtains the non-certified private information belonging to an honest $\hat{A}$, excluding the session-related ephemeral private keys. Parties against which this query was issued are called *DoS-exposed*, otherwise they are called *DoS-unexposed*.

The role of the new $\mathsf{DoSExpose}$ query in our model is to allow us to identify the parties which ought to still be resilient to denial of service attacks, namely those parties which are DoS-unexposed. For example, adversary-controlled parties are not relevant to the DoS portion of the protocol. Moreover, a separate $\mathsf{DoSExpose}$ query allows us to separate denial of service resilience from key agreement security: compromise of key agreement secrets can be an orthogonal issue to compromise of denial of service.

Key Agreement Security Definitions. Security is defined via indistinguishability. At any time during the experiment $\mathcal{M}$ can make one special query, $\mathsf{Test}(\mathsf{sid})$, to a session sid that must remain fresh throughout the experiment. The goal of $\mathcal{M}$ is to guess whether the response to the query is sid's key or a random key.

Definition 1 (Fresh session). *Let* sid *be the identifier of a completed session, owned by an honest party* $\hat{A}$ *with peer* $\hat{B}$*, who is also honest. Let* sid^* *be the identifier of the matching session of* sid*, if it exists. Then* sid *is* fresh *if none of the following conditions hold:*

1. $\mathcal{M}$ *issued* $\mathsf{SessionKeyReveal}(\mathsf{sid})$ *or* $\mathsf{SessionKeyReveal}(\mathsf{sid}^*)$ *(if* sid^* *exists).*
2. sid^* *exists and* $\mathcal{M}$ *issued one of the following: either*
   (a) both $\mathsf{StaticKeyReveal}(\hat{A})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*
   (b) both $\mathsf{StaticKeyReveal}(\hat{B})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid}^*)$*.*

3. sid^* *does not exist and* $\mathcal{M}$ *issued one of the following: either*
   (a) both $\mathsf{StaticKeyReveal}(\hat{A})$ *and* $\mathsf{EphemeralKeyReveal}(\mathsf{sid})$*, or*
   (b) $\mathsf{StaticKeyReveal}(\hat{B})$.

Definition 2 (Secure key agreement protocol). *A key agreement protocol is* secure *if the following conditions hold: (i) two honest parties that complete matching sessions then, except with negligible probability they compute the same session key; and (ii) no polynomially bounded adversary* $\mathcal{M}$ *can distinguish the session key of a fresh session from a randomly chosen session key with probability greater than* $\frac{1}{2}$ *plus a negligible fraction (in the security parameter).*

Denial of Service Definitions. In the protocol there is a test that the server performs on some of the messages received to determine if the client has done sufficient work to merit the server performing expensive operations. The client's work, modelled by a puzzling relation, is used to define the protocol's denial of service resilience.

Definition 3 (Puzzling relation). *Let* $\mathsf{Challenges}$ *and* $\mathsf{Responses}$ *be sets. A relation* $\mathcal{R} \subseteq \mathsf{Parties} \times \mathsf{Parties} \times \mathsf{Challenges} \times \mathsf{Responses}$ *is a* puzzling relation *if*

1. *deciding if* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *is "easy", and*
2. *given* $\hat{A}, \hat{B}, \mathsf{ch}$*, and an oracle* U *that, on input* $(\hat{A}', \hat{B}', \mathsf{ch}')$*, returns* re' *with* $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$*, it is "hard" to produce* re *such that* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ *and* re *was not a response generated by the oracle* U *upon input* $(\hat{A}, \hat{B}, \mathsf{ch})$.

The notion of "expensive operation", "easy", and "hard" will depend on the application context. Although we have left these notion vague, they can be formalized, for example as a proof of work [25]. We omit this formalization as the focus of our work is on the integration of denial of service resilience techniques into key agreement protocols, not the construction of suitable puzzles.

Definition 4 (Acceptable pre-session). *A pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *is an* acceptable pre-session *for* $\hat{B}$ *if* $\hat{B}$ *generated* ch.

Definition 5 (Denial-of-service-resilient protocol). *Let* $\mathcal{R}$ *be a puzzling relation. A protocol* Π *is* denial-of-service-resilient *if the following hold for every DoS-unexposed server* $\hat{B}$*:*

1. $\hat{B}$ *only performs expensive operations (a) in a session, or (b) for some (low frequency) periodic update of its non-certified private information* ρ*, and*
2. $\hat{B}$ *only establishes a session* $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$ *if the pre-session* $[\hat{A}, \hat{B}, \mathsf{ch}]$ *was an acceptable pre-session for* $\hat{B}$ *and* $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$.

Note that we have explicitly avoided merging Definitions 4 and 5 (as Definition 4 could be part of Condition 2 of Definition 5) because we acknowledge that there may be situations that require a different notion of acceptable pre-session, for example one-pass key agreement protocols where both ch and re are generated by the initiator. In particular, it appears to suffice that $\hat{B}$ has an assurance that ch was generated independently at random before re.

3.2 Model Implications

In this section we explain how our formal model of denial of service resilience in Definition 5 satisfies the informal goals for denial of service resilience of Sect. 3.

DoS-1. An uncompromised honest server $\hat{B}$ does not perform any expensive operations with a client unless it is convinced the client is trying to make a legitimate connection.

By condition 1 of Definition 5, a server $\hat{B}$ does not perform any expensive operation with a client until it has established a session, and by condition 2 it will not establish a session until it received a response to its challenge that satisfies the puzzling relation. In order to satisfy the puzzling relation, the client must do a significant amount of work because of condition 2 of Definition 3; by doing this work, the client convinces the server that it is trying to make a legitimate connection. Moreover, since sessions are unique within a party, replay attacks of legitimate connection requests are prevented.

DoS-2. Moreover, a server $\hat{B}$ does not perform any expensive operations unless it is convinced that the client wants to talk to $\hat{B}$ and not another server $\hat{M}$.

Condition 2 of Definition 3 allows us to meet this criterion: even if an adversary obtains any tuple $(\hat{A}, \hat{M}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$, the tuple $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})$ is unlikely to be in $\mathcal{R}$ and moreover it remains hard to produce a response re' such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}') \in \mathcal{R}$. Since it is hard to create such a tuple given oracle access to $\mathcal{R}$, then it is still hard to construct any tuple in $\mathcal{R}$ without oracle access.

Our approach avoids the attack of Mao and Paterson [17] against IKEv2 in which an attacker can redirect traffic from her server towards other servers and can cause the receiving server to deplete its connection resources at low expense to the attacker. That attack is possible because there is no cryptographic binding between the denial of service countermeasure and the identities of the parties involved. By including the names of the client and server in the puzzling relation, a server $\hat{B}$ can be assured that whoever solved the puzzle intended to communicate with $\hat{B}$.

DoS-3. A client $\hat{A}$ who commits significant resources to prove its legitimate intentions cannot have her work stolen: the work that $\hat{A}$ does to convince $\hat{B}$ that it wants to communicate legitimately with $\hat{B}$ cannot convince anyone of anything else.

Suppose $\hat{B}$ is a DoS-unexposed server and suppose an honest client $\hat{A}$ starts a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, and then finds a value re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$. The client wishes that the response value should not be useful to anyone else trying to establish a session; in other words, no one should be able to steal $\hat{A}$'s work and use it in another pre-session.

Suppose $[\hat{A}', \hat{B}', \mathsf{ch}']$ is another pre-session. Given these values, it is hard to produce re' such that $(\hat{A}', \hat{B}', \mathsf{ch}', \mathsf{re}') \in \mathcal{R}$, even with help from another pre-session such as $[\hat{A}, \hat{B}, \mathsf{ch}]$, because $\mathcal{R}$ is a puzzling relation. The help given by the other pre-session can be modelled as one response of the oracle U in Definition 3

for another pre-session, and this is of no help in a puzzling relation. Thus, an honest client's work in solving a puzzle is of no use to anyone else responding to a different challenge, or with a different server, or with a different user name.

If the adversary $\mathcal{M}$ simply relays $\hat{A}$'s entire response and then participates in $\hat{A}$'s place, the server will proceed with key agreement but this session will ultimately fail, since it is secure in the sense of Definition 2 key agreement protocols, and thus $\mathcal{M}$ cannot complete a session masquerading as $\hat{A}$.

DoS-4. A malicious party must use a very significant amount of resources if it wishes to prepare sufficiently many connection requests and "flood" a server with many valid connection requests.

As noted in DoS-1, a server will only perform expensive operations if it has been convinced that the client is trying to make a legitimate connection, meaning the client has solved an instance of the puzzling relation, which is "hard" and requires a significant amount of resources. Suppose that for an attacker to start a session requires t steps of computation (e.g., t may be the number of cycles it takes to solve a computationally bound puzzling relation), and a server has enough computational resources to support n connections per second. Then, roughly speaking, an attacker's computers must be able to perform tn steps of computation per second to sustainably render the server unavailable through a denial of service attack, which may require distributed resources. By registering many dishonest parties the above attack can be incorporated in our model. While not completely defending against such powerful distributed attacks, we can at least allow the amount of denial of service resilience to be tuned in the event of heavy traffic. On the other hand what our model assures is that the adversary cannot *use* honest parties to mount such distributed attacks. For example, a DoS-resilient protocol guards against the attack where the adversary registers a single malicious party $\hat{M}$, initiates pre-sessions between honest parties and $\hat{M}$, and then forwards messages from the honest parties to create many sessions at an honest server. That is, the model defends against Mao-Patterson type of attacks.

Consider also the case of *replay attacks*, in which an attacker resends the same message many times to a server. Suppose in particular that an attacker replays a response value re for a pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$. This set of values leads to the server session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$, which already exists in the server. Since sessions identifiers are unique in the model, the server will not start a new session and hence commit no new resources as a result of this replay. This requires the server to store a table of session identifiers, but this does not result in a denial of service attack in theory since the server commits memory resources for the entries in the table only once the puzzling relation has been passed. To limit the size of the table, the server could change the non-certified information ρ periodically. When receiving a previous challenge and response, an entirely acceptable action would be for the server to respond to the replay with the same response it gave previously; this prevents puzzle stealing attacks where the adversary responds to a puzzle faster than a legitimate client. Now, if the attacker were to compute a different response value re' for the pre-session $[\hat{A}, \hat{B}, \mathsf{ch}]$, then the server would commit

new resources to the new session, but this is acceptable since the attacker solved the puzzling relation, just as a legitimate client must.

Since in the Canetti-Krawczyk model we allow the adversary to control the delivery of messages, an adversary may choose not to deliver the final message from the client to the server and leave the server with an incomplete session (similar to the half-open connections of TCP SYN flood attacks [24]). Our model does not view this as a denial of service attack, because the server has been assured that the other party performed many expensive computations to create the connection. This type of attack can be mitigated, without affecting the security assurance nor preventive DoS countermeasures, by conventional server policies to deal with open connections that are not completed for a predefined period of time. Such countermeasures can be separately addressed once the server is assured about the correct connection between the proof of work, the client's identity and the client's intended recipient – exactly what our model provides.

4 A Secure DoS-Resilient Key Agreement Protocol

Our DoS-CMQV protocol, given in Fig. 1, is an adaptation of the CMQV [26] secure authenticated key agreement protocol. We use the problem of finding preimages for a random hash function as the expensive puzzle at the heart of the puzzling relation that a client needs to solve.

The notation $L[i]$ refers to the ith component in the tuple L. H_0 and H_1 are random hash functions [1] that return bit strings; all other hash functions return random integers between 1 and q, the order of the group $\mathcal{G}$ generated by g. We use $x_{[1\ldots w]}$ to denote the first w bits of x. We note that in practice H_1 should be chosen so as to be unique to the protocol so that puzzles cannot be outsourced to another protocol; for example, $H_1(\ldots) = \mathsf{SHA\text{-}256}(\text{“DoS-CMQV”}, 1, \ldots)$.

4.1 Security Analysis

Theorem 1. *If $H_0, H_1, \ldots, H_5$ are random oracles [1], and $\mathcal{G}$ is a group where the Gap Diffie-Hellman (GDH) assumption [27] holds, then DoS-CMQV is a secure key agreement protocol.*

Argument. The DoS-CMQV security argument is similar to the argument presented for CMQV in [26]. We proceed to outline the argument. Verifying condition 1 of Definition 2 is straightforward. It remains to verify condition 2.

In the model here parties possess additional (non-certified) private information ρ, which the adversary can obtain via DoSExpose query. For each of the events in the analysis of CMQV, the solver establishes and simulates the parties similar to the CMQV analysis. The main difference is that when parties are established the solver selects randomly the value ρ for each party. The DoSExpose queries are answered faithfully and they do not affect the freshness of the session. Since the new adversary query is not relevant to the security analysis of the events, the solver can transform the DoS-CMQV adversary to a GDH solver with similar success and running time as a CMQV adversary. Hence a polynomially bounded DoS-CMQV adversary contradicts the assumptions in the theorem. □

DoS-CMQV with security parameter λ		
	Client $\hat{A}$	Server $\hat{B}$
0.	$g, a, A = g^a, B$	$g, b, B = g^b, A, \rho \in_R \{0,1\}^\lambda$
1.		$\xrightarrow{\text{“hello”}, \hat{A}, \hat{B}}$ $i \in_R \{0,1\}^\lambda$
2.		$j = H_0(\rho, \hat{A}, \hat{B}, i)$
3.	store $\tilde{x} \in_R \{0,1\}^\lambda$	$\xleftarrow{\mathsf{ch}}$ $\mathsf{ch} = (i, j)$
4.	$x = H_2(\tilde{x}, a), X = g^x$	
5.	find ℓ s.t. $H_1(\hat{A}, \hat{B}, \mathsf{ch}, X, \ell)_{[1\ldots20]} = 0\ldots0$	
6.	$\mathsf{re} = (X, \ell), \Psi = (\mathsf{ch}, \mathsf{re})$	
7.	establish session $[\hat{A}, \hat{B}, \Psi]$	$\xrightarrow{\hat{A}, \mathsf{ch}, \mathsf{re}}$ verify $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$
8.		verify $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0\ldots0$
9.		establish unique session $[\hat{B}, \hat{A}, \mathsf{ch}, \mathsf{re}]$
10.		store $\hat{A}, \Psi = (\mathsf{ch}, \mathsf{re})$
11.		$X = \mathsf{re}[1]$
12.		verify $X \in \mathcal{G}$
13.		$\tilde{y} \in_R \{0,1\}^\lambda, y = H_2(\tilde{y}, b)$
14.		store $Y = g^y$
15.		$d = H_3(X, \hat{A}, \hat{B}), e = H_3(Y, \hat{A}, \hat{B})$
16.		$\sigma = (XA^d)^{y+eb}$
17.		store $M_1 = H_4(\text{“server finished”}, \hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
18.		store $M_2 = H_4(\text{“client finished”}, \hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
19.	verify $Y \in \mathcal{G}$	$\xleftarrow{\Psi, Y, M_1}$ store $K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$
20.	$d = H_3(X, \hat{A}, \hat{B}), e = H_3(Y, \hat{A}, \hat{B})$	
21.	$\sigma = (YB^e)^{x+da}$	
22.	verify M_1	
23.	$M_2 = H_5(\text{“client finished”}, \hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$	
24.	$K = H_5(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}, Y, \sigma)$	$\xrightarrow{\Psi, M_2}$ verify M_2

Fig. 1. DoS-CMQV: A denial-of-service-resilient adaptation of the CMQV protocol

4.2 Denial of Service Resilience Analysis

In this section we show that the DoS-CMQV protocol given in Fig. 1 is denial-of-service-resilient according to Definition 5. Since this definition (and the related definition of a puzzling relation) includes the intentionally vague terms “expensive operation”, “easy”, and “hard”, we need to define what these terms mean for a concrete instantiation of the definition.

For our purposes, an *expensive operation* is one of the following operations: storing a per-connection or per-session value in memory (other than a long-term value), performing a group exponentiation, or making a large number of calls (say, more than 2^{10}) to a hash oracle.

We first establish, via the following lemma, that the relation used in the DoS-CMQV protocol is a puzzling relation and then show that our protocol is resilient to denial of service attacks

Lemma 1. *Let $\mathcal{R}$ be the relation defined such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ if and only if $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0\ldots0$, where H_1 is a random hash function. Then $\mathcal{R}$ is a puzzling relation, where "hard" means requiring approximately 2^{20} hash function queries on average, and "easy" is something that is not an "expensive operation" as defined above.*

Argument. Deciding membership in $\mathcal{R}$ is easy for a particular tuple because it involves only a single call to H_1.

Moreover, given $\hat{A}$, $\hat{B}$, and a random ch, producing a value re such that $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re})_{[1\ldots20]} = 0\ldots0$ is hard and requires approximately 2^{20} hash oracle queries on average. To find such an re requires finding a preimage for the random hash function. The oracle U helps us find other preimages of H_1. Our task, then is to find a preimage of the correct format involving $\hat{A}$, $\hat{B}$, ch. But since H_1 is a random hash function, other outputs do not help in finding a preimage for this input. Since H_1 is a random hash function outputting 20 bits, this is a hard task that requires approximately 2^{20} queries on average.

This hash puzzle is similar to the partial inversion proof of work (PIPOW) problem of Jakobsson and Juels [25, §3.1]. By their Claim 1, we know that any prover $\hat{A}$ with memory bounded by m who performs on average at most w steps of computation and is given $(\hat{A}, \hat{B}, \mathsf{ch})$ can find a response re such that $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$ with probability at most $p + o(m/2^{20})$ where $p = 1/(2^{20} - w)$.

Theorem 2. *The DoS-CMQV protocol is a denial-of-service-resilient protocol, where "easy", "hard", and "expensive operation" are defined as above.*

Argument. By the Lemma above, $\mathcal{R}$ is a puzzling relation.

Let $[\hat{A}, \hat{B}, \mathsf{ch}]$ be a pre-session. According to the protocol, $\hat{B}$ does not perform any expensive operation until line 10, which is not reached unless the server's checks on lines 7 and 8 are passed and a new session is established on line 9.

If the check on line 8 is passed, namely if $H_1(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}[1], \mathsf{re}[2])_{[1\ldots20]} = 0\ldots0$, then $(\hat{A}, \hat{B}, \mathsf{ch}, \mathsf{re}) \in \mathcal{R}$. If the check on line 7 is passed, namely if $\mathsf{ch}[2] = H_0(\rho, \hat{A}, \hat{B}, \mathsf{ch}[1])$, then, except with negligible probability, $\mathsf{ch}[2]$ was generated only be someone who knew both ρ and $\mathsf{ch}[1]$. Since $\hat{B}$ is a DoS-unexposed party, no $\mathsf{DoSExpose}(\hat{B})$ query could have been issued and since ρ is only ever used as an input to a random oracle, only $\hat{B}$ knows ρ. Thus, $[\hat{A}, \hat{B}, \mathsf{ch}]$ is an acceptable pre-session.

Hence, $\hat{B}$ establishes a session only if the corresponding pre-session is acceptable and the tuple is in the puzzling relation. Note that since sessions must be unique within a party, the server only performs these expensive operations once per session. Thus, DoS-CMQV is a denial-of-service-resilient protocol.

Tuning the Puzzling Relation. The puzzle used in DoS-CMQV can be tuned by the server based on its load. The client must find a hash function preimage; for concreteness, we have specified that the first 20 bits should be zeros, but the length could be a parameter w set by the server depending on its current load.

The server would need to include w in the computation of j on line 2, return w as part of ch on line 3, and include w in the check on line 7 to avoid spoofing.

In practice, H_1 could be implemented by using a standard cryptographic hash function, such as SHA-1, and truncating the output to the first w bits. In times of light load, the server could require that clients truncate only to the first 5 or 10 bits of output, but in heavier load could require that clients truncate to 20 or 25 bits of output to make the cost of mounting a denial of service attack higher. It takes just under 3 seconds to perform 2^{20} SHA-1 evaluations on one core of our 2 GHz Intel Core 2 Duo processor using OpenSSL 0.9.7ℓ. This may be an acceptable computational burden for the client in many scenarios.

5 Other Denial of Service Constructions

Memory-Bound Puzzling Relations. While the protocol given in Sect. 4 uses a puzzling relation based on finding preimages in a hash function, other types of puzzling relations can be used, demonstrating the flexibility of our framework. Abadi et al. [21], for example, described puzzles in which memory access time provides an expected lower bound on the time it takes to solve the puzzle, removing disparities in processor speed between large computers and small devices. However, care must be taken in choosing parameters for memory-bound puzzles: the cost incurred by a server in setting up one of these memory-bound puzzles, while much less expensive than the cost incurred by a client solving the puzzle, can still be significant. For the memory-bound puzzles of [21], it took a 2.4 GHz Pentium 4 server approximately 2^{-7} seconds to create a puzzle that takes approximately 2^2 seconds to solve. By comparison, the time to do one 1024-bit modular exponentiation on a computer of similar speed is less: only 2^{-9} seconds.

JFKi. In the JFKi protocol of [16, §2.3], the denial of service resilience goal for the server is to avoid expensive operations unless the client performs resource-heavy operations, namely group exponentiations. There are two main ideas used: reuse of ephemeral public keys and use of a keyed hash function. The purpose of reusing ephemeral public keys is to distribute the cost of an expensive operation across multiple sessions. This allows the authors to argue the client must perform her share of the work first, in terms of bearing the cost of establishing a round communication trip. The keyed hash function is used by the server to verify that the client indeed executed the round. Note that the server does not need to dedicate any resources to verify the challenge was created by the server. This can be viewed as the pre-session stage of the protocol since the goal of the first round trip is to filter out bogus connections.

JFKi can be described in our model of denial of service resilience, but with weak definitions of "hard" in the puzzling relation. In the implied puzzling relation in JFKi, the client must echo back to the server all the received values and the preimage of the client's nonce. If the puzzling relation test passes, then the server establishes a new session and computes the shared Diffie-Hellman key.

The problem with JFKi's puzzling relation is that there is no binding between the client's ephemeral public key and the solution to the puzzling relation. A

dishonest client can use the same solution to the puzzling relation with different ephemeral public keys (and it can generate these ephemeral public keys very cheaply, for example, by generating g^i, $g \cdot g^i = g^{i+1}, g \cdot g^{i+1} = g^{i+2}, \ldots$) to cause the server to perform many exponentiations with little cost to the client. Thus, given the JFKi puzzling relation and a single solution to the puzzling relation, generating more solutions is easy, contradicting Condition 2 of Definition 3.

Hence, JFKi does not satisfy informal goal DoS-4: the protocol is not resilient to flooding attacks. DoS-CMQV avoids this problem: producing a solution to the puzzling relation means finding a preimage in the hash function and the values cannot be repeated if the server is to establish a new session.

One approach to fixing the denial of service resilience of JFKi was given by Smith et al. [12]. They note that JFKi is not denial-of-service-resilient when analyzed under Meadows' framework [9]. They use a hash function preimage puzzle as well to bind the puzzle solution to the key exchange session at hand. There construction still preserves a fundamental design characteristic of JFKi: the responder must reuse its ephemeral private key in order to achieve denial of service resilience, preventing full freshness in the Canetti-Krawczyk model.

Host Identity Protocol. The Host Identity Protocol (HIP) [28] was designed to offer protection against denial of service attacks. HIP (in [28, §4.1.1]) uses a similar puzzling relation to that of Sect. 4: the client must find a preimage in SHA-1 such that the k lowest-order bits of the output are zero. HIP includes the identities of the initiator and responder in the hash function computation as we have done. Our model provides a theoretical interpretation of the security of HIP against denial of service attacks and the value of including the client and server identities in the hash function computation.

6 Conclusion and Open Problems

We have given the first formal definition of denial of service resilience for secure key agreement protocols. Our model uses puzzles solved by the client as an indication of interest in a legitimate connection, and a variety of puzzles, both memory-bound and computation-bound, can be used. We described a protocol, DoS-CMQV, that provides resilience to denial of service attacks and offers secure key agreement. Additionally, we analyzed the existing JFKi and HIP protocols to compare their notions of denial of service resilience.

Denial of service resistance countermeasures often depend on the freshness of challenges. Our model could be extended to explicitly consider the update of puzzle freshness and how past puzzles affect current denial of service resilience.

This new framework for analyzing denial of service resilience can be applied in conjunction with other goals for key agreement protocols. For example, the JFK protocols [16] aim to offer privacy features: JFKi protects the initiator's identity and JFKr protects the responder's identity. Future work could involve designing denial-of-service-resilient protocols with similar privacy measures.

This model can also be applied to give denial of service resilience to other types of key agreement protocols, for example password-authenticated key agreement

protocols. Adapting this technique for use in IPsec or TLS would provide denial of service resilience in important Internet protocols.

Acknowledgements. This work was performed while the authors were at the University of Waterloo. D.S. was supported by an NSERC Canada Graduate Scholarship. The authors are grateful for the helpful advice of Alfred Menezes and Ian Goldberg.

References

1. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
2. Blake-Wilson, S., Johnson, D., Menezes, A.: Key agreement protocols and their security analysis. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355. Springer, Heidelberg (1997)
3. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 453–474. Springer, Heidelberg (2001)
4. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
5. LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
6. Boyd, C., Mathuria, A.: Protocols for Authentication and Key Establishment. Springer, Heidelberg (2003)
7. Wang, X., Reiter, M.: Defending against denial-of-service attacks with puzzle auctions. In: Proc. 2003 IEEE Symposium on Security and Privacy (SP 2003), pp. 78–92. IEEE Press, Los Alamitos (2003)
8. Aura, T., Nikander, P.: Stateless connections. In: Han, Y., Okamoto, T., Qing, S. (eds.) ICICS 1997. LNCS, vol. 1334, pp. 87–97. Springer, Heidelberg (1997)
9. Meadows, C.: A formal framework and evaluation method for network denial of service. In: Proc. 1999 IEEE Computer Security Foundations Workshop (CSFW), vol. 4, IEEE Computer Society Press, Los Alamitos (1999)
10. Gong, L., Syverson, P.: Fail-stop protocols: An approach to designing secure protocols. In: Proceedings of the 5th IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-5), pp. 44–55 (September 1995)
11. Diffie, W., van Oorschot, P., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2(2), 107–125 (1992)
12. Smith, J., Gonzalez-Nieto, J., Boyd, C.: Modelling denial of service attacks on JFK with Meadows's cost-based framework. In: Buyya, R., Ma, T., Safavi-Naini, R., Steketee, C., Susilo, W. (eds.) Proc. 4th Australasian Information Security Workshop – Network Security (AISW-NetSec) 2006. CRPIT, vol. 54, pp. 125–134. Australian Computer Society (2006)
13. Karn, P., Simpson, W.A.: Photuris: Session-key management protocol, RFC 2522 (March 1999)

14. Krawczyk, H.: SIGMA: The 'SIGn-and-MAc' approach to authenticated Diffie-Hellman and its use in the IKE protocols. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 400–425. Springer, Heidelberg (2003)
15. Kaufman, C.: Internet Key Exchange (IKEv2) protocol, RFC 4306 (December 2005)
16. Aiello, W., Bellovin, S.M., Blaze, M., Canetti, R., Ioannidis, J., Keromytis, A.D., Reingold, O.: Just Fast Keying: Key agreement in a hostile Internet. ACM Transactions on Information and System Security 7(2), 1–30 (2004)
17. Mao, W., Paterson, K.G.: On the plausible deniability feature of Internet protocols (manuscript, 2002)
18. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 139–147. Springer, Heidelberg (1993)
19. Back, A.: A partial hash collision based postage scheme (1997), http://www.hashcash.org/papers/announce.txt
20. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Proc. Internet Society Network and Distributed System Security Symposium (NDSS), pp. 151–165. Internet Society (1999)
21. Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. In: Proc. Internet Society Network and Distributed System Security Symposium (NDSS 2003). Internet Society (2003)
22. Waters, B., Juels, A., Halderman, J.A., Felten, E.W.: New client puzzle outsourcing techniques for DoS resistance. In: Proc. 11th ACM Conference on Computer and Communications Security (CCS), pp. 246–256. ACM, New York (2004)
23. Aura, T., Nikander, P., Leiwo, J.: DOS-resistant authentication with client puzzles. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2000. LNCS, vol. 2133, pp. 170–177. Springer, Heidelberg (2001)
24. Eddy, W.M.: TCP SYN flooding attacks and common mitigations, RFC 4987 (August 2007)
25. Jakobsson, M., Juels, A.: Proofs of work and bread pudding protocols. In: Preneel, B. (ed.) Proceedings of the IFIP TC6/TC11 Joint Working Conference on Secure Information Networks: Communications and Multimedia Security. IFIP Conference Proceedings, vol. 152, pp. 258–272. Kluwer Academic Publishers, Dordrecht (1999)
26. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Designs, Codes and Cryptography 46(3), 329–342 (2008)
27. Okamoto, T., Pointcheval, D.: The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001)
28. Moskowitz, R., Nikander, P., Jokela, P., Henderson, T.R.: Host Identity Protocol, Internet-Draft (February 2004)

A Commitment-Consistent Proof of a Shuffle

Douglas Wikström

CSC KTH Stockholm, Sweden
dog@csc.kth.se

Abstract. We introduce a pre-computation technique that drastically reduces the online computational complexity of mix-nets based on homomorphic cryptosystems.

More precisely, we show that there is a permutation commitment scheme that allows a mix-server to: (1) commit to a permutation and efficiently prove knowledge of doing so correctly in the offline phase, and (2) shuffle its input and give an extremely efficient commitment-consistent proof of a shuffle in the online phase.

We prove our result for a general class of shuffle maps that generalize all known types of shuffles, and even allows shuffling ciphertexts of different cryptosystems in parallel.

1 Introduction

Consider a situation where N senders $S_1, \ldots, S_N$ each have some input and wish to compute the sorted list of their inputs without revealing who submitted which message. A trusted party can do this by waiting until all senders have submitted some input, and then sort and output the list of all inputs. A protocol that emulates the trusted party is called a *mix-net* and the parties $M_1, \ldots, M_k$ that execute the protocol are referred to as *mix-servers*. As long as a certain fraction of the mix-servers are honest, the result should be correct and nobody should learn the correspondence between input ciphertexts and output messages. The obvious application for mix-nets is to conduct electronic elections, and this is also one of the applications Chaum [6] had in mind when he introduced mix-nets.

Many constructions of mix-nets are proposed in the literature, but few have provable security properties and many are actually flawed. The basic approach of all mix-nets with provable properties are based on ideas of Sako and Kilian [23]. The first rigorous definition of security was given by Abe and Imai [1], but they did not construct a scheme satisfying their construction. Wikström [25] gives the first definition of a universally composable (UC) mix-net, the first UC-secure construction, and also a more efficient UC-secure scheme [26]. An important building block in the construction of a mix-net is a so called *proof of a shuffle* that allows the mix-servers to prove that they follow the protocol. The first efficient proofs of shuffles were given by Neff [18] and Furukawa and Sako [13].

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 407–421, 2009.

1.1 Mix-Nets Based on Homomorphic Cryptosystems

Recall the mix-net of Sako and Kilian [23]. They present their scheme in terms of the El Gamal cryptosystem [14], but the idea works for any homomorphic cryptosystem.

A homomorphic cryptosystem $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ that allows threshold decryption is employed. A cryptosystem is said to be homomorphic if for every public key $pk \in \mathcal{PK}$, the plaintext space $\mathcal{M}_{pk}$, the randomness space $\mathcal{R}_{pk}$, and the ciphertext space $\mathcal{C}_{pk}$ are groups, and for every $m_0, m_1 \in \mathcal{M}_{pk}$ and $r_0, r_1 \in \mathcal{M}_{pk}$: $\mathsf{E}_{pk}(m_0, r_0)\mathsf{E}_{pk}(m_1, r_1) = \mathsf{E}_{pk}(m_0 m_1, r_0 r_1)$. A joint public key pk is generated somehow such that each mix-server holds a secret share of the corresponding secret key sk. Each sender S_i, holding a message m_i, computes a ciphertext $c_{0,i} = \mathsf{E}_{pk}(m_i)$, and then somehow submits it to the mix-servers. The mix-servers then take turns at re-encrypting and permuting these ciphertexts. Let $L_0 = (c_{0,1}, \ldots, c_{0,N})$ be the list of submitted ciphertexts. For $j = 1, \ldots, k$, M_j chooses a permutation π and $r_{j,i} \in \mathcal{R}_{pk}$ randomly, computes $c_{j,i} = c_{j-1,\pi(i)}\mathsf{E}_{pk}(1, r_{j,\pi(i)})$ for $i = 1, \ldots, N$, and then publishes $L_j = (c_{j,1}, \ldots, c_{j,N})$. In other words, each mix-server randomly re-encrypts each ciphertext and then outputs the resulting ciphertexts in random order. Then it proves, using a proof of a shuffle, that it formed L_j from L_{j-1} in this way. Finally, the mix-servers jointly threshold-decrypt L_k and output the resulting list of plaintexts. The idea is that since all mix-servers have randomly permuted the ciphertexts and the cryptosystem is assumed secure, it is infeasible to tell which plaintext corresponds to which original ciphertext in L_0.

The above description is simplified in that the senders submit homomorphic ciphertexts directly, which is not secure [22]. In a provably secure construction, the plaintexts of corrupted senders must be extractable by the simulator without the secret key of the cryptosystem. Until recently, all known submission schemes were either only heuristically secure, or involved costly interaction, but there is now a provably secure solution to this problem for several well known homomorphic cryptosystems [27].

Alternative Constructions. In the scheme of Furukawa et al. [12], each mix-server not only re-encrypts and permutes its input, but also partially decrypts it. As a result, the final list L_k essentially contains the plaintexts and no joint decryption step is needed. In the scheme of Wikström [26], re-encryption is also eliminated entirely, i.e., each mix-server only partially decrypts and permutes its input. In a preliminary unpublished version of Neff [18] a proof of a shuffle for the first type of mix-net is described as well [19]. These schemes have special advantages over the above, but do not lend themselves well to pre-computation, since partial decryption must be done sequentially.

Very few other approaches to constructing mix-nets have any provable security properties [16] and several are actually flawed [1,9,24].

1.2 Previous Work On Improving Efficiency

There are more or less obvious techniques that can be used to reduce the computational complexity of a mix-net. If a threshold below k is used for the decryption

key, then all mix-servers do not need to take part in the mixing process. In the execution of a public-coin honest verifier proof of knowledge the random challenge of the honest verifier must be generated jointly by the mix-servers, which is costly. But if unpredictability suffices, then longer challenges can be extracted from a random seed using a PRG. Pre-computation can also be used in the coin-flipping protocols. The re-encryption factors can also be pre-computed and batch proof techniques [4] can be used to reduce the complexity of the proofs of correctness needed during joint decryption.

If such optimizations and pre-computations are used, the main computational cost lies in the proofs of shuffles. Thus, most previous work on reducing the complexity, e.g. [12,13,18,15,26], focus on reducing the complexity of a particular proof of a shuffle. Some parts of these proofs can easily be pre-computed as well.

An alternative approach is used by Adida and Wikström [3], who show that when the number of senders is relatively small, ideas from homomorphic election schemes [5] can be used to construct a mix-net where the online phase only requires decryption of a single ciphertext. The public-key obfuscated shuffle of Adida and Wikström [2] may also be viewed as a form of pre-computation, but their goal is not improved efficiency. In fact, their scheme is quite inefficient.

1.3 Our Contribution

We show how to split a proof of a shuffle into two protocols. The first protocol is used by a mix-server in the offline phase to prove knowledge of how to open a commitment to a permutation. The second protocol is used by a mix-server in the online phase to prove that it uses the permutation it committed to also during shuffling.

The first protocol is almost as efficient as the known proofs of shuffles; in fact it can be constructed from these, e.g., [13,15,18,26]. Even without any standard optimization techniques such as simultaneous exponentiations, the computational complexity of the second protocol is half an exponentiation per sender in the El Gamal case and has similar properties for other cryptosystems. Thus, our pre-computation technique reduces the online computational complexity of virtually all mix-nets.

We also show that all known types of shuffles are instances of a generalized shuffle, where some homomorphic map $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \rightarrow \mathcal{C}_{pk}$ is applied to each ciphertext and randomizer pair, and the resulting ciphertexts are permuted. In fact, we prove our results for this generalized shuffle. The generality of our result immediately gives that ciphertexts can be shuffled in parallel. Even ciphertexts of different cryptosystems can be shuffled in parallel, and distinct homomorphic maps can be used for ciphertexts of different cryptosystems.

The inspiration of this work comes from both Neff [18] and Furukawa and Sako [13]. Neff writes as follows about his "simple shuffle": "A single instance of this proof can be constructed to essentially 'commit' a particular permutation", but we are unable to derive our results starting from his "commitment". On the other hand, the Pedersen permutation commitment scheme used implicitly in the proof of a shuffle of Furukawa and Sako is perfectly suitable for constructing a fast commitment-consistent proof of a shuffle.

1.4 Notation

Natural numbers and integers are denoted by $\mathbb{N}$ and $\mathbb{Z}$ respectively. The ring of integers modulo n is denoted by $\mathbb{Z}_n$, $\mathbb{Z}_n^*$ denotes its multiplicative group, and SQ_n denotes the subgroup of squares in $\mathbb{Z}_n^*$. We use κ as the main security parameter, but also introduce several related parameters, e.g., the bit-size of challenges κ_c. We identify the set of κ-bit strings and the set of positive integers in $[0, 2^\kappa - 1]$ when convenient. A function $\epsilon(\kappa)$ is negligible if for every constant c and sufficiently large κ it holds that $\epsilon(\kappa) < \kappa^{-c}$. A function $f(\kappa)$ is overwhelming if $1 - f(\kappa)$ is negligible. We denote the set of N-permutations by $\mathbb{S}_N$.

The Discrete Logarithm (DL) assumption for a group G_q with generator g states that given a random element $y \in G_q$, it is infeasible to compute x such that $y = g^x$. The decision Diffie-Hellman (DDH) assumption states that when $x, y, r \in \mathbb{Z}_q$ are randomly chosen, then it is infeasible to distinguish the distributions of (g^x, g^y, g^{xy}) and (g^x, g^y, g^r). See full version for formal definitions.

We view a commitment scheme as consisting of a parameter generation algorithm Gen and a deterministic commitment algorithm Com. On input 1^κ, Gen outputs a parameter ck which defines a message set $\mathcal{M}_{ck}$, a polynomially sampleable randomness space $\mathcal{R}_{ck}$, and a commitment space $\mathcal{K}_{ck}$. We write $\mathcal{CK}$ for the set of commitment parameters. On input $ck \in \mathcal{CK}$, $m \in \mathcal{M}_{ck}$, and $r \in \mathcal{R}_{ck}$, Com outputs a commitment. To open a commitment the message and randomness is revealed.

We write $\mathcal{CS} = (\mathsf{Kg}, \mathsf{E}, \mathsf{D})$ for a homomorphic cryptosystem and $\mathcal{M}_{pk}$, $\mathcal{R}_{pk}$, and $\mathcal{C}_{pk}$ for the abealian groups of messages, randomness, and ciphertexts defined by a public key pk. We let $\mathcal{PK}$ denote the set of all public keys. A homomorphic cryptosystem satisfies $\mathsf{E}_{pk}(m_1, r_1)\mathsf{E}_{pk}(m_2, r_2) = \mathsf{E}_{pk}(m_1 m_2, r_1 r_2)$ for every $pk \in \mathcal{PK}$, $m_1, m_2 \in \mathcal{M}_{pk}$, and $r_1, r_2 \in \mathcal{R}_{pk}$.

We denote the set $\{1, \ldots, l\}$ by $[l]$ and sometimes denote a list of elements $(a_1, \ldots, a_l)$ by $a_{[l]}$.

Throughout we assume that the order of the largest cyclic subgroup of $\mathcal{C}_{pk}$, and the order of any groups on which we base our commitment schemes, are bounded by 2^κ.

2 Background and Informal Description

Before we give details, it is worthwhile to recall some properties of batch proofs of discrete logarithms and proofs of shuffles. We also give a brief informal description of our commitment-consistent proof of a shuffle.

Batch Proofs. Consider a setting where many group elements $y_1, \ldots, y_N$ in some prime order group G_p with generator g are given, and the prover knows $x_i \in \mathbb{Z}_p$ such that $y_i = g_i^{x_i}$. It is expensive to prove knowledge of each logarithm x_i independently, but the use of batching [4] decreases this cost substantially as the following example shows.

1. Verifier picks $e_1, \ldots, e_N \in \mathbb{Z}_p$ randomly and hands them to prover.
2. Both parties compute $y = \prod_{i=1}^N y_i^{e_i}$.

3. Prover shows that it knows the logarithm w such that $y = g^w$ using a standard honest verifier zero-knowledge proof of knowledge.

The reason that this is a proof of knowledge is that the extractor may rewind the prover to the first step several times until it has found N linearly independent vectors $\overline{e}_j = (e_{j,1}, \ldots, e_{j,N})$ in $\mathbb{Z}_p^N$ for $j = 1, \ldots, N$ and extracted logarithms $w_1, \ldots, w_N$ such that $\prod_{i=1}^N y_i^{e_{j,i}} = g^{w_j}$. Note that linear independence imply that for every $l = 1, \ldots, N$ there are $d_{l,j}$ such that $\sum_{j=1}^N d_{l,j}\overline{e}_j$ is the lth standard unit vector in $\mathbb{Z}_p^N$. This gives

$$y_l = \prod_{j=1}^N \left(\prod_{i=1}^N y_i^{e_{j,i}} \right)^{d_{l,j}} = \prod_{j=1}^N (g^{w_j})^{d_{j,i}} = g^{\sum_{j=1}^N d_{l,j} w_j} \quad ,$$

which means that the logarithm of every individual element y_l can be computed as $x_l = \sum_{j=1}^N d_{l,j} w_j$. We remark that the components of the vectors can be chosen randomly in $\{0,1\}^{\kappa_e}$ for a κ_e much smaller than κ. From now on we use κ_e to denote the bit-size of components of random vectors as the above. Another important observation, used to reduce the need for jointly generated randomness when the honest verifier is implemented jointly by several parties, is that it suffices that the vectors are unpredictable, e.g., the verifier may instead choose a random seed z for a PRG, hand it to the prover, and define $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$.

Proofs of Shuffles. Due to space restrictions, we can not go into the details of any particular proof of a shuffle, but we can explain one of the ideas that appear in different forms in all known efficient schemes.

Consider a homomorphic cryptosystem such that the order of every non-trivial element in $\mathcal{C}_{pk}$ equals a prime p. Given are a public key pk and ciphertexts $(c_1, \ldots, c_N)$ and $(c'_1, \ldots, c'_N)$ that are related by $c'_i = c_{\pi(i)}\mathsf{E}_{pk}(1, r_{\pi(i)})$ for some permutation π and randomness $r_1, \ldots, r_N$.

A key observation, first made by Neff [18] and Furukawa and Sako [13], is that batch proofs are in some sense invariant under permutation and that this means that we can use batch techniques to construct an efficient proof of a shuffle. The idea can be described as follows, where we use a PRG to expand a seed into an unpredictable vector.

1. $\mathcal{V}$ picks a seed $z \in \{0,1\}^\kappa$ randomly and hands it to $\mathcal{P}$.
2. Both parties compute $c = \prod_{i=1}^N c_i^{e_i}$, where $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$ and $e_i \in [0, 2^{\kappa_e} - 1]$.
3. $\mathcal{P}$ computes $c' = \prod_{i=1}^N (c'_i)^{e_{\pi(i)}}$, hands it to $\mathcal{V}$, and convinces $\mathcal{V}$ that it is formed correctly.
4. $\mathcal{P}$ proves knowledge of $r \in \mathcal{R}_{pk}$ such that $c' = c\mathsf{E}_{pk}(1, r)$.

Note that the linear independence argument used in the basic batch proof above carries over to the shuffle setting, despite that some of the exponents are permuted (see Proposition 3 in full version for details). The above description is simplified in that the prover must blind c' to avoid leaking knowledge. The

problem of convincing the verifier that the original exponents, re-ordered using a fixed permutation π, are used to form c' is non-trivial, and solved differently in the various proofs of shuffles. If we ignore the cost of Step 3, then the above protocol is very efficient.

2.1 Commitment-Consistent Proofs of Shuffles

We observe that we can design Step 3 in such a way that almost all of it can be moved to the offline phase. Generators $g_1, \ldots, g_N$ of a group G_p of prime order p are given as part of the setup of the proof of a shuffle, and it is assumed to be infeasible to compute any non-trivial relations among these (this follows from the DL assumption).

Suppose that each mix-server commits to a permutation π using Pedersen commitments [21] $(a_1, \ldots, a_N) = (g^{r_1} g_{\pi^{-1}(1)}, \ldots, g^{r_N} g_{\pi^{-1}(N)})$ for random $r_1, \ldots, r_N \in \mathbb{Z}_p$, and also proves knowledge of the r_i and π such that $(a_1, \ldots, a_N)$ was formed in this way. Then in the online phase the verifier can choose, and hand to the prover, a random seed $z \in \{0,1\}^\kappa$, set $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$, and compute

$$a = \prod_{i=1}^N a_i^{e_i} = \prod_{i=1}^N g^{r_i e_i} g_{\pi^{-1}(i)}^{e_i} = g^r \prod_{i=1}^N g_i^{e_{\pi(i)}} \quad ,$$

where $r = \sum_{i=1}^N r_i e_i$. Note that a is of a perfect form for executing a standard proof of knowledge of equal exponents. More precisely, we may now replace Step 3 above in the online phase by:

- Prover computes $c' = \prod_{i=1}^N (c'_i)^{e_{\pi(i)}}$ and hands it to the verifier.
- Prover proves knowledge of $r' \in \mathbb{Z}_p$ and $e'_1, \ldots, e'_N \in \{0,1\}^{\kappa_e}$ with

$$a = g^{r'} \prod_{i=1}^N g_i^{e'_i} \quad \text{and} \quad c' = \prod_{i=1}^N (c'_i)^{e'_i} \quad .$$

The above is simplified in that some blinding factors are missing. The proof of knowledge of the exponents $r', e'_1, \ldots, e'_N$, combined with the computational binding property of multi-base Pedersen commitments implies that $e'_i = e_{\pi(i)}$ for some permutation $\pi(i)$. The computational complexity of the above protocol is very low, since almost all exponents have very few bits also in the proof of knowledge of equal exponents.

3 A Commitment-Consistent Proof of a Shuffle

In this section we first give more details of the commitment scheme and explain how any of the known proofs of shuffles can be used to prove knowledge of an opening of the commitment to a permutation. Then we present the commitment-consistent proof of a shuffle.

3.1 Permutation Commitments

We formalize the property we need from the Pedersen commitments above. A permutation commitment should allow the committer to compute a commitment $\mathsf{Com}^\star(\pi)$ of a permutation π, but obviously any string commitment can be used to commit to a permutation. The special property of a permutation commitment is that if the *receiver* holds a list $(e_1, \ldots, e_N)$, it can transform the permutation commitment into a commitment $\mathsf{Com}^e(e_{\pi(1)}, \ldots, e_{\pi(N)})$, of another type, of the the list elements, but in order defined by π. Here κ_{com} denotes the maximal bit size of each component of a list commitment.

Definition 1. *Let* $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ *be a commitment scheme for* $\mathbb{S}_N$ *and let* $(\mathsf{Gen}^e, \mathsf{Com}^e)$ *be a commitment scheme for* $[0, 2^{\kappa_{com}} - 1]^N$*. The former is a* κ_{com}*-permutation commitment scheme of the latter if* $\mathsf{Gen}^\star = \mathsf{Gen}^e$ *and there exist deterministic polynomial time algorithms* Map *and* Rand *s.t. for every* $ck \in \mathcal{CK}$*,* $r^\star \in \mathcal{R}^\star_{ck}$*,* $\pi \in \mathbb{S}_N$ *and* $e = (e_1, \ldots, e_N) \in [0, 2^{\kappa_{com}} - 1]^N$

$$\mathsf{Map}_{ck}(\mathsf{Com}^\star_{ck}(\pi, r^\star), e) = \mathsf{Com}^e_{ck}((e_{\pi(1)}, \ldots, e_{\pi(N)}), \mathsf{Rand}(r^\star, e)) \ .$$

Construction 1 (Pedersen Commitment). The generation algorithm $\mathsf{Gen}^\star$ outputs random generators $g_1, \ldots, g_N \in G_q$, where G_q is a cyclic group of known order $q = \prod_{i=1}^{t} p_i$ with $p_i \geq 2^{\kappa_{com}}$. On input $\pi \in \mathbb{S}_N$ and $r_1, \ldots, r_N \in \mathbb{Z}_q$, the commitment algorithm $\mathsf{Com}^\star$ computes $a_i = g^{r_i} g_{\pi^{-1}(i)}$, and outputs $(a_1, \ldots, a_N)$. The parameter algorithm Gen^e is identical to $\mathsf{Gen}^\star$. On input $(e_1, \ldots, e_N) \in [0, 2^{\kappa_{com}} - 1]^N$ and $r \in \mathbb{Z}_q$, the algorithm Com^e computes $a = g^r \prod_{i=1}^{N} g_i^{e_i}$, and outputs a.

The idea of using (generalized) Pedersen commitments [21] to commit to permutations is not novel, e.g., it is used implicitly in [13], but the observation that a commitment of the first kind can be transformed into a commitment of the second kind seems new.

Proposition 1. *Both* $(\mathsf{Gen}^\star, \mathsf{Com}^\star)$ *and* $(\mathsf{Gen}^e, \mathsf{Com}^e)$ *of Construction 1 are perfectly hiding and computationally binding under the DL assumption. The former is a permutation commitment of the latter.*

The proof of the binding property is well known for prime order groups. A proof is given in the full version.

We will later make use of the following relation that corresponds to breaking a commitment scheme, i.e., finding two different ways to open a commitment.

Definition 2. *The relation* $\mathscr{R}^{twin}_{ck}$ *consists of all pairs* $\left(ck, (s_{[l]}, s_0, s'_{[l]}, s'_0)\right)$ *such that* $s_{[l]} \neq s'_{[l]}$ *and* $\mathsf{Com}^e_{ck}(s_{[l]}, s_0) = \mathsf{Com}^e_{ck}(s'_{[l]}, s'_0)$.

Suppose a committer produces a permutation commitment $a^\star$ and the verifier computes $a = \mathsf{Map}_{ck}(a^\star, (e_1, \ldots, e_N))$. Then we expect that the committer only can open a as $(e_{\pi(1)}, \ldots, e_{\pi(N)})$ for a *fixed* permutation π, i.e., if we repeat this procedure with different lists $(e_1, \ldots, e_N)$ the same permutation must be used

by the committer every time. We can not prove this, but it is easy to see that if it also can open $a^\star$ to a permutation π, then it must use this permutation every time. Recall that in our application, each mix-server proves knowledge of how to open $a^\star$ during the offline phase. Thus, if a witness for the following relation can be extracted in the online phase we reach a contradiction. This suffices to prove the overall security of a mix-net.

Definition 3. *The relation* $\mathscr{R}_{ck}^{perm}$ *consists of all* $\big(ck, (a^\star, s_{[N]}, s_0, s'_{[N]}, s'_0)\big)$ *such that* $\mathsf{Map}_{ck}(a^\star, s_{[N]}) = \mathsf{Com}^e_{ck}((s_{\pi(1)}, \ldots, s_{\pi(N)}), s_0)$, $\mathsf{Map}_{ck}(a^\star, s'_{[N]}) = \mathsf{Com}^e_{ck}((s'_{\pi'(1)}, \ldots, s'_{\pi'(N)}), s'_0)$, $\pi \neq \pi'$, *and* $s_i \neq s_j$ *and* $s'_i \neq s'_j$ *for all* $i \neq j$.

3.2 Proof of Knowledge of Opening

We now explain how we can construct, from any proof of a shuffle of El Gamal ciphertexts over a prime order group G_p, a proof of knowledge that a Pedersen permutation commitment indeed is a commitment to a permutation.

Definition 4. *The relation* $\mathscr{R}_{ck}^{open}$ *consists of all* $\big((ck, a^\star), (\pi, r^\star)\big)$ *such that* $a^\star = \mathsf{Com}^\star_{ck}(\pi, r^\star)$.

Protocol 1 (Proof of Knowledge of Correct Opening).
COMMON INPUT: Pedersen commitment parameters $g, g_1, \ldots, g_N \in G_p$ and a commitment $(a_1, \ldots, a_N) \in G_p^N$.
PRIVATE INPUT: Permutation $\pi \in \mathbb{S}_N$ and exponents $r_1, \ldots, r_N \in \mathbb{Z}_p$ such that $a_i = g^{r_i} g_{\pi^{-1}(i)}$.

1. $\mathcal{P}$ chooses $r'_i \in \mathbb{Z}_p$ and $h \in G_p$ randomly, computes $a'_i = g^{r'_i} a_i$ and $b_i = h^{r_i + r'_i}$, and hands $(a'_1, \ldots, a'_N)$ and $(h, b_1, \ldots, b_N)$ to $\mathcal{V}$.
2. $\mathcal{P}$ proves to $\mathcal{V}$ that it knows r'_i such that $a'_i = g^{r'_i} a_i$.
3. $\mathcal{P}$ and $\mathcal{V}$ view (h, g) as an El Gamal public key, and $\mathcal{P}$ uses its random commitment exponents $r_1 + r'_1, \ldots, r_N + r'_N$ to give a proof of a shuffle that the list $(b_1, a'_1), \ldots, (b_N, a'_N)$ is a re-encryption and permutation of the list of trivial ciphertexts $(1, g_1), \ldots, (1, g_N)$ using the public key (h, g), i.e., it proves that it knows some r''_i such that $(b_i, a_i) = (h^{r''_i}, g^{r''_i} g_{\pi^{-1}(i)})$.

Proposition 2. *The protocol inherits properties of the proof of a shuffle.*

1. *If the proof of a shuffle is public-coin, overwhelmingly (computationally) sound, and a proof of knowledge, then so is the protocol above.*
2. *If the proof of a shuffle is honest verifier (computationally under assumption A) zero-knowledge, then the above protocol is computationally zero-knowledge under the DDH assumption (and assumption A).*

A proof is given in the full version. Without the blinding exponent r'_i the protocol is not even computationally zero-knowledge, since the adversary could in principle know r_i. Some proofs of shuffles do not satisfy the standard computational versions of soundness, proof of knowledge, and zero-knowledge. In those cases

the correspondingly more complicated security properties are also inherited, but we use the above proposition for simplicity. Readers with deeper understanding of proofs of shuffles should note that the basic principles of any proof of a shuffle can be used directly to construct a more efficient protocol, but this is not our focus here. We stress that the above simple solution is presented for completeness and ease of presentation. It is non-trivial to extend the above result to groups of *composite* order such as those considered in Construction 1.

3.3 Proof of Knowledge of Equal Exponents

Recall from our sketch in Section 2.1 that in our commitment-consistent proof of a shuffle, the prover essentially hands the product $\prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}$ to the verifier and shows that the exponents used are those committed to in a commitment $\mathsf{Com}^e(e_{\pi(1)},\ldots,e_{\pi(N)})$. More precisely, we assume that: $\{h_1,\ldots,h_k\}$ is a generator set of the group $\mathcal{C}_{pk}$ of ciphertexts, ck is a commitment parameter, and that the prover hands $\prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}$ to the verifier in blinded form, i.e., it hands $\left(\mathsf{Com}^e_{ck}(s_{[k]},s_0),\prod_{i=1}^{k}h_i^{s_i}\prod_{i=1}^{N}(c'_i)^{e_{\pi(i)}}\right)$ to the verifier for random exponents $s_{[k]}$ (and $s_0 \in \mathcal{R}_{ck}$), and then proves that it knows all of these exponents and that they are consistent with the exponents committed to in $\mathsf{Com}^e_{ck}((e_{\pi(1)},\ldots,e_{\pi(N)}),e_0)$ for some $e_0 \in \mathcal{R}_{ck}$. Thus, we construct a protocol for the following relation.

Definition 5. *From a scheme* $(\mathsf{Gen}^e,\mathsf{Com}^e)$ *for* $[0,2^{\kappa_{com}}-1]^N$*, a commitment parameter* ck *output by* Gen^e*, and a public key* $pk \in \mathcal{PK}$ *we define* $\mathscr{R}^{eq}_{ck,pk}$ *to consist of all* $\left((pk,ck,h_{[k]},c_{[N]},a,b_1,b_2),(e_0,e_{[N]},s_0,s_{[N]})\right)$ *satisfying* $a=\mathsf{Com}^e_{ck}(e_{[N]},e_0)$*,* $b_1=\mathsf{Com}^e_{ck}(s_{[k]},s_0)$*, and* $b_2=\prod_{i=1}^{k}h_i^{s_i}\prod_{i=1}^{N}c_i^{e_i}$*.*

If the largest cyclic subgroup of $\mathcal{C}_{pk}$ has order $q=\prod_{i=1}^{t}p_i$ with $p_i \geq 2^{\kappa_c}$, and a group G_q of order q is available for which the DL problem is hard, then a sigma protocol with the challenge chosen from $[0,2^{\kappa_c}-1]$, can be constructed using fairly standard methods. For completeness we give such a protocol in the full version.

Otherwise, we can either use Pedersen commitments over some prime order group G_p and use a proof of equal exponents over groups of different orders using a Fujisaki-Okamoto commitment [11] as a "bridge", or we can replace the permutation commitment by a corresponding Fujisaki-Okamoto commitment directly. It is not hard to derive a shuffle of such commitments from Wikström's shuffle [26]. The drawback of using Fujisaki-Okamoto commitments is that they are based on the use of an RSA modulus, and such moduli are costly to generate in a distributed setting. We detail both solutions in the the full version.

3.4 Shuffle-Friendly Maps

To *randomly shuffle* a list of homomorphic ciphertexts $(c_1,\ldots,c_N)$ usually means that each ciphertext is randomly re-encrypted and the resulting ciphertexts randomly permuted, but there are other possible shuffles. For the El Gamal cryptosystem, one can also partially decrypt during shuffling [12], or if a special key

set-up is used one can avoid random re-encryption entirely [26]. There are also at least two types of shuffles of (variants of) Paillier [20] ciphertexts. A careful look at these shuffles reveal that they are all defined by evaluating a homomorphic map and permuting the result.

Definition 6. *A map ϕ_{pk} is shuffle-friendly for a public key $pk \in \mathcal{PK}$ of a homomorphic cryptosystem if it defines a homomorphic map $\phi_{pk} : \mathcal{C}_{pk} \times \mathcal{R}_{pk} \to \mathcal{C}_{pk}$.*

Example 1. Using the El Gamal cryptosystem over a group G_p with public key $pk = (g, y)$, where $y = g^x$ and x is the secret key, we have $\mathcal{M}_{pk} = G_p$, $\mathcal{R}_{pk} = \mathbb{Z}_p$, and $\mathcal{C}_{pk} = G_p \times G_p$. Then $\phi_{(g,y)}((u, v), r) = (g^r u, y^r v)$ describes re-encryption when $r \in \mathbb{Z}_p$ is randomly chosen. If $y_i = g^{x_i}$, $y = y_1 y_2 y_3$, and $x = x_1 + x_2 + x_3$, then $\phi^{x_1}_{(g,y)}((u, v), r) = (g^r u, (y/y_1)^r u^{-x_1} v)$ denotes partial decryption and re-encryption using the secret share x_1 and randomness r. The decryption shuffle in [26] can be described similarly.

Example 2. Using the Paillier cryptosystem with a public key $pk = n$ consisting of a random RSA modulus, we have $\mathcal{M}_{pk} = \mathbb{Z}_n$, $\mathcal{R}_{pk} = \mathbb{Z}^*_n$, and $\mathcal{C}_{pk} = \mathbb{Z}^*_{n^2}$ with encryption defined by $\mathsf{E}_{pk}(m, r) = (1 + n)^m r^n \bmod n^2$. Re-encryption is then defined by $\phi_n(c, r) = cr^n \bmod n^2$.

Suppose we wish to prove that a ciphertext c' is the result of invoking a particular shuffle-friendly map ϕ_{pk} on another ciphertext c. If the shuffle-friendly map ϕ_{pk} is public, e.g., it represents re-encryption, then what is needed is a proof that there exists some randomness r such that $\phi_{pk}(c, r) = c'$. If the shuffle-friendly map itself is not public, e.g., re-encryption and partial decryption, then the map ϕ_{pk} must then be defined by some hidden parameters. Without loss we assume that the map is defined by some relation to the public key. In the typical cases, the public key defines a secret key and the shuffle-friendly map is defined by the secret key. We consider a situation where the output ciphertext c' is committed to as $(\mathsf{Com}^e_{ck}((s_1, \ldots, s_k), s_0), c' \prod_{i=1}^k h_i^{s_i})$, and define a relation for a shuffle-friendly map as follows.

Definition 7 (Shuffle-Friendly Relation). *Let $pk \in \mathcal{PK}$, let ϕ_{pk} be a shuffle-friendly map for pk and let ck be a commitment parameter. We define $\mathscr{R}^{map}_{\phi_{pk}}$ to consist of all pairs $\big((pk, ck, h_{[k]}, c, b_1, b_2), (r, s_0, s_{[k]})\big)$ such that $b_1 = \mathsf{Com}^e_{ck}(s_{[k]}, s_0)$ and $b_2 = \phi_{pk}(c, r) \prod_{i=1}^k h_i^{s_i}$.*

Example 3 (Example 1 continued). Note that $\mathcal{C}_{pk} = G_p \times G_p$ is generated by $h_1 = (g, 1)$ and $h_2 = (1, g)$ with component-wise multiplication. If we consider a re-encryption and permutation shuffle and use Pedersen commitments over the group G_p with parameter $ck = (g_1, g_2)$, then the relation consists of all pairs $\big(((g, y), (g_1, g_2), (u, v), b_1, b_2), (r, s_0, s_1, s_2)\big)$ such that $b_1 = g^{s_0} g_1^{s_1} g_2^{s_2}$ and $b_2 = h_1^{s_1} h_2^{s_2} (g^r u, y^r v)$.

For the typical shuffle-friendly maps of the El Gamal and Paillier cryptosystems, it is well known how to construct sigma protocols [7] for the corresponding shuffle-friendly relation using standard methods. We give some examples in the full version.

3.5 Details of the Commitment-Consistent Proof of a Shuffle

Next we give a detailed description of the protocol that allows a mix-server to prove in the online phase that it re-encrypted and permuted its input and that the permutation used is the same permutation it committed to in the offline phase. We denote by κ_r a parameter that decides how well the commitments hide the committed values.

The two subprotocols can be executed in parallel and the second step of the protocol can be combined with the first move of the combined subprotocols.

Protocol 2 (Commitment-Consistent Proof of a Shuffle).
COMMON INPUT: A public key pk of a cryptosystem $\mathcal{CS}$, a generating set $\{h_1, \ldots, h_k\}$ of $\mathcal{C}_{pk}$, a commitment parameter ck, a permutation commitment $a^\star \in \mathcal{K}^{\pi}_{ck}$, ciphertexts $(c_1, \ldots, c_N) \in \mathcal{C}^N_{pk}$, and $(c'_1, \ldots, c'_N) \in \mathcal{C}^N_{pk}$.
PRIVATE INPUT: Permutation $\pi \in \mathbb{S}_N$, $s^\star \in \mathcal{R}^\star_{ck}$ and $r_1, \ldots, r_N \in \mathcal{R}_{pk}$ such that $a^\star = \mathsf{Com}^\star_{ck}(\pi, s^\star)$, and $c'_i = \phi_{pk}(c_{\pi(i)}, r_{\pi(i)})$.

1. $\mathcal{V}$ chooses a seed $z \in \{0,1\}^\kappa$ randomly and hands it to $\mathcal{P}$. Then both parties set $(e_1, \ldots, e_N) = \mathsf{PRG}(z)$, where $e_i \in \{0,1\}^{\kappa_e}$, and computes $a = \mathsf{Map}_{ck}(a^\star, (e_1, \ldots, e_N))$.
2. $\mathcal{P}$ chooses $t_0 \in \mathcal{R}_{ck}$ and $t_1, \ldots, t_k \in [0, 2^{\kappa+\kappa_r} - 1]$ randomly, and computes and hands to $\mathcal{V}$
$$b_1 = \mathsf{Com}^e_{ck}((t_1, \ldots, t_k), t_0) \text{ and } b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \prod\nolimits_{i=1}^{N} (c'_i)^{e_{\pi(i)}} \ .$$
3. $\mathcal{P}$ proves, using a proof of equal exponents, that it knows exponents $t_0, \ldots, t_k$, $(e'_1, \ldots, e'_N)$ (computed as $(e_{\pi(1)}, \ldots, e_{\pi(N)})$), and e_0 (computed as $\mathsf{Rand}(s^\star, (e_1, \ldots, e_N))$) such that
$$b_1 = \mathsf{Com}^e_{ck}((t_1, \ldots, t_k), t_0) \ , \ b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \prod\nolimits_{i=1}^{N} (c'_i)^{e'_i} \ , \text{ and}$$
$$a = \mathsf{Com}^e_{ck}((e'_1, \ldots, e'_N), e_0) \ .$$
4. $\mathcal{P}$ proves, using a proof of a shuffle map, that it knows exponents $t_0, \ldots, t_k$ and r (computed as $\prod_{i=1}^{N} r_i^{e_i}$) such that
$$b_1 = \mathsf{Com}^e_{ck}((t_1, \ldots, t_k), t_0) \text{ and } b_2 = \prod\nolimits_{i=1}^{k} h_i^{t_i} \phi_{pk}\left(\prod\nolimits_{i=1}^{N} c_i^{e_i}, r\right) \ .$$

Note that the protocol and the proposition below are quite general; they apply for all the usual homomorphic cryptosystems, any shuffle-friendly map, and any number of ciphertexts shuffled in parallel (this is considered as a separate case in [18]). It even applies to mixed settings where ciphertexts from different cryptosystems are shuffled in parallel. To state the security properties of the protocol we need to define a relation that captures a shuffle.

Definition 8. *Let $pk \in \mathcal{PK}$, let ϕ_{pk} be a shuffle-friendly map for pk. Then we define the shuffle relation $\mathscr{R}^{shuf}_{\phi_{pk}}$ to consist of all pairs of the form $\big((pk, c_{[N]}, c'_{[N]}), (\pi, r_{[N]})\big)$ with $c'_i = \phi_{pk}(c_{\pi(i)}, r_{\pi(i)})$.*

In the proposition we consider the relation $\mathscr{R}^{shuf}_{\phi_{pk}} \vee \mathscr{R}^{twin}_{ck} \vee \mathscr{R}^{perm}_{ck}$. In general, for two relations $\mathscr{R}_1$ and $\mathscr{R}_2$, the relation $\mathscr{R}_1 \vee \mathscr{R}_2$ denotes the relation consisting of all pairs $((x_1, x_2), w)$ such that $(x_1, w) \in \mathscr{R}_1$ or $(x_2, w) \in \mathscr{R}_2$.

Proposition 3. *Let the subprotocols be overwhelmingly complete sigma protocols for the relations $\mathscr{R}^{eq}_{ck,pk} \vee \mathscr{R}^{twin}_{ck}$ and $\mathscr{R}^{map}_{\phi_{pk}}$ respectively, and let the commitment scheme be statistically hiding.*

Then for every $pk \in \mathcal{PK}$ and $ck \in \mathcal{CK}$, the protocol is an sound public-coin honest verifier statistical zero-knowledge proof of the relation $\mathscr{R}^{shuf}_{\phi_{pk}} \vee \mathscr{R}^{twin}_{ck} \vee \mathscr{R}^{perm}_{ck}$, and overwhelmingly complete for witnesses of $\mathscr{R}^{shuf}_{\phi_{pk}}$.

It is a proof of knowledge with negligible knowledge error of a string w such that $\mathscr{R}^{shuf}_{\phi_{pk}}\big((pk, c_{[N]}, c'_{[N]}), (w, r_{[N]})\big) = 1$, $\mathscr{R}^{twin}_{ck}(ck, w) = 1$, or $\mathscr{R}^{perm}_{ck}(ck, w) = 1$, is satisfied for some randomness $r_{[N]} \in \mathcal{R}_{pk}$, where we use the notation for inputs to the protocol as defined above.

Remark 1. It is observed in [26] that it does not suffice that a proof of a re-encryption and permutation shuffle is sound to be used in a provably secure mix-net. The permutation used by the mix-server to shuffle must also be extractable. However, extracting the *permutation* suffices.

A proof of the proposition is given in the full version. The basic idea is explained already in Section 2.1, except that in the general case the order q of the maximal cyclic subgroup of $\mathcal{C}_{pk}$ may not be prime or may even be unknown. Note that if q is not prime, then the "random vectors" are in fact defined over a ring and not over a field, and consequently they are not vectors at all. Thus, not all elements are invertible, which potentially is a problem when trying to find a linear combination of the "random vectors" equal to any standard unit vector, which is needed to extract a witness. Since we assume that all factors of the order of $\mathcal{C}_{pk}$ are large and all elements that must be inverted are random, this is not a problem and a witness can be extracted. However, if it is infeasible to compute the factorization of the order of $\mathcal{C}_{pk}$, or if the order itself is unknown, then this seems difficult. Fortunately, it suffices for the overall security of the mix-net that only the permutation can be extracted.

4 Application To Mix-Nets

At this point the reader should be comfortable with the idea that a proof of a shuffle can be split into a relatively costly offline part (Protocol 1) and a very efficient online part (Protocol 2), but how exactly do they fit into a mix-net?

Below we give a brief informal description of a mix-net based on the El Gamal cryptosystem over a group G_p of prime order p. This illustrates how our protocols are used and gives an idea of the complexity of a complete mix-net using our approach.

Offline Phase

1. The mix-servers, $M_1, \ldots, M_k$, run a distributed key generation protocol to generate a joint public key (g, y) such that the corresponding secret key x, with $y = g^x$, is secret shared among the mix-servers.

2. M_j chooses $r_{j,i} \in \mathbb{Z}_p$ randomly and computes $(g^{r_{j,i}}, y^{r_{j,i}})$ for $i = 1, \ldots, N$.
3. M_j chooses a random permutation $\pi_j \in \mathbb{S}_N$, publishes a permutation commitment $a_j^\star = \mathsf{Com}^\star(\pi_j)$, and proves knowledge of the committed permutation using Protocol 1 (and verifies the proofs of knowledge of all other mix-servers).

Online Phase

4. S_i chooses $r_i \in \mathbb{Z}_p$ randomly, computes $(u_{0,i}, v_{0,i}) = \mathsf{E}_{(g,y)}(m_i, r_i)$, where $m_i \in \mathbb{Z}_p$ is its message, and publishes this ciphertext.
5. Let $L_0 = (u_{0,i}, v_{0,i})_{i=1}^N$ be the input ciphertexts. For $l = 1, \ldots, k$:
   (a) If $l = j$, then M_j computes $(u_{j,i}, v_{j,i}) = (g^{r_{j,i}} u_{j-1,\pi_j(i)}, y^{r_{j,i}} v_{j-1,\pi_j(i)})$, publishes $L_j = (u_{j,i}, v_{j,i})_{i=1}^N$, and proves using Protocol 2 that L_{j-1} and L_j are consistent with $a_j^\star$.
   (b) If $l \neq j$, then M_j verifies the proof of M_l, i.e., that L_{l-1} and L_l are consistent with $a_l^\star$.
6. The mix-servers perform a threshold decryption of L_k using their shares of x and output the list of plaintexts $(m_{\pi(1)}, \ldots, m_{\pi(N)})$, where $\pi = \pi_k \cdots \pi_1$.

The random challenges needed in the subprotocols are generated jointly using a coin-flipping protocol over a broadcast channel or bulletin board. Thus, all verifiers jointly either accept or reject proofs. It is natural to ask why the security property of our commitment-consistent proof suffices, since it is sound for $\mathscr{R}_{\phi_{pk}}^{shuf} \vee \mathscr{R}_{ck}^{twin} \vee \mathscr{R}_{ck}^{perm}$ and not for $\mathscr{R}_{\phi_{pk}}^{shuf}$. This follows from the proof of knowledge property. For any successful prover there exists an extractor that outputs: a valid permutation π used to shuffle, a witness for $\mathscr{R}_{ck}^{twin}$, or a witness for $\mathscr{R}_{ck}^{perm}$. The second type of output directly contradicts the security of the commitment scheme. The third type of output combined with knowledge of how to open $a_j^\star$ (such an opening can be extracted during the offline phase), also contradicts the security of the commitment scheme. Thus, in a simulation the extractor outputs the permutation with overwhelming probability, which suffices to prove the overall security of the mix-net.

Depending on the secret sharing threshold, all mix-servers may not need to shuffle the ciphertexts, and there are obvious ways to avoid the assumption that all senders submit an input. Many details are of course missing from the above description, but in the El Gamal case all subprotocols missing from the description are available. Distributed key generation can be done using Feldman and Pedersen secret sharing [10,21]. The submission of inputs must allow extraction of the plaintexts of corrupt senders without using the secret key of the cryptosystem. This can be done [27] based on the Cramer-Shoup cryptosystem [8] in such a way that each mix-server essentially pays the cost of checking the validity of N Cramer-Shoup ciphertexts. Batch techniques [4] can be used to reduce this further if most ciphertexts are expected to be valid, and validity checks can be done concurrently with receiving new ciphertexts. Random challenges can be generated using Pedersen verifiable secret sharing [21]. The sharing phase of many coins can be pre-computed, but since we only need a

small number of bits in each challenge this type of optimization does not give much. Finally, during threshold decryption each mix-server must exponentiate N group elements to decrypt, but proving that this was done correctly can be done using batch proofs [4]. To summarize, the online running time of the mix-net is roughly the time to: validate N Cramer-Shoup ciphertexts, run the prover or verifier of k commitment-consistent proofs of shuffles of lists of ciphertexts of length N, decrypt N El Gamal ciphertexts, and prove or verify correctness of joint decryption, which is done using a batch proof.

Recall that κ_e denotes the bit-size of elements in random "vectors", κ_c denotes the bit-size of challenges, and κ_r decides the statistical error in simulations and also the completeness of our subprotocols. For practical security parameters, e.g., $\kappa = 1024$, $\kappa_e = \kappa_c = 80$ and $\kappa_r = 20$, we estimate the complexity of our protocol to $N/2$ square-and-multiply exponentiations. This can be reduced by a factor of $1/5$ if simultaneous exponentation [17] is used, giving a complexity corresponding to $N/10$ square-and-multiply exponentations (see full version for details).

Thus, our commitment-consistent proof of a shuffle is several times faster in the online phase than any of the known proofs of shuffles. As far as we know this makes our mix-net faster in the online phase than any previous mix-net.

Acknowledgments

We thank Jun Furukawa and Andy Neff for helpful comments.

References

1. Abe, M., Imai, H.: Flaws in some robust optimistic mix-nets. In: Safavi-Naini, R., Seberry, J. (eds.) ACISP 2003. LNCS, vol. 2727, pp. 39–50. Springer, Heidelberg (2003)
2. Adida, B., Wikström, D.: How to shuffle in public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
3. Adida, B., Wikström, D.: Offline/online mixing. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 484–495. Springer, Heidelberg (2007)
4. Bellare, M., Garay, J.A., Rabin, T.: Batch verification with applications to cryptography and checking. In: Lucchesi, C.L., Moura, A.V. (eds.) LATIN 1998. LNCS, vol. 1380, pp. 170–191. Springer, Heidelberg (1998)
5. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: 26th ACM Symposium on the Theory of Computing (STOC), pp. 544–553. ACM Press, New York (1994)
6. Chaum, D.: Untraceable electronic mail, return addresses and digital pseudo-nyms. Communications of the ACM 24(2), 84–88 (1981)
7. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)

8. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (1998)
9. Desmedt, Y., Kurosawa, K.: How to break a practical MIX and design a new one. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 557–572. Springer, Heidelberg (2000)
10. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 427–438. IEEE Computer Society Press, Los Alamitos (1987)
11. Fujisaki, E., Okamoto, T.: Statistical zero knowledge protocols to prove modular polynomial relations. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 16–30. Springer, Heidelberg (1997)
12. Furukawa, J., Miyauchi, H., Mori, K., Obana, S., Sako, K.: An implementation of a universally verifiable electronic voting scheme based on shuffling. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 16–30. Springer, Heidelberg (2003)
13. Furukawa, J., Sako, K.: An efficient scheme for proving a shuffle. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 368–387. Springer, Heidelberg (2001)
14. El Gamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory 31(4), 469–472 (1985)
15. Groth, J.: A verifiable secret shuffle of homomorphic encryptions. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 145–160. Springer, Heidelberg (2003)
16. Jakobsson, M., Juels, A., Rivest, R.: Making mix nets robust for electronic voting by randomized partial checking. In: 11th USENIX Security Symposium, pp. 339–353. USENIX (2002)
17. Menezes, A., Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
18. Neff, A.: A verifiable secret shuffle and its application to e-voting. In: 8th ACM Conference on Computer and Communications Security (CCS), pp. 116–125. ACM Press, New York (2001)
19. Neff, A.: Private communication (May 2008)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
21. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
22. Pfitzmann, B.: Breaking an efficient anonymous channel. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 332–340. Springer, Heidelberg (1995)
23. Sako, K., Killian, J.: Reciept-free mix-type voting scheme. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 393–403. Springer, Heidelberg (1995)
24. Wikström, D.: Five practical attacks for "optimistic mixing for exit-polls". In: Matsui, M., Zuccherato, R.J. (eds.) SAC 2003. LNCS, vol. 3006, pp. 160–174. Springer, Heidelberg (2004)
25. Wikström, D.: A universally composable mix-net. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 315–335. Springer, Heidelberg (2004)
26. Wikström, D.: A sender verifiable mix-net and a new proof of a shuffle. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 273–292. Springer, Heidelberg (2005)
27. Wikström, D.: Simplified submission of inputs to protocols. Cryptology ePrint Archive, Report 2006/259 (2006), http://eprint.iacr.org/

Finite Field Multiplication Combining AMNS and DFT Approach for Pairing Cryptography

Nadia El Mrabet[1] and Christophe Negre[2]

[1] LIRMM Laboratory, I3M, CNRS, University Montpellier 2, France
elmrabet@lirmm.fr
[2] DALI/ELIAUS, University of Perpignan, France

Abstract. Pairings over elliptic curves use fields $\mathbb{F}_{p^k}$ with $p \geq 2^{160}$ and $6 < k \leq 32$. In this paper we propose to represent elements in $\mathbb{F}_p$ with AMNS sytem of [1]. For well chosen AMNS we get roots of unity with sparse representation. The multiplication by these roots are thus really efficient in $\mathbb{F}_p$. The DFT/FFT approach for multiplication in extension field F_{p^k} is thus optimized. The resulting complexity of a multiplication in $\mathbb{F}_{p^k}$ combining AMNS and DFT is about 50% less than the previously recommended approach [2].

Keywords: Pairing, finite fields, AMNS, discrete Fourrier transform.

1 Introduction

Bilinear pairing in cryptography got an increasing interest during the past decade. Pairings were first use to attack discrete logarithm problem over elliptic curves like in MOV attack [3]. Since 2001, they are used also in a constructive way. Specifically, new important and original protocols using bilinear pairing have been proposed (e.g. Identity Based Cryptography [4] or Short Signature [5]). The most popular pairings used in pairing cryptography are defined over elliptic curves $E(\mathbb{F}_{q^k})$ (namely the Weil, Tate, η_T and Ate pairings [6]). Pairing evaluation over elliptic curve $E(\mathbb{F}_{q^k})$ involves arithmetical operations as multiplications and additions in the field $\mathbb{F}_{q^k}$ [2].

Fields $\mathbb{F}_{q^k}$ used in elliptic pairings are specific : q must have bit length bigger than 160 for security reasons and $6 < k < 32$ for optimization and security reasons. For now, the principal method [2] proposed to multiply elements in $\mathbb{F}_{q^k}$ uses a mix of Karatsuba and Toom-Cook multiplications methods. Consequently, they focus on k of the form $k = 2^i 3^j$, the resulting fields are called *friendly field* and the cost of a multiplication in $\mathbb{F}_{q^k}$ is equal to $3^i 5^j$ multiplications in $\mathbb{F}_q$.

Recently Discrete Fourier Transform approach has been proposed [7] to implement multiplication in $\mathbb{F}_{q^6}$ where $q = 3^n$. In practice Discrete Fourier Transform approach is interesting for quite large extension degree k. But here the underlying fields are quite big, so if the use of DFT can save even a small number of multiplications in $\mathbb{F}_q$ this can be advantageous.

In this paper, we extend the use of DFT for fields $\mathbb{F}_{p^k}$ where p is now a prime integer. The multiplication with DFT requires, in the best case, $2k - 1$

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 422–436, 2009.

multiplications in $\mathbb{F}_p$ and $O(k^2)$ multiplications by roots of unity. If FFT can be used, the cost of DFT approach becomes $O(n\log(n))$ operations in $\mathbb{F}_p$. If the field $\mathbb{F}_p$ is represented in usual way, the DFT approach remains too costly. Indeed, in this case the roots of unity are generally not nice (e.g. with a dense binary representation) and multiplications by these roots are costly. We propose here to use the AMNS system of $\mathbb{F}_p$ defined in [1]. In this situation we can manage to get roots of unity with nice representation, providing multiplications which are almost cost free. These multiplications can thus be neglect and the resulting multiplication algorithm in $\mathbb{F}_{p^k}$ requires only $2k-1$ multiplications in $\mathbb{F}_p$.

The paper is organized as follows : in Section 2 we recall the definition of the AMNS for representing integers modulo p and the arithmetic in this system. In Section 3 we recall the discrete Fourier transform approach for multiplication in $\mathbb{F}_{p^k}$ and extend it to specific cases in Subsection 3.3. We then focus on DFT friendly fieds (Section 4) which get benefit of a combination of AMNS and DFT for field multiplicattion. We evaluate the complexity of our approach for several field extensions and compare them to friendly fields. We finally conclude in Section 5.

2 Prime Field Arithmetic in AMNS System

Modular arithmetic operations like addition or multiplication modulo p consist to add or multiply two integers $0 \leq a, b < p$ and reduce the result modulo p if it is bigger than p.

Efficient arithmetic modulo a prime integer p is generally deeply related to the system of representation used to represent the elements. Generally integers are expressed as a sum $a = \sum_{i=0}^{\ell} a_i \gamma^i$ where $0 \leq a_i < \gamma$ and γ^ℓ has approximately the size of p. In practice γ is often chosen as 2^w where w is the size of computer words.

Here we will use an original system of representation in $\mathbb{F}_p$ introduced in [1] by Bajard, Imbert and Plantard called the *Adapted Modular Number System*. The main idea of the AMNS consists to relax the fact that $\gamma \cong p^{1/\ell}$. We take γ freely in $[0, p]$ such that each $0 \leq a < p$ can be written as $a = \sum_{i=0}^{\ell} a_i \gamma^i \mod p$ with $a_i \in [0, p^{1/\ell}]$. The advantage is that γ can be taken as $\gamma^\ell = \lambda \mod p$ where λ is small.

Definition 1 (AMNS [1]). *An Adapted Modular Number System $\mathcal{B}$, is a quadruple $(p, \ell, \gamma, \rho)_E$, where $E = t^\ell - \lambda$ such that $\gamma^\ell - \lambda = 0 \mod p$ and such that for all positive integers $0 \leq a < p$ there exists a polynomial $\mathfrak{a}(t) = \sum_{=0}^{\ell-1} a_i t^i$ satisfying*

$$\begin{aligned} &\mathfrak{a}(\gamma) = a \bmod p, \\ &\deg(\mathfrak{a}(t)) < \ell, \\ &\|\mathfrak{a}\|_\infty = \max_{i=1}^{\ell} |a_i| < \rho. \end{aligned} \tag{1}$$

The polynomial $\mathfrak{a}(t)$ is a representation of a in $\mathcal{B}$.

Generally in AMNS we have $\gamma \cong p$ and small coefficients $|a_i| < \rho \cong p^{1/\ell}$.

Table 1. The elements of $\mathbb{Z}_{17}$ in $\mathcal{B} = MNS(17, 3, 7, 2)$

0	1	2	3	4	5
0	1	$-t^2$	$1-t^2$	$-1+t+t^2$	$t+t^2$

6	7	8	9	10	11
$-1+t$	t	$1+t$	$-t-1$	$-t$	$-t+1$

12	13	14	15	16
$-t-t^2$	$1-t-t^2$	$-1+t^2$	t^2	-1

Example 1. In Table 1, we give the representation in the AMNS $\mathcal{B} = (17, 3, 7, 2)$ for each element modulo $p = 17$.

In particular, we can verify that if we evaluate $(-1 + t + t^2)$ in γ, we have $-1+\gamma+\gamma^2 = -1+7+49 = 55 \equiv 4 \bmod 17$. We have also that $\| -1+t+t^2\|_\infty = 1 < 2$.

In [8] the authors showed that it is possible to build an AMNS of length ℓ when it is possible to compute a polynomial $\mathfrak{m}(\gamma) = 0 \mod p$ with $\|\mathfrak{m}\|_\infty$ small.

Proposition 1. *Let p be a prime integer and $\lambda \in \mathbb{Z}, \ell \in \mathbb{N}$ such that the polynomial $E = t^\ell - \lambda$ admits a root γ in $\mathbb{F}_p$. Then the following statements are true.*

i) There exists a polynomial $\mathfrak{m}$ such that $\mathfrak{m}(\gamma) = 0$ and $\|\mathfrak{m}\|_\infty \leq (\ell!)^{1/\ell} p^{1/\ell}$.
ii) Let $\sigma = \|\mathfrak{m}\|_\infty$ and $\rho = 2|\lambda|\ell\sigma$ then the system $\mathcal{B} = (p, \ell, \gamma, \rho)_E$ is an AMNS of $\mathbb{F}_p$.

Fields used in cryptographic pairing have a p randomly constructed. Thus multiplication modulo p cannot use some rare property of p, like the primes considered in [1]. The better algorithm in AMNS which does not use rare property of prime p is the Montgomery-like multiplication presented in [8].

Algorithm 1: AMNS Multiplication

Input : $\mathfrak{a}, \mathfrak{b} \in \mathcal{B} = (p, \ell, \gamma, \rho)_E$ with $E = t^\ell - \lambda$
Data : $\mathfrak{m}$ a polynomial such that $\mathfrak{m}(\gamma) \equiv 0 \pmod p$
an integer ϕ and $\mathfrak{m}' = -\mathfrak{m}^{-1} \bmod (E, \phi)$
Output: $\mathfrak{r}(t)$ such that $\mathfrak{r}(\gamma) = \mathfrak{a}(\gamma)\mathfrak{b}(\gamma)\phi^{-1} \bmod p$
begin
  $\mathfrak{c} \leftarrow \mathfrak{a} \times \mathfrak{b} \bmod E$;
  $\mathfrak{q} \leftarrow \mathfrak{c} \times \mathfrak{m}' \bmod (E, \phi)$;
  $\mathfrak{r} \leftarrow (\mathfrak{c} + \mathfrak{q} \times \mathfrak{m} \bmod E)/\phi$;
end

According to [8] this algorithm is correct if $\phi \geq 2\ell\lambda\rho$. Concerning the implementation of Algorithm 1, it requires essentially three polynomial multiplications where polynomial coefficients are smaller than ρ and ϕ.

Such polynomial multiplication can be implemented using classical approach : for really small length ℓ schoolbook method are generally recommended, for bigger ℓ Karatsuba or Toom-Cook should be better. We will use here $\ell \leq 60$, thus, we will always use one of this two methods.

3 Field Extension Arithmetic

An extension field $\mathbb{F}_{p^k}$ can be seen as the set of polynomials with coefficient in $\mathbb{F}_p$ and degree less than k

$$\mathbb{F}_{p^k} = \{U(X) \in \mathbb{F}_p[X] \text{ s.t. } \deg U < k\}.$$

Arithmetic in this set is done modulo an irreducible polynomial P with degree k. Since p is large, the polynomial P can be taken, in general, with a binomial form $X^k - \alpha$ with small α (cf. [2,9]). In this situation the multiplication modulo P of two elements

$$U = \sum_{i=0}^{k-1} u_i X^i \text{ and } V = \sum_{i=0}^{k-1} v_i X^i$$

consists first to compute the product $W = U \times V$ and then to reduce it modulo P. Since P is a binomial, the reduction modulo P is simple. We split $W = \underline{W} + X^k \overline{W}$ with $\deg \underline{W} \leq k$ and compute $\underline{W} + \alpha \overline{W}$ since $X^k \equiv \alpha \mod P$. The main challenge is thus to perform efficiently the polynomial multiplication $U \times V$.

3.1 Polynomial Multiplication Using DFT

We recall here the Discrete Fourier Transform (DFT) approach for polynomial multiplication. This approach is a special case of the multi-evaluation/ interpolation strategy [10]. Multi-evaluation/interpollation perform a polynomial multiplication of two polynomials U and V by evaluating both of them in $n \geq 2k-1$ elements of $\mathbb{F}_p$. Then we deduce the evaluation of $W = U \times V$ by computing term by term the evaluation of U and V. Finally we perform a Lagrange interpolation to get the polynomial form of W.

In the DFT approach the evaluation set used is the set of n-th roots of unity. Specifically, let $\omega \in \mathbb{F}_p$ be a primitive root of unity, then the DFT works as follow.

1. *Multi-evaluation.* Let U, V be two polynomials in $\mathbb{F}_p[X]$ with degree k. We compute the multi-evaluation of U

$$\hat{U} = DFT_\omega(U) = (U(1), U(\omega), \ldots, U(\omega^{n-1})).$$

This operation is usually called the Discrete Fourier Transform of U. The same is done for V. This operation can be done through a matrix vector product

$$\hat{U} = \begin{bmatrix} 1 & \omega & \omega^2 & \cdots & \omega^{k-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{(k-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(k-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{k-1} \end{bmatrix}.$$

2. *Term by term multiplications.* They are performed on $\hat{U}$ and $\hat{V}$

$$\hat{W} = (\hat{u}_1 \times \hat{v}_1, \hat{u}_2 \times \hat{v}_2 \ldots, \hat{u}_n \times \hat{v}_n),$$

we get the multi-evaluation of W where $W = U \times V$.

3. *Interpolation.* The interpolation consists to compute the polynomial form of W knowing its multi-evaluation in $\boldsymbol{\omega} = (1, \omega, \omega^2, \ldots, \omega^{(n-1)})$.

Lemma 1. *Let $\mathbb{F}_p$ be a prime field and ω be a primitive n-th root of unity. Let*

$$\Omega = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{(n-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \ldots & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{2}$$

its inverse is given by

$$\Omega^{-1} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega' & \omega'^2 & \cdots & \omega'^{n-1} \\ 1 & \omega'^2 & \omega'^4 & \cdots & \omega'^{(n-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega'^{n-1} & \omega'^{2(n-1)} & \ldots & \omega'^{(n-1)(n-1)} \end{bmatrix} \tag{3}$$

where $\omega' = \omega^{-1} = \omega^{n-1}$.

In this situation the interpolation is computed by applying Ω^{-1} to $\hat{W}$ and keeping only the first $2k-1$ coefficients. We obtain

$$W = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega' & \omega'^2 & \cdots & \omega'^{n-1} \\ 1 & \omega'^2 & \omega'^4 & \cdots & \omega'^{(n-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega'^{2k-2} & \omega'^{2(2k-2)} & \ldots & \omega'^{(n-1)(2k-2)} \end{bmatrix} \cdot \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \vdots \\ \hat{w}_n \end{bmatrix}.$$

Remark 1 (Montgomery representation.). We can avoid the division by n in the interpolation process. Indeed, if we use a Montgomery representation (cf [11]) of U and V

$$\widetilde{U} = \frac{1}{n}U \text{ and } \widetilde{V} = \frac{1}{n}V.$$

If we perfom DFT approach without the division by n to multiply $\widetilde{U}$ and $\widetilde{V}$ we get

$$n\widetilde{U}\widetilde{V} = n(\frac{1}{n}U) \times (\frac{1}{n}V) = \widetilde{W},$$

where $W = U \times V$. In other words DFT multiplication without division by n is stable in Montgomery representation. This representation is also stable under addition and reduction modulo P. It can thus be used in a chain of multiplication/addition, like in pairing evaluation over elliptic curves.

3.2 Fast Fourier Transformation (FFT)

Let $U = \sum_{i=0}^{n-1} u_i X^i \in \mathbb{F}_p[X]$ and ω be a primitive root of unity. The fast Fourier transform (FFT) is an algorithm which performs efficiently the evaluation of U in $\boldsymbol{\omega} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$. The FFT process is based on the following two-way splitting of U

$$\begin{aligned} U_1 &= \textstyle\sum_{j=0}^{n/2-1} a_{2j} X^{2j}, \\ U_2 &= \textstyle\sum_{j=0}^{n/2-1} a_{2j+1} X^{2j}, \end{aligned}$$

such that $U = U_1 + XU_2$.

Let $\widehat{U}[i] = U(\omega^i)$ be the i-th coefficient of $\hat{U} = DFT_\omega(U)$. Let us also denote by $\hat{U}_1[i], \hat{U}_2[i]$ the coefficients of $DFT_{\omega^2}(U_1)$ and $DFT_{\omega^2}(U_2)$ in $\{1, \omega^2, \omega^4, \ldots, (\omega^2)^{n/2-1}\}$. If we evaluate $U = U_1 + XU_2$ in ω^i and $\omega^{i+n/2} = -\omega^i, i < n/2$ we get

$$\begin{aligned} \widehat{U}[i] &= \widehat{U}_1[i] + \omega^i \widehat{U}_2[i] \\ \widehat{U}[i + n/2] &= \widehat{U}_1[i] - \omega^i \widehat{U}_2[i] \end{aligned}$$

The computation $DFT_\omega(U)$ is thus reduced to the computation of $DFT_{\omega^2}(U_1)$ and $DFT_{\omega^2}(U_2)$. These computations can be done recursivelly. The resulting algorithm has a cost of $\frac{n}{2}\log_2(n)$ multiplications by ω^i in $\mathbb{F}_p$ and $n\log_2(n)$ additions/subtractions.

3.3 Multiplication with DFT When $n \leq 2k-2$

DFT approach for multiplication uses evaluations and interpolation in a set of $n \geq 2k-2$ roots of unity in order to get the correct product W. In some situations there is no primitive n-th root of unity with $n \geq 2k-1$ and n close to $2k-1$. In these situations DFT approach is not practical. We present here an extension of the DFT approach when there exists primitive a n-th root of unity smaller than $2k-1$. We focus here on two cases $n = 2k-2$ and $n = 2k-4$, which correspond to practical situations (see Section 4). The following approach generalizes the method presented in [7] when $k = 3$ and $n = 6$ to other value of k.

Lemma 2. *Let $\mathbb{F}_p$ be a prime field, ω be a primitive n-th root of unity and Ω and Ω^{-1} be the matrices defined in Lemma 1. We consider $U = \sum_{i=0}^{k-1} u_i X^i$, $V = \sum_{i=0}^{k-1} v_i X^i$ and $W = U \times V$ and we assume that $n = 2k-2$. Then W can be computed as follow.*

1. $\hat{U} = DFT_\omega(U), \hat{V} = DFT_\omega(V)$.
2. $w_{2k-2} = u_{k-1} \times v_{k-1}$
3. *The coefficients* w_i *for* $i = 0, \ldots, 2k-3$ *are computed as*

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{2k-3} \end{bmatrix} = \Omega^{-1} \cdot \begin{bmatrix} \hat{u}_i \times \hat{v}_i - w_{2k-2} \\ \hat{u}_i \times \hat{v}_i - w_{2k-2} \\ \vdots \\ \hat{u}_i \times \hat{v}_i - w_{2k-2} \end{bmatrix}. \tag{4}$$

Proof. Since $n = 2k-2$ and U and V have degree $k-1$ then $U \times V = W = \sum_{i=0}^{n} w_i X^i$ has degree n and $W = \sum_{i=0}^{n} w_i X^i$. The evaluation $\hat{W}$ of W in the n elements ω^i gives

$$\begin{cases} \hat{w}_0 = W(1) = w_0 + w_1 + \ldots + w_{n-1} + w_n \\ \hat{w}_1 = W(\omega) = w_0 + w_1\omega + \ldots + w_{n-1}\omega^{n-1} + w_n\omega^n \\ \hat{w}_2 = W(\omega^2) = w_0 + w_1(\omega^2) + \ldots + w_{n-1}(\omega^2)^{n-1} + w_n(\omega^2)^n \\ \quad \vdots \qquad\qquad\qquad \vdots \\ \hat{w}_{n-1} = W(\omega^{n-1}) = w_0 + w_1(\omega^{n-1}) + \ldots + w_{n-1}(\omega^{n-1})^{n-1} + w_n(\omega^{n-1})^n \end{cases}.$$

Now, since $\omega^n = 1$, we have $(\omega^i)^n = 1$ for $i = 1, \ldots, n-1$. The right part of the previous equations rewrites as

$$\begin{cases} \hat{w}_0 = W(1) = w_0 + w_1 + \cdots + w_n \\ \hat{w}_1 = W(\omega) = w_0 + w_1\omega + \cdots + w_n\omega^{n-1} + w_n \\ \hat{w}_2 = W(\omega^2) = w_0 + w_1(\omega^2) + \cdots + w_{n-1}(\omega^2)^{n-1} + w_n \\ \quad \vdots \qquad\qquad\qquad \vdots \\ \hat{w}_{n-1} = W(\omega^{n-1}) = w_0 + \cdots + w_{n-1}(\omega^{n-1})^{n-1} + w_n \end{cases} \tag{5}$$

The coefficient w_n is already known since $w_n = w_{2k-2} = u_{k-1}v_{k-1}$. Using (5), we remark that the vector $(\hat{w}_0 - w_n, \hat{w}_1 - w_n, \ldots, \hat{w}_{n-1} - w_n)$ is the discrete Fourier transform of the polynomial $W' = \sum_{i=0}^{n-1} w_i X^i$. Thus we get back to the coefficients of W' by computing

$$\Omega^{-1} \cdot \left[\hat{w}_0 - w_n \ \hat{w}_1 - w_n \cdots \ \hat{w}_{n-1} - w_n) \right]^t.$$

This corresponds to Eq. (4).

We focus now on the case $n = 2k-4$.

Lemma 3. *Let* $\mathbb{F}_p$ *be a prime field and* $\omega \in \mathbb{F}_p$ *a primitive* n*-th root of unity. Let* $U = \sum_{i=0}^{k-1} u_i X^i$ *and* $V = \sum_{i=0}^{k-1} v_i X^i$ *in* $\mathbb{F}_p[X]$*. Let* $W = U \times V$ *and assume that* $n = 2k-4$*, then the coefficients of* W *can be computed as follows.*

1. $\hat{U} = DFT_\omega(U), \hat{V} = DFT_\omega(V)$.
2. $w_{2k-2} = u_{k-1} \times v_{k-1}, w_0 = u_0 \times v_0$
3. $w_{2k-3} = u_{k-1} \times v_{k-2} + u_{k-2} \times v_{k-1}$

4. The coefficients w_i for $i = 1, \ldots, 2k-4$ are computed as

$$\begin{bmatrix} w_1 \\ w_2 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \Omega^{-1} \cdot \begin{bmatrix} \hat{w}_0 - w_0 - w_{n+1} - w_{n+2} \\ (\hat{w}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} \\ (\hat{w}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} \\ \vdots \\ (\hat{w}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} \end{bmatrix} \tag{6}$$

where Ω^{-1} is defined in Eq. (3).

Proof. The proof is similar to the proof Lemma 2. Since $n = 2k-4$ and U and V have degree $k-1$ then $U \times V = W = \sum_{i=0}^{n+2} w_i X^i$ has degree $n+2 = 2k-2$. The evaluation $\hat{W}$ of W in the n elements ω^i gives

$$\begin{cases} \hat{w}_0 = W(1) = w_0 + w_1 + \cdots + w_n + w_{n+1} + w_{n+1} \\ \hat{w}_1 = W(\omega) = w_0 + w_1\omega + \cdots + w_n\omega^n + w_{n+1}\omega^{n+1} + w_{n+2}\omega^{n+2} \\ \hat{w}_2 = W(\omega^2) = w_0 + w_1(\omega^2) + \cdots + w_n\omega^n + w_n(\omega^2)^n + w_{n+2}(\omega^2)^{n+2} \\ \quad\vdots \qquad\qquad\qquad\qquad \vdots \\ \hat{w}_{n-1} = W(\omega^{n-1}) = w_0 + w_1(\omega^{n-1}) + \cdots + w_n(\omega^{n-1})^n \\ \qquad\qquad\qquad\qquad + w_{n+1}(\omega^{n-1})^{n+1} + w_{n+2}(\omega^{n-1})^{n+2} \end{cases}$$

Now, since $\omega^n = 1$, we have $(\omega^i)^{n+1} = \omega^i$ and $(\omega^i)^{n+2} = (\omega^i)^2$for $i = 1, \ldots, n-1$ in the right part of the previous equations. We get for $\hat{w}_i$

$$\begin{cases} \hat{w}_0 = W(1) = w_0 + w_1 + \cdots + w_n + w_{n+1} + w_{n+2} \\ \hat{w}_1 = W(\omega) = w_0 + w_1\omega + \cdots + w_n\omega^n + w_{n+1}\omega + w_{n+1}\omega^2 \\ \hat{w}_2 = W(\omega^2) = w_0 + w_1(\omega^2) + \cdots + w_n(\omega^2)^n + w_{n+1}\omega^2 + w_{n+2}(\omega^2)^2 \\ \quad\vdots \qquad\qquad\qquad\qquad \vdots \\ \hat{w}_{n-1} = W(\omega^{n-1}) = w_0 + \cdots + w_n(\omega^{n-1})^n + w_{n+1}\omega^{n-1} + w_{n+2}(\omega^{n-1})^2 \end{cases}$$

The three coefficients w_0, w_{n+1}, w_{n+2} are already known. Then we rewrite the previous equations as follows

$$\begin{cases} \hat{w}_0 - w_0 - w_{n+1} - w_{n+2} = w_1 + \cdots + w_{n-1} \\ (\hat{w}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} = w_1 + \cdots + w_n\omega^{n-1} \\ (\hat{w}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} = w_1 + \cdots + w_{n-1}(\omega^2)^{n-1} \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \qquad\qquad\qquad \vdots \\ (\hat{w}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} = w_1 + \cdots + w_{n-1}(\omega^{n-1})^{n-1} \end{cases}$$

We remark now that the right part corresponds to the evaluation of the polynomial $W' = \sum_{i=0}^{n-1} w_{i+1}X^i$. Consequently we get the coefficients of W' by computing the matrix vector product

$$\begin{bmatrix} w_1 \\ w_2 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \frac{1}{n}\Omega \cdot \begin{bmatrix} \hat{w}_0 - w_0 - w_{n+1} - w_{n+2} \\ (\hat{w}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} \\ (\hat{w}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} \\ \vdots \\ (\hat{w}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} \end{bmatrix}$$

This corresponds to Eq. (6).

The previous Lemmas are just special cases. They can be extended to smaller value of n by increasing the number of precomputed w_i for small indices $i = 1, 2, \ldots$ and big indices $i = 2k-2, 2k-3, \ldots$ before the interpolation process using Ω^{-1}.

Remark 2. In the two situations of Lemma 2 and Lemma 3 we can use also Montgomery representation of polynomials to avoid division by n. For example if Lemma 2 is applied without the division by n to $\widetilde{U} = \frac{1}{n}U$ and $\widetilde{V} = \frac{1}{n}V$, we obtain the coefficients $\widetilde{w_i}$ of $\widetilde{W} = \frac{1}{n}U \times V$ for $i = 0, \ldots, 2k-2$. Only $\widetilde{w}_{2k-1}$ must be computed separatly $\widetilde{w}_{2k-2} = \frac{1}{n}w_{2k-1}$ to get all the coefficients of $\widetilde{W}$. The same strategy can be applied to Lemma 3 with 3 more multiplications by $\frac{1}{n}$.

3.4 Complexity of Different DFT Methods

We evaluate the complexity of the different DFT approaches. We distinguished the cases where DFT is performed through a matrix vector product and where DFT is performed using FFT algorithm. We express the costs in term of the number of operations in $\mathbb{F}_p$: the number of multiplication by roots of unity, addition/subtraction, and multiplication. For multi-evaluation and interpolation we used the fact that the entries in Ω and Ω^{-1} are all powers of ω. We also assume that multiplication are done in Montgomery representation, in order to avoid the division by $\frac{1}{n}$ (cf. Remark 1 and Remark 2). We obtain the complexity given in Table 2.

Table 2. Complexity of DFT approaches

Method	# Mult. by ω^i	# Mult.	# Add.
General DFT	$4nk - 3n$	n	$4nk - 3n$
General FFT	$\frac{3n}{2}\log_2(n)$	n	$3n\log_2(n)$
Lemma 2	$3(2k-3)^2$	$2k$	$(2k-2)(6k-8)$
Lemma 2 with FFT	$3(k-1)\log_2(2k-2)$	$2k$	$3(2k-2)\log_2(2k-2) + (2k-2)$
Lemma 3	$3(2k-5)^2 + 2(2k-5)$	$2k+3$	$3(2k-4)(2k-4-1) + 2(2k-4)$
Lemma 3 with FFT	$3(k-2)\log_2(2k-4) + 3(2k-5)$	$2k+3$	$3(2k-4)(\log_2(2k-4)+1)$

4 DFT Friendly Field

We focus in this section on specific fields called DFT friendly fields. These fields admit an AMNS which provide efficient multiplication by roots of unity.

4.1 Definition of DFT Friendly Field

The main goal is to find a way to obtain fields $\mathbb{F}_p$ with n-roots of unity such that $n \in \{2k-1, 2k-2, 2k-3\}$ and such that the multiplication by these roots are really efficient. We propose to consider fields $\mathbb{F}_{p^k}$ satisfying the following definition.

Definition 2 (DFT Friendly Field). *We call a DFT friendly field an extension field $\mathbb{F}_{p^k}$ such that $\mathbb{F}_p$ admits an AMNS $\mathcal{B} = (p, \ell, \gamma, \rho)_E$ of length ℓ and such that one of the following conditions holds*

1. *$\lambda = 1$ and $\ell \in \{2k-1, 2k-2, 2k-4\}$ and γ is primitive ℓ-th root of unity.*
2. *$\lambda = -1$ and $\ell \in \{k-1, k-2\}$ and γ is primitive 2ℓ-th root of unity.*

Since we have roots of unity with appropriate order, we can use DFT approaches presented in Section 3 to perform the multiplication of elements in $\mathbb{F}_{p^k}$. Indeed the condition on ℓ in each cases of Definition 2 enables us to use at least one of the strategies expressed in Subsection 3.1 or Lemma 2 and Lemma 3.

In DFT Friendly fields, the roots of unity are the elements $\pm\gamma^i$. The multiplication by these roots can be done using the formula stated in the following Lemma.

Lemma 4. *Let an AMNS $\mathcal{B} = (p, \ell, \gamma, \rho)_E$ and $\mathfrak{a} = \sum_{i=0}^{n-1} a_i \gamma^i$ be expressed in $\mathcal{B}$. The multiplication of $\mathfrak{a}$ by the power γ^i of γ is given by*

$$\mathfrak{a}\gamma^i = \lambda a_{n-i} + \lambda a_{n-i+1}\gamma + \cdots + \lambda a_{n-1}\gamma^{i-1} + a_0\gamma^i + \cdots + a_{n-i-1}\gamma^{n-1}$$

Proof. The proof is a direct consequence of the definition of an AMNS.

In our case, the field $\mathbb{F}_p$ is represented with an AMNS where $\lambda = \pm 1$. Consequently the multiplication by $\pm\gamma^i$ consists of only a cyclic shift, with eventually some changes of sign. The multiplication by $\pm\gamma^i$ is almost free of computations. Consequently, they do not add multiplications. The total cost of an AMNS multiplication in term of operations on the finite field is given in Table 3.

4.2 Fields Used Pairing Cryptography

We recall here different methods used to construct ordinary elliptic curves and corresponding finite fields providing pairing. The curve order $\#E(\mathbb{F}_p)$ must have a big prime factor, called r and an extension degree $6 < k \leq 32$. To get such curve the most used method is based on Complex Multiplication (CM).

The construction of a curve with the CM method requires to solve a system of equations (7) where the indeterminates are an integer D, the embedding degree k, the prime factor r, t the trace of the Frobenius on $E(\mathbb{F}_p)$ and p the characteristic of the finite field:

$$\begin{cases} r \mid p + t - 1, \\ r \mid p^k - 1, \text{ for primes } r, p, \\ Dy^2 = 4p - t^2 \text{ for some integer } y. \end{cases} \tag{7}$$

Several methods exist to solve this system. An overview of these different methods is given in [12]. We recall here the two following methods

– The Miyaji-Nakabayashi-Takano (MNT) strategy is one of the first CM method [13] to construct elliptic curves suitable for ECC. It was extended

by Barreto and Naehrig [14] to construct elliptic curves with embedding degree 12. These curves with embedding degree 12 are given by the following parametrization:

$$\begin{aligned} k &= 12, \\ p &= x^4 + 36x^3 + 24x^2 + 6x + 1, \\ r &= 36x^4 + 36x^3 + 18x^2 + 6x + 1, \\ t &= 6x^2 + 1. \end{aligned}$$

– The second method which could be used in order to build curves with arbitrary embedding degree k is the Cocks-Pinch method [15]. This method generates curves with arbitrary r, such that $\#E(\mathbb{F}_p)/r \approx 2$. The extension of the Cocks-Pinch method given in [16] provides smaller value for $\#E(\mathbb{F}_p)/r$. Their method can be applied for general embedding degree. For example in [16] they generated a family of curves with embedding degree 16. This family is given by the following polynomials:

$$\begin{aligned} k &= 16, \text{ for } x \equiv \pm 25 \bmod (70) \\ p &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 \\ &\quad + 625x^2 + 2398x + 3125)/980, \\ r &= (x^8 + 4x^4 + 625)/61250, \\ t &= (2x^5 + 41x + 35)/35. \end{aligned}$$

For all these constructions, the prime p is constructed randomly. Proposition 1 tells us that if there exists a primitive ℓ-th (or 2ℓ-th) root of unity, where ℓ satisfies the conditions of Definition 2, then we can construct an AMNS satisfying Definition 2.

For a random prime p, the probability that it has a primitive ℓ-th root of unity is roughly $1/(\ell - 1)$. Indeed p has a root of unity if and only if $p \equiv 1 \mod \ell$. But prime are equally distributed in the set of classes modulo ℓ. Consequently for small value of ℓ, we can easily find a DFT friendly field $\mathbb{F}_p$ and an elliptic curve over this field providing practical pairing.

4.3 Complexity Comparison

We present in Table 3 the complexity of a multiplication in DFT friendly fields $\mathbb{F}_{p^k}$ for different sizes of k. This complexity is given in term of the number of multiplications and additions in $\mathbb{F}_p$. The complexity is deduced from Table 2 : we neglect the cost of the multiplication by a root of unity since it is almost cost free. Indeed, a multiplication by a root of unity consists only in cyclic shifts. We also specify if we use FFT or not. Table 3 we also give the complexity of the multiplication in friendly field using Karatsuba and Toom Cook multiplications [9].

We remark that even for small value of k, DFT approach seems competitive regarding to the number of multiplications. When no FFT can be used, the number of additions increases significantly and should make our approach incompetitive in these special cases.

Table 3. Complexity comparison for practical extension degree k

Method	k	Cost of $Mult_{F_{p^k}}$	
		# Add. in $\mathbb{F}_p$	# Mult. in $\mathbb{F}_p$
Karatsuba/Toom-Cook [2,9]	6	60	15
Karatsuba/Toom-Cook [2,9]	8	72	27
Subsection 3.1 with FFT and $E = t^8 + 1$	8	192	16
Karatsuba/Toom-Cook [2,9]	9	160	25
Lemma 2 with FFT and $E = t^8 + 1$	9	208	18
Lemma 3 with FFT and $E = t^8 + 1$	10	240	23
Subsection 3.1 with $E = \sum_{i=0}^{10}(-t)^i$	11	902	22
Karatsuba/Toom-Cook [2,9]	12	180	45
Lemma 2 with $E = \sum_{i=0}^{10}(-t)^i$	12	1408	24
Lemma 3 with $E = \sum_{i=0}^{10}(-t)^i$	13	1430	28
Karatsuba/Toom-Cook [2,9]	16	248	81
Subsection 3.1 with FFT and $E = t^{16} + 1$	16	480	32
Lemma 2 with FFT and $E = t^{16} + 1$	17	512	34
Lemma 3 with FFT and $E = t^{16} + 1$	18	576	39
Karatsuba/Toom-Cook [2,9]	24	588	135

5 Conclusion

We have presented in this paper a new approach for multiplication in fields $\mathbb{F}_{p^k}$ used in pairing cryptography. We used AMNS [1] to represent elements in $\mathbb{F}_p$ and DFT approach for extension field arithmetic. Specificaly we pointed out that some AMNS provide efficient multiplication by roots of unity and thus optimize DFT approach. Our approch decrease the number of multiplications, but increase the number of additions in $\mathbb{F}_p$ in order to compute a multiplication in $\mathbb{F}_{p^k}$. However, the number additional additions is not so important compared to the saved multiplications. The resulting multiplication in the extension field $\mathbb{F}_{p^k}$ requires less multiplications in $\mathbb{F}_p$ for different practical sizes of k than previously recommended method [2,9]. Specifically for $k \geq 12$ combined AMNS and DFT approach in DFT friendly fields, proposed in this paper, decreases the number of multiplications in $\mathbb{F}_p$ by 50%.

References

1. Plantard, T.: Modular arithmetic for cryptography. PhD thesis, LIRMM, Université Montpellier 2 (2005)
2. Koblitz, N., Menezes, A.: Pairing-based cryptography at high security levels. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 13–36. Springer, Heidelberg (2005)
3. Menezes, A., Vanstone, S., Okamoto, T.: Reducing elliptic curve logarithms to logarithms in a finite field. In: STOC 1991: Proceedings of the twenty-third annual ACM symposium on Theory of computing, pp. 80–89. ACM Press, New York (1991)

4. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
5. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Matsuda, S., Kanayama, N., Hess, F., Okamoto, E.: Optimised versions of the ate and twisted ate pairings. In: Galbraith, S.D. (ed.) Cryptography and Coding 2007. LNCS, vol. 4887, pp. 302–312. Springer, Heidelberg (2007)
7. Gorla, E., Puttmann, C., Shokrollahi, J.: Explicit formulas for efficient multiplication in $\mathbb{F}_{3^{6m}}$. In: Adams, C., Miri, A., Wiener, M. (eds.) SAC 2007. LNCS, vol. 4876, pp. 173–183. Springer, Heidelberg (2007)
8. Negre, C., Plantard, T.: Efficient modular arithmetic in adapted modular number system using lagrange representation. In: Proceedings of Australasian Conference on Information Security and Privacy (ACISPP 2008) (2008)
9. Bajard, J., Mrabet, N.E.: Pairing in cryptography: an arithmetic point of view. In: Advanced Signal Processing Algorithms, Architectures and Implementations XVI, SPIE (August 2007)
10. ZurGathen, J.V., Gerhard, J.: Modern Computer Algebra. Cambridge University Press, New York (2003)
11. Montgomery, P.L.: Modular multiplication without trial division. Mathematics of Computation 44(170), 519–521 (1985)
12. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive (2006), http://eprint.iacr.org/2006/372
13. Miyaji, A., Nakabayashi, M., Takano, S.: New explicit conditions of elliptic curve traces for fr-reduction (2001)
14. Barreto, P., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 319–331. Springer, Heidelberg (2006)
15. Cocks, C., Pinch, R.: Identity-based cryptosystems based on the Weil pairing (2001)
16. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs Codes and Cryptography 37(1), 133–141 (2005)
17. Kachisa, E.J., Schaefer, E.F., Scott, M.: Constructing brezing-weng pairing friendly elliptic curves using elements in the cyclotomic field. In: Pairing 2008: Proceedings of the 2nd international conference on Pairing-Based Cryptography, pp. 126–135 (2008)

6 Annexe

We present in this section some examples of curves with embedding degree $6 < k \leq 32$ over DFT friendly field.

We first briefly recall the method given in [8] to construct an AMNS for a fixed prime p. We choose a polynomial $E(t) = t^\ell - \lambda$ of degree ℓ and compute γ a root of E in $\mathbb{F}_p$. Then we construct the matrix M:

$$M = \begin{bmatrix} p & 0 & 0 & \cdots & 0 \\ -\gamma & 1 & 0 & \cdots & 0 \\ -\gamma^2 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ -\gamma^{(\ell-1)} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

We apply LLL algorithm to this matrix and we obtain a short vector $\mathfrak{m}$ satisfying $\mathfrak{m}(\gamma) = 0 \mod p$. We finally get $\rho = 2\ell|\lambda|\|\mathfrak{m}\|_\infty$.

6.1 Curves with Embedding Degree $k = 12$

We use the parametrization of Barreto and Naehrig [14] which provides elliptic curves with embedding degree 12:

$$\begin{aligned} k &= 12, \\ p &= 36x^4 + 36x^3 + 24x^2 + 6x + 1, \\ r &= 36x^4 + 36x^3 + 18x^2 + 6x + 1, \\ t &= 6x^2 + 1. \end{aligned}$$

We use also the polynomial $E(t) = \sum_{i=0}^{10}(-t)^i$ to build the AMNS of p. For a security level of 80 (i.e. the best attack requires 2^{80} operations) we find the following example:

$$\begin{aligned} x &= 1099511637026, \\ p &= 52614060714492069992659260093542155440429911322253, \\ r &= 52614060714492069992659252839987115706863666574197, \\ t &= 725355503973366244748057, \\ \gamma &= 14348622953168487070046731700990451973985348345534, \\ \mathfrak{m}(X) &= 12376 - 49167X + 48460X^2 + 18281X^3 + 15213X^4 - 10299X^5 \\ &\quad +11263X^6 - 70120X^7 - 13636X^8 - 18106X^9. \end{aligned}$$

For a security level of 160 we found:

$$\begin{aligned} x &= 18446744073709692895, \\ p &= 41685152125435107379370057363152675115521120051443074052763868100 \\ &\quad 45976396949971, \\ r &= 41685152125435107379370057363152675115500703109427817432219558905 \\ &\quad 25925116063821, \\ t &= 20416942015256620544309195200512808861 51, \\ \gamma &= 37771108084317046107302986198165197419883853864047699317 64 \\ &\quad 1286502190684739139, \\ \mathfrak{m}(X) &= 8053715 - 20923230X + 23417521X^2 - 26826999X^3 + 19243643X^4 \\ &\quad +1059907X^5 - 41954237X^6 - 42180723X^7 + 5371359X^8 - 19196965X^9. \end{aligned}$$

6.2 Curves with Embedding Degree $k = 16$

We use the parametrization of [17]:

$$\begin{aligned} k &= 16, \\ p &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980, \\ r &= (x^8 + 48x^4 + 625)/61250, \\ t &= (2x^5 + 41x + 35)/35. \end{aligned}$$

We use the polynomial $E(t) = t^{16} + 1$.

For a security level of 80 we found the following example:

$$
\begin{aligned}
x &= 74156485,\\
p &= 513174771603192518069857791127277415092088396567880595361684047 8\\
&\quad 933959934561,\\
r &= 14930934707260179303940284190066288525962852908481890536993,\\
t &= 128146760584932038247348983414439772062,\\
\gamma &= 38696828658217738947551865824060486351009548229977673384 13\\
&\quad 19721386977404674,\\
\mathfrak{m}(X) &= 7400X + 49262X^2 - 3010X^3 - 14335X^4 + 34360X^5\\
&\quad +43021X^6 + 6813X^7 + 5184X^8 + 13206X^9 + 10037X^{10}\\
&\quad +2540X^{11} - 7384X^{12} - 66117X^{13} - 57557X^{14} + 32450X^{15}.
\end{aligned}
$$

For a security level of 160 we found :

$$
\begin{aligned}
x &= 300650886015,\\
p &= 615742037941290064431987586434433942899976129045006271675961520 9\\
&\quad 367079614301451112752451271493775945297121074689,\\
r &= 108991796562856949188268637826460013069843005574422145615453925 9\\
&\quad 930378582049607058754993,\\
t &= 140370029614552009401267496693144064107592433030506438440,\\
\gamma &= 551737151471267013665906013312108810638488413639246814149706\\
&\quad 947418249015319821682977158544996914977939922628908,\\
\mathfrak{m}(X) &= 11792 + 15441X - 25387X^2 + 11348X^3 + 20103X^4 + 25605X^5\\
&\quad -8716X^6 + 9091X^7 + 19039X^8 + 13855X^9 - 22021X^{10} - 15182X^{11}\\
&\quad -4543X^{12} + 1417X^{13} - 26776X^{14} + 11502X^{15}.
\end{aligned}
$$

Random Order m-ary Exponentiation

Michael Tunstall

Department of Computer Science, University of Bristol,
Merchant Venturers Building, Woodland Road,
Bristol BS8 1UB, United Kingdom
tunstall@cs.bris.ac.uk

Abstract. This paper describes a m-ary exponentiation algorithm where the radix-m digits of an exponent can be treated in a somewhat random order without using any more group operations than a standard right-to-left m-ary exponentiation. This paper demonstrates that the random order countermeasure, commonly used to protect implementations of secret key cryptographic algorithms, can be applied to public key cryptographic algorithms.

Keywords: Exponentiation algorithms, random order countermeasure, side channel analysis.

1 Introduction

Implementations of exponentiation algorithms in microprocessors need to be resistant to side channel analysis. For example, it has been demonstrated that a private key used in RSA [21] can be derived by observing a suitable side channel, such as power consumption [15] or electromagnetic emanations [11,20]. These attacks targeted implementations of the square and multiply algorithm, where an exponent is read bit-by-bit and a zero in the exponent results in a squaring operation, whereas a one results in a squaring operation followed by a multiplication. Bits of a private key can be seen directly if these two operations can be distinguished by observing a suitable side channel while an exponentiation is being computed.

The first proposed countermeasure was to always compute a squaring operation followed by a (possibly fake) multiplication for each bit of an exponent, referred to as the square and multiply *always* algorithm [10]. This algorithm has a large impact on the efficiency of the computation of an exponentiation. An alternative was proposed in [9] that proposed efficient algorithms with a fixed structure, i.e. a structure independent to the value of the exponent. A further suggestion was made in [8], where it was proposed that a squaring operation and a multiplication be rendered indistinguishable via a side channel.

More complex attacks are proposed in [15], where numerous acquisitions are taken and an attacker attempts to observes a statistical relationship between an observed side channel and an intermediate state. In order to prevent this class of

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 437–451, 2009.

attack individual variables are combined with random values generated for each instantiation of an algorithm [6].

In block ciphers the order in which variables are treated can also be randomised to augment the side channel resistance of an implementation [17]. In this paper an algorithm is proposed that allows an exponentiation to be computed in a random order, although not all orders will occur with with the same probability. This algorithm is intended to allow an exponentiation to be computed without the currently used blinding algorithms. However, the proposed algorithm could also be used to augment the security of an exponentiation implementation, and will inhibit attacks that allow operations to be distinguished from one acquisition. The proposed algorithm requires a large amount of memory. However, some modern smart card chips are using 32-bit architectures and one can expect to have around 4k of RAM available [4,18]. This would allow for the proposed algorithm to be implemented for elliptic curve cryptography, but it is unlikely to be possible for exponentiation in $(\mathbb{Z}/N\mathbb{Z})^*$.

The rest of this paper is organised as follows. In the next section, we review some methods for evaluating an exponentiation. In Section 3 we describe the methods of side channel analysis that could potentially be used to attack an exponentiation. The previously published uses of random values as a countermeasure to side channel analysis is described in Section 4. In Section 5 we detail an exponentiation algorithm where the digits can be treated in a random order. In Section 6 we describe how an attacker would try to derive information by side channel analysis. Finally, we conclude in Section 7.

2 Exponentiation Algorithms

2.1 The Square and Multiply Algorithm

The simplest algorithm for computing an exponentiation is the square and multiply algorithm. This is where an exponent is read left-to-right bit-by-bit, a zero results in a squaring operation being performed, and a one results in a squaring operation followed by a multiplication with the original message. This algorithm is detailed in Algorithm 1, where we define the function $\texttt{bit}(n, i)$ as a function returning the i-th bit of n. The input is an element x in a (multiplicatively written) group $\mathbb{G}$ and a positive ℓ-bit integer n; the output is the element $z = x^n$ in $\mathbb{G}$. This algorithm requires two group elements in memory and, on average, $1.5\,(\lfloor \log_2 n \rfloor - 1)$ group operations to compute x^n.

2.2 Left-to-Right m-Ary Exponentiation

In order to compute $z = x^n$ more efficiently it is possible to use an algorithm that is described in [13]. In this algorithm one precomputes some values that are small powers of x and the exponent is broken up into ℓ words in base m (this is called the m-ary expansion and ℓ is called the m-ary length). Typically, m is

Algorithm 1: The Square and Multiply Algorithm

Input: $x \in \mathbb{G}$, $n \geq 1$, ℓ the binary length of n (i.e. $2^{\ell-1} \leq n < 2^{\ell}$)
Output: $z = x^n$

$A \leftarrow x$
$R \leftarrow x$
for $i = \ell - 2$ **down to** 0 **do**
  $A \leftarrow A^2$
  if $(\texttt{bit}(n, i) \neq 0)$ **then** $A \leftarrow A \cdot R$
end
return A

chosen to be equal to 2^k, for some convenient value of k, to enable the relevant digits to simply be read from the exponent. The m-ary algorithm is shown in Algorithm 2, where we define the function $\texttt{digit}(n, i)$ as a function returning the i-th radix-m digit of n. This algorithm requires m group elements in memory and, on average, $\left(m + \frac{m-1}{m}\right)(\lfloor \log_m n \rfloor - 1) + m - 2$ group operations to compute x^n.

Algorithm 2: Left-to-Right m-ary Exponentiation Algorithm

Input: $x \in \mathbb{G}$, $n \geq 1$, ℓ the m-ary length of n
Output: $z = x^n$
Uses: A, $R[i]$ for $i \in \{1, 2, \ldots, m-1\}$

$R[1] \leftarrow x$
for $i \leftarrow 2$ **to** $m - 1$ **do** $R[i] \leftarrow R[i-1] \cdot x$
$b \leftarrow \texttt{digit}(n, \ell - 1)$
$A \leftarrow R[b]$
for $i = \ell - 2$ **down to** 0 **do**
  $A \leftarrow A^m$
  $b \leftarrow \texttt{digit}(n, i)$
  if $(b \neq 0)$ **then** $A \leftarrow A \cdot R[b]$
end
return A

2.3 Right-to-Left m-Ary Exponentiation

A right-to-left version of Algorithm 2 was described in [24]. This algorithm is detailed in Algorithm 3, and requires slightly more operations than the left-to-right algorithm. The structure of the algorithm is different as the effect of the value of each digit is not taken into account until the final stage of the algorithm. This algorithm requires $(m-1)+1 = m$ group elements in memory and, on average, $\left(\frac{m-1}{m} + m\right)(\lfloor \log_m n \rfloor - 1) + 2m - 3$ group operations to compute x^n.

Algorithm 3: Right-to-Left m-ary Exponentiation

Input: $x \in \mathbb{G}$, $n \geq 1$
Output: $z = x^n$
Uses: A, $R[j]$ for $j \in \{1, \ldots, m-1\}$

for $j = 1$ **to** $m - 1$ **do** $R[j] \leftarrow 1_{\mathbb{G}}$

$A \leftarrow x$
while $(n \geq m)$ **do**
  $d \leftarrow n \bmod m$
  if $(d \neq 0)$ **then** $R[d] \leftarrow R[d] \cdot A$
  $A \leftarrow A^m$
  $n \leftarrow \lfloor n/m \rfloor$
end
$R[n] \leftarrow R[n] \cdot A$

$A \leftarrow R[m-1]$
for $j = m - 2$ **down to** 1 **do**
  $R[j] \leftarrow R[j] \cdot R[j+1]$
  $A \leftarrow A \cdot R[j]$
end

return A

3 Side Channel Analysis

3.1 Simple Side Channel Analysis

The most basic form of side channel analysis is to simply inspect one acquisition of a suitable side channel, referred to as Simple Side Channel Analysis (SPA). This involves observing a suitable side channel whilst a computation is taking place. An attacker will then try and make deductions about what calculations are being performed based on these observations. If we consider the square and multiply algorithm, it has been shown that bit values of an exponent can be distinguished by observing a suitable side channel, such as the power consumption [15] or electromagnetic emanations [11,20].

3.2 Differential Side Channel Analysis

It has been demonstrated that in microprocessors the instantaneous power consumption is typically proportional to the Hamming weight of data being manipulated at a given point in time [5], and the same relationship has been observed in electromagnetic emanations [11,20]. This difference in Hamming weight was first exploited in [15] to attack block ciphers. This was extended in [5] to give a more complete analysis of the power consumption.

In this attack, an attacker acquires M power consumption traces (w_i for $i \in \{1, 2, \ldots, M\}$) during the computation of a cryptographic algorithm, and chooses one word, b, of an intermediate state generated while the acquisition is taking place. For a given hypothesis for a key value (or portion of the key) K

the Hamming weight of b is predicted and the correlation between this value and the instantaneous power consumption is calculated. A significant correlation will confirm the hypotheses, allowing an attacker to derive information on the key.

In order to attack an exponentiation, an attacker could use this to confirm hypotheses on an intermediate state of a multiplication at an arbitrary point in the computation to derive portions of the exponent. A significant correlation would indicate that the hypothesised exponent bits are correct.

4 Using Random Values as a Countermeasure

In this section we describe some of the countermeasures that can be used to prevent side channel analysis. We concentrate on countermeasures that are specific to exponentiation algorithms. The interested reader is referred to [16] for a discussion of countermeasures to side channel analysis in more general terms.

4.1 Blinding

The use of random values to modify the behaviour of an exponentiation has taken many different forms. One of the first proposals was to modify all the variables in a modular exponentiation [14]. If we consider the modular exponentiation $z = x^n \bmod m$, this could be implemented as

$$z = \left((x + r_1 \cdot m)^{n + r_2 \cdot \phi(m)} \bmod r_3 \cdot m \right) \bmod m \,,$$

where r_1, r_2 and r_3 are random values and ϕ is Euler's Totient function. Typically, the bit lengths of r_1, r_2 and r_3 are chosen to increase x, d and m by one word of the processor computing the exponentiation. The above equation is specific to $(\mathbb{Z}/N\mathbb{Z})^*$, but equivalent algorithms exist for exponentiation algorithms in other groups (e.g. elliptic curves over $\mathbb{F}_p$ and $\mathbb{F}_{2^q}$ [10]).

4.2 Randomised Algorithms

When implementing block ciphers one would combine masking with a random ordering. For example, if a given function were to be applied to a series of bytes, one would implement the function such that the bytes were treated in some random order. If an attacker were to try and predict a byte value occurring at a given point in time, the prediction would only be correct a small proportion of the time. This has a direct impact on the computed correlation coefficient, discussed in Section 3.2, since an attacker's prediction will often be incorrect.

An equivalent countermeasure for exponentiation algorithms has not previously been proposed in the literature, but other methods of randomising the computation of an exponentiation have been proposed. In this section we review some of these methods.

Random Digit Sizes

A right-to-left m-ary exponentiation algorithm is proposed in [23] where the radix of the digits of the exponent is varied during the computation of an exponentiation. The algorithm proceeds as described in Algorithm 3, but the radix of the digits of the exponent digits is chosen randomly between 2,3 and 5 whenever a new digit is required during the main loop of the algorithm.

Overlapping Exponent Digits

Another algorithm that modifies the exponent digits every time an exponentiation is computed is presented in [12]. When the exponent digits are read from the exponent the number if bits is extended such that the bits of one digit will overlap with the bits of the neighbouring digits. The digits are modified such that the sum of the effect of the overlapping digits is equivalent to the desired exponent. Given that there are numerous combinations of overlapping digits that are equivalent to the desired exponent, some bits of each digit can be randomly set each time the algorithm is executed.

5 Random Order Exponentiation

In this section we define a right-to-left m-ary exponentiation algorithm, where the radix-m digits of the exponent are treated in a random order. Before the algorithm is defined we demonstrate that, if enough memory is available, the digits of an exponent could be treated in an arbitrary order when using Algorithm 3.

Consider the right-to-left m-ary exponentiation algorithm to compute $z = x^n$. We can write the radix-m expansion of the exponent as $n = \sum_{i=0}^{\ell-1} d_i\, m^i$. Then

$$z = \prod_{\substack{0 \le i \le \ell-1 \\ d_i = 1}} x^{m^i} \cdot \prod_{\substack{0 \le i \le \ell-1 \\ d_i = 2}} x^{2 \cdot m^i} \cdots \prod_{\substack{0 \le i \le \ell-1 \\ d_i = j}} x^{j \cdot m^i} \cdots \prod_{\substack{0 \le i \le \ell-1 \\ d_i = m-1}} x^{(m-1) \cdot m^i}$$

$$= \prod_{j=1}^{m-1} (R[j])^j \quad \text{where } R[j] = \prod_{\substack{0 \le i \le \ell-1 \\ d_i = j}} x^{m^i} \ .$$

In Algorithm 3, each $R[j]$ for $j \in \{1, \ldots m-1\}$ is computed and then combined in the final loop to raise each $R[j]$ to the power of j and compute the product of the results, i.e. $\prod_j (R[j])^j$.

We can note that each $R[j]$ is the product of x^{m^i} for all $i \in \{0, \ldots, \ell-1\}$ where d_i is equal to j. This can be computed in any order if all the values x^{m^i} are known. Therefore, if it would be possible to precompute and store all the group elements x^{m^i}, for $i \in \{0, \ldots, \ell-1\}$, they could be multiplied with the relevant $R[j]$ in an arbitrary order. A worked example of this is shown in Appendix A. However, there would need to be enough memory available to store all ℓ group elements.

If we consider the bit lengths of variables in exponentiations that would be of use in cryptography, it is unlikely that all the values of x^{m^i}, for $i \in \{0, \ldots, \ell-1\}$, could be stored in the memory of a microprocessor.

However, if we assume that there is enough memory available to store r group elements, we can precompute and store x^{m^i}, for $i \in \{0, \ldots, r-1\}$. Suppose these are stored in an array

$$S = \{x, x^m, x^{m^2}, \ldots, x^{m^{r-1}}\},$$

and list the first r digits of the exponent as

$$D = \{d_0, d_1, d_2, \ldots, d_{r-1}\} \ .$$

If we initialise a set of registers $R[i]$, for $i \in \{1, \ldots, m-1\}$, to $1_{\mathbb{G}}$. We can treat the r values held in S and D in some arbitrary order, where, for each $j \in \{0, \ldots, r-1\}$, we compute $R[D[j]] = R[D[j]] \cdot S[j]$. The contents of $R[i]$, for $i \in \{1, \ldots, m-1\}$, will then contain the same group elements one would expect if the standard right-to-left m-ary exponentiation had treated the first r digits of the exponent (see Algorithm 3). One could then assign the next r values that would be required to continue computing the exponentiation to S and D, which then become

$$S = \{x^{m^r}, x^{m^{r+1}}, x^{m^{r+2}}, \ldots, x^{m^{2r-1}}\},$$

and

$$D = \{d_r, d_{r+1}, d_{r+2}, \ldots, d_{2r-1}\} \ .$$

These values could also be treated in some arbitrary order, as for for each $j \in \{0, \ldots, r-1\}$ we compute $R[D[j]] = R[D[j]] \cdot S[j]$. After which the values held in $R[i]$, for $i \in \{1, \ldots, m-1\}$, will be the same as if the standard right-to-left m-ary exponentiation has treated the first $2r$ digits of the exponent. This could be continued until all the digits of the exponent have been treated.

However, we can do better by noting that once $R[D[j]] = R[D[j]] \cdot S[j]$ has been computed, for a given $j \in \{0, \ldots, r-1\}$, then $D[j]$ and $S[j]$ are no longer required by the exponentiation algorithm. These values in memory can be safely overwritten. Consider again the initial state of S and D

$$S = \{x, x^m, x^{m^2}, \ldots, x^{m^{r-1}}\}, D = \{d_0, d_1, d_2, \ldots, d_{r-1}\} \ .$$

If we compute $R[D[j]] = R[D[j]] \cdot S[j]$, for some arbitrary $j \in \{0, \ldots, r-1\}$, we can replace $D[j]$ with d_r and $S[j]$ with x^{m^r}. If, for example, we take $j = 1$ then, after computing $R[D[1]] = R[D[1]] \cdot S[1]$, we can replace $D[1]$ and $S[1]$. That is,

$$S = \{x, x^{m^r}, x^{m^2}, \ldots, x^{m^{r-1}}\},$$

and

$$D = \{d_0, d_r, d_2, \ldots, d_{r-1}\} \ .$$

This could be repeated for another $D[j]$ and $S[j]$, for some arbitrary $j \in \{0, \ldots, r-1\}$, and the j-th values of S and D replaced with $x^{m^{r+1}}$ and

d_{r+1} respectively. For an ℓ-bit exponent this could be repeated $\ell - r$ times, at which point one would be unable to include any new digits in D. One could then compute $R[D[j]] = R[D[j]] \cdot S[j]$ for each $j \in \{0, \ldots, r-1\}$ without replacing any of the values in D or S. After which, all of the digits of the exponent, i.e. all d_i for $i \in \{0, \ldots, \ell - 1\}$, will have been treated. A worked example of this process is given in Appendix A.

An example of how this could be implemented to produce an exponentiation algorithm where the digits are treated in some random order is shown in Algorithm 4, where we define the function `RandomInteger`(x, y) as returning a random integer in the interval $[x, y]$.

In order to minimise the number of operations required it is necessary to keep track of which element of S contains the largest power of x, so that x^{m^i} can be computed from $x^{m^{i-1}}$ with a minimum, and constant, number of operations. In Algorithm 4 we use the variable γ as a pointer to the largest power of x present in S.

It is interesting to note that this algorithm does not require any more group operations than Algorithm 3 so the performance should remain unaffected. However, this does assume that random values can be generated instantly. The difference in performance between Algorithm 3 and Algorithm 4 will be the time required to generate $\ell - r$ random values.

Algorithm 4 requires significantly more memory than Algorithm 3, as $(m - 1) + r + 1 = m + r$ group elements need to be stored in memory. A further r radix-m digits will also need to be held in memory.

6 Security Analysis

In this section we describe how an attacker would attempt to derive digits of the exponent used in an implementation of Algorithm 4 by side channel analysis.

6.1 Simple Side Channel Analysis

Algorithm 4 is, to a certain extent, vulnerable to Simple Side Channel Analysis. It would be expected that an attacker would be able to detect when zero digits are treated. In order to counter this, the multiplication and squaring operations can be implemented such that they use identical code and, therefore, cannot be distinguished [8]. There are methods of statistically distinguishing a multiplication and a squaring operation based on the design [2] or the distribution of the bits of single-precision operations [3]. It is expected that similar attacks may be possible on a single acquisition using template attacks [7].

If an attacker is able to distinguish multiplications from squaring operations the amount of information available is limited. An attacker would be able to determine the total number of zero digits in an exponent, but only approximately where they are in an exponent. The position of the first zero digit in the exponent could be determined by taking enough acquisitions that it is possible to determine the earliest in an exponentiation that a zero digit is visible in a side

Algorithm 4: Random Order Right-to-Left m-ary Exponentiation

Input: $x \in \mathbb{G}$, $n \geq 1$, r number of values to store in memory.
Output: $z = x^n$
Uses: A, $R[j]$ for $j \in \{1, \ldots, m-1\}$, $D[i]$ for $i \in \{0, \ldots, r-1\}$, $S[i]$ for $i \in \{0, \ldots, r-1\}$

for $j = 1$ **to** $m - 1$ **do** $R[j] \leftarrow 1_{\mathbb{G}}$

$S[0] \leftarrow x$
for $i = 1$ **to** $r - 1$ **do** $S[i] \leftarrow S[i-1]^m$

for $i = 0$ **to** $r - 1$ **do**
    $D[i] \leftarrow n \bmod m$
    $n \leftarrow \lfloor n/m \rfloor$
end
$\gamma \leftarrow r - 1$

while $(n > 0)$ **do**
    $\tau =$ `RandomInteger`$(0, r-1)$
    if $D[\tau] \neq 0$ **then**
        $R[D[\tau]] \leftarrow R[D[\tau]] \cdot S[\tau]$
    end
    $S[\tau] \leftarrow S[\gamma]^m$
    $D[\tau] \leftarrow n \bmod m$
    $n \leftarrow \lfloor n/m \rfloor$
    $\gamma \leftarrow \tau$
end

for $i = r - 1$ **down to** 0 **do**
    $R[D[i]] \leftarrow R[D[i]] \cdot S[i]$
end

$A \leftarrow R[m-1]$
for $i = m - 2$ **down to** 1 **do**
    $R[i] \leftarrow R[i] \cdot R[i+1]$
    $A \leftarrow A \cdot R[i]$
end

return A

channel. However, it is unclear how an attacker would derive further zero digits unless they occur infrequently in an exponent, i.e. an attacker is able to locate the earliest point each zero digit is used during the computation of an exponentiation. This is unlikely to be the case, especially if the number of elements in D and S is of a similar size to the radix of the digits being taken from the exponent, in which case we would expect, statistically, there to be one zero digit in D most of the time.

In [19], Möller describes a recoding algorithm for m-ary exponentiation where each digit that is equal to zero is replaced with $-m$, and the next most significant digit is incremented by one. This leads to an exponent recoded with digits in the set $\{1, \ldots, m-1\} \cup \{-m\}$. An unsigned version of Möller's algorithm is described in [22] where the digits are recoded with digits in the set $\{1, \ldots, m\}$. Where

each zero digit is replaced with m and the next digit is decremented by one, which removes the need to compute a potentially costly inversion. Both of these algorithms render a subsequent m-ary exponentiation regular, and, therefore, resistant to simple side channel analysis.

6.2 Differential Side Channel Analysis

In order to conduct a side channel attack using differential side channel analysis, an attacker needs to construct hypotheses on data being manipulated at a specific point in time for each acquisition in a set of acquisitions. An attacker can then attempt to confirm these hypotheses by computing the correlation between them and each instant in time of during the acquisitions.

In this section we describe how differential side channel analysis could be applied to Algorithm 4. We assume that an attacker has acquired sufficient traces of a side channel to conduct a differential side channel attack, as described in Section 3.2, on Algorithm 4. We also assume that the exponent has been recoded such that the digits are in the set $\{1, \ldots, m\}$, as described above.

If an attacker were to try and conduct an differential side channel attack it would be assumed that the digit treated at the ℓ-th round is the $(\ell + r - 1)$-th digit of the exponent (counting from the right). This is because at this at this point the $(\ell + r - 1)$-th digit will just have been included into the digits from which the algorithm will randomly select a digit to treat and will occur at this point with the highest frequency.

This can be seen if we consider that once a digit has been included, it has a probability of $1/r$ of being used in the next round. It has the same probability of being used in the following round but this can only occur if the digit was not treated in the previous round, so the probability of the digit not being used in the first round but being used in the second round is $\left(1 - \frac{1}{r}\right)\frac{1}{r}$. In general, we can say that the probability of a digit being treated θ rounds after being included is $\frac{1}{r}\left(1 - \frac{1}{r}\right)^{\theta - 1}$. The highest probability occurs one round after a digit is included when $\theta = 1$.

In [5] it is pointed out that if the correlation coefficient of η independent bits amongst δ is calculated, a partial correlation still exists and its size can be predicted as a function of the coefficient that would be generated if all the bits of δ were included. This is given as:

$$\rho_{\eta/\delta} = \rho\sqrt{\frac{\eta}{\delta}}$$

where ρ is the correlation if everything is known and $\rho_{\eta/\delta}$ is the predicted partial correlation.

This also applies to η intermediate states being correctly predicted out of a total of δ. Therefore, in conducting a side channel attack against a random order exponentiation the correlation coefficient seen would be reduced to $\sqrt{1/r}$ of the correlation coefficient that would be seen if a non-randomised algorithm were under attack if R is known. This would indicate to an attacker which previously treated digits of the exponent are equal to the $(\ell + r - 1)$-th digit.

However, the state of R will not be known to an attacker. To conduct a differential side channel attack an attacker would be obliged to form hypotheses on the number and positions of the digits with the same value as the $(\ell+r-1)$-th digit and therefore predict the state of R. An attacker would then be obliged to generate a correlation trace from the acquired traces for all of the possible values of R.

The state of R will be different for each execution since the digits that have not been treated at the ℓ-th round will vary from one execution to another. An attacker would, therefore, be obliged to predict the most likely state of R and assume this is the state for every acquisition. This will further reduce the correlation coefficient visible via a differential side channel attack.

An attacker can, therefore, expect to be able to derive a correlation whose size is reduced to $\sqrt{1/r}$ of what one would be able to produce when attacking a naïve implementation in the first round of the algorithm. In subsequent rounds the largest correlation an attacker could hope to produce will diminish rapidly. It is not clear exactly how the correlation will diminish, and it is left as open problem, as to exactly how an attacker would attempt to derive the exponent via differential side channel analysis.

7 Conclusion

In this paper we present a method of computing an exponentiation where radix-m digits can be treated in a random order. The algorithm is intended to provide resistance to side channel analysis, and some informal arguments are given as to the side channel analysis resistance of this algorithm. However, we can note that not all the possible orderings of exponent digits will be equally likely.

Algorithm 4 will most likely find use as a supplement, rather than as a replacement, to the blinding countermeasure described in Section 4.1. This is because it may be possible to derive the digits used in an m-ary exponentiation from one trace using template attacks [7]. In the example given, $d + \phi(m)$ gives a value equivalent to the private key and could be used to start to make attempts at factoring m. If Algorithm 4 were also used, an attacker would have to test all the possible orderings of the exponent to find $d+\phi(m)$. This is an advantage over the previously proposed randomised exponentiation algorithms [12,23], described in Section 4.2, that would provide a value equivalent to the exponent if one were able to derive the digits of an exponent from one acquisition.

Another advantage over previously proposed countermeasures [12,23], is that the digit size can be chosen such that it evenly divides a computer word. It is, therefore, not necessary to read digits that will have bits on two computer words, which has implications for both security and efficiency.

As stated in the introduction, Algorithm 4 requires a large amount of memory, as Algorithm 4 requires $m+r$ group elements to be stored in memory. There are 32-bit secure microprocessors that have enough memory to allow exponentiation in $\mathbb{F}_p$ or $\mathbb{F}_{2^q}$ to be implemented [4,18]. More recently, processors with larger memories are being studied with regard to their side channel resistance [1], on

which one would be able to implement Algorithm 4 in $(\mathbb{Z}/N\mathbb{Z})^*$. This is unlikely to be possible on a smart card microprocessor unless a cryptographic coprocessor with a large amount of registers is included.

The side channel resistance of the algorithm proposed in this paper is only briefly analysed. There are some open problems that arise from this work.

The most obvious question is exactly how one would mount a side channel attack against Algorithm 4. Some brief details are given, but the magnitude of the correlation coefficient one would expect to be able to observe for a given r is not defined. Indeed, it remains to be shown what values of r would provide a suitable level of random ordering.

Another question is how one would attack Algorithm 4 if it were combined with other countermeasures. For example, one could easily combine Algorithm 4 with Walter's MIST algorithm [23] where the radix of the digits read from an exponent is also randomised.

Acknowledgements

The author would like to thank Elisabeth Oswald and Dan Page for their helpful comments related to the ideas presented in this paper. The author would also like to thank Steven Galbraith for his help in preparing the final version of this paper. The work described in this paper has been supported in part by the European Commission IST Programme under Contract IST-2002-507932 ECRYPT and EPSRC grant EP/F039638/1 "Investigation of Power Analysis Attacks".

References

1. Side-channel attack standard evaluation board (SASEBO), http://www.rcis.aist.go.jp/special/SASEBO
2. Akishita, T., Takagi, T.: Power analysis to ECC using differential power between multiplication and squaring. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 151–164. Springer, Heidelberg (2006)
3. Amiel, F., Feix, B., Tunstall, M., Whelan, C., Marnane, W.P.: Distinguishing multiplications from squaring operations. In: SAC 2008. LNCS. Springer, Heidelberg (2008)
4. ARM. SecurCore family, http://www.arm.com/products/CPUs/families/SecurCoreFamily.html
5. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004)
6. Chari, S., Jutla, C.S., Rao, J.R., Rohatgi, P.: Towards approaches to counteract power-analysis attacks. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 398–412. Springer, Heidelberg (1999)
7. Chari, S., Rao, J.R., Rohatgi, P.: Template attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 13–28. Springer, Heidelberg (2003)

8. Chevallier-Mames, B., Ciet, M., Joye, M.: Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. IEEE Transactions on Computers 53(6), 760–768 (2004)
9. Clavier, C., Joye, M.: Universal exponentiation algorithm. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 300–308. Springer, Heidelberg (2001)
10. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, Ç.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
11. Gandolfi, K., Mourtel, C., Olivier, F.: Electromagnetic analysis: Concrete results. In: Koç, Ç.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 251–261. Springer, Heidelberg (2001)
12. Itoh, K., Yajima, J., Takenaka, M., Torii, N.: DPA countermeasures by improving the window method. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 303–317. Springer, Heidelberg (2003)
13. Knuth, D.E.: The Art of Computer Programming, 2nd edn. Seminumerical Algorithms, vol. 2. Addison-Wesley, Reading (1981)
14. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
15. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
16. Mangard, S., Oswald, E., Popp, T.: Power Analysis Attacks — Revealing the Secrets of Smart Cards. Springer, Heidelberg (2007)
17. Messerges, T.S.: Power Analysis Attacks and Countermeasures for Cryptographic Algorithms. PhD thesis, University of Illinois, Chicago (2000)
18. MIPS-Technologies. SmartMIPS ASE, `http://www.mips.com/content/Products/`
19. Möller, B.: Securing elliptic curve point multiplication against side-channel attacks. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 324–334. Springer, Heidelberg (2001)
20. Quisquater, J.-J., Samyde, D.: Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In: Attali, S., Jensen, T. (eds.) E-smart 2001. LNCS, vol. 2140, pp. 200–210. Springer, Heidelberg (2001)
21. Rivest, R., Shamir, A., Adleman, L.M.: Method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
22. Vuillaume, C., Okeya, K.: Flexible exponentiation with resistance to side channel attacks. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006. LNCS, vol. 3989, pp. 268–283. Springer, Heidelberg (2006)
23. Walter, C.D.: MIST: An efficient, randomized exponentiation algorithm for resisting power analysis. In: Preneel, B. (ed.) CT-RSA 2002. LNCS, vol. 2271, pp. 53–66. Springer, Heidelberg (2002)
24. Yao, A.C.: On the evaluation of powers. SIAM Journal on Computing 5(1), 100–103 (1976)

A Worked Example

We wish to compute $z = x^n$, where n is set to 738530 (`B44E2` in hexadecimal) and the digits will be read from this exponent two bits at a time, i.e. $m = 4$. The digits and powers of x (variable A) computed in the main loop of Algorithm 3

are shown below. If enough memory were available then x^{m^i} for $i \in \{0, \ldots, 9\}$ could all be precomputed. Then we can set

$$S = \{x, x^4, x^{16}, x^{64}, x^{256}, x^{1024}, x^{4096}, x^{16384}, x^{65536}, x^{262144}\},$$

and

$$D = \{2, 0, 2, 3, 0, 1, 0, 1, 3, 2\} .$$

The exponentiation can be computed by treating the elements of S and D in an arbitrary order. We initially set $R[j]$ for $j \in \{1, 2, 3\}$ to $1_{\mathbb{G}}$. An arbitrary $j \in \{0, \ldots, 9\}$ is chosen, and we compute $R[D[j]] = R[D[j]] \cdot S[j]$ except when $D[j]$ is equal to zero when no operation is performed. This is repeated once for each possible value of j, which will result in:

$$\begin{aligned} R[1] &= S[5] \cdot S[7] = x^{1024}\, x^{16384} = x^{17408} \\ R[2] &= S[0] \cdot S[2] \cdot S[9] = x\, x^{16}\, x^{262144} = x^{262161} \\ R[3] &= S[3] \cdot S[8] = x^{64}\, x^{65536} = x^{65600} \end{aligned}$$

where $z = R[1] \cdot R[2]^2 \cdot R[3]^3 = x^{17408}\, x^{2 \cdot 262161}\, x^{3 \cdot 65600} = x^{738530}$, which is computed by the final loop in Algorithm 3.

If the above computation of $z = x^n$ were conducted using Algorithm 4, where we set $r = 6$ so that there is only enough memory to store six group elements in S. The initial values in memory would be x^{m^i} for $i \in \{0, \ldots, 5\}$, i.e.

$$S = \{x, x^4, x^{16}, x^{64}, x^{256}, x^{1024}\},$$

and the corresponding digits of the exponent would be

$$D = \{2, 0, 2, 3, 0, 1\} .$$

Again, initially set $R[j]$ for $j \in \{1, 2, 3\}$ to $1_{\mathbb{G}}$. An arbitrary $j \in \{0, \ldots, 5\}$ is chosen, and we compute $R[D[j]] = R[D[j]] \cdot S[j]$. As above, all $R[j]$ for $j \in \{1, 2, 3\}$ are initialised to $1_{\mathbb{G}}$. We will take $j = 3$ which give $S[3] = x^{64}$ and $D[3] = 3$, and we compute $R[3] = R[3] \cdot x^{64}$. The values in $S[3]$ and $D[3]$ are no longer required and can be replaced. We, therefore set $S[3] = x^{m^7} = x^{4096}$ and $D[3] = 0$. The values contained in memory would then be

$$S = \{x, x^4, x^{16}, x^{4096}, x^{256}, x^{1024}\},$$

and

$$D = \{2, 0, 2, 0, 0, 1\} .$$

Another arbitrary $j \in \{0, \ldots, 5\}$ can then be selected. We will take $j = 0$, which will mean we will compute $R[2] = R[2] \cdot x$. After which $S[0]$ and $D[0]$ can be replaced with x^{m^8} and 1 respectively, giving

$$S = \{x^{16384}, x^4, x^{16}, x^{2048}, x^{256}, x^{1024}\},$$

and

$$D = \{1, 0, 2, 0, 0, 1\} \ .$$

Next, we take $j = 4$. $D[4]$ is equal to zero, so no operation is conducted with $S[4]$, and these elements can be replaced with 3 and x^{m^9}, giving

$$S = \{x^{16384}, x^4, x^{16}, x^{4096}, x^{65536}, x^{1024}\} ,$$

and

$$D = \{1, 0, 2, 0, 3, 1\} \ .$$

We now take $j = 1$. Again, the chosen digit, $D[1]$, is equal to zero and no operation takes place. $S[1]$ and $D[1]$ can be replaced with $x^{m^{10}}$ and the last digit of the exponent, giving

$$S = \{x^{16384}, x^{262144}, x^{16}, x^{4096}, x^{65536}, x^{1024}\} ,$$

and

$$D = \{1, 2, 2, 0, 3, 1\} \ .$$

There are now no remaining digits that could be D, so there is no further need to select digits to be replaced. The remaining digits can be treated, where for each $j \in \{0, \ldots, 5\}$ we compute $R[D[j]] = R[D[j]] \cdot S[j]$ except when $D[j]$ is equal to zero when no operation is performed. This can be performed in an arbitrary order and will result in:

$$\begin{aligned} R[1] &= S[7] \cdot S[5] = x^{16384}\, x^{1024} = x^{17408} \\ R[2] &= S[0] \cdot S[9] \cdot S[2] = x\, x^{262144}\, x^{16} = x^{262161} \\ R[3] &= S[3] \cdot S[8] = x^{64}\, x^{65536} = x^{65600} \end{aligned}$$

This is exactly the same result as given where all the x^{m^i} for $i \in \{0, \ldots, 9\}$ are precomputed. The only difference being the order in which the x^{m^i}, for $i \in \{1, \ldots, 10\}$, are multiplied together. The final stage is the same as described above, since $z = R[1] \cdot R[2]^2 \cdot R[3]^3 = x^{17408}\, x^{2 \cdot 262161}\, x^{3 \cdot 65600} = x^{738530}$.

Jacobi Quartic Curves Revisited

Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson

Information Security Institute,
Queensland University of Technology, QLD, 4000, Australia
{h.hisil,kk.wong,g.carter,e.dawson}@qut.edu.au

Abstract. This paper provides new results about efficient arithmetic on Jacobi quartic form elliptic curves, $y^2 = dx^4 + 2ax^2 + 1$. With recent bandwidth-efficient proposals, the arithmetic on Jacobi quartic curves became solidly faster than that of Weierstrass curves. These proposals use up to 7 coordinates to represent a single point. However, fast scalar multiplication algorithms based on windowing techniques, precompute and store several points which require more space than what it takes with 3 coordinates. Also note that some of these proposals require $d = 1$ for full speed. Unfortunately, elliptic curves having 2-times-a-prime number of points, cannot be written in Jacobi quartic form if $d = 1$. Even worse the contemporary formulae may fail to output correct coordinates for some inputs. This paper provides improved speeds using fewer coordinates without causing the above mentioned problems. For instance, our proposed point doubling algorithm takes only 2 multiplications, 5 squarings, and no multiplication with curve constants when d is arbitrary and $a = \pm 1/2$.

Keywords: Efficient elliptic curve arithmetic, point multiplication, Jacobi model of elliptic curves.

1 Introduction

Cryptology as a computational science has been a driving force behind the arithmetic of elliptic curves in the past few decades. The demand for more speed led researchers to propose new formulae/algorithms/point-representations for several different elliptic curve models. However, the speed limitation for performing arithmetic on elliptic curves —like many other computational problems— is still an open question.

The historical roots of the topic dates back to late 18^{th} and early 19^{th} century: the time of Euler, Abel and Jacobi. An outline of the previous work restricted to the efficient arithmetic on Jacobi quartic curves is as follows. Chudnovsky and Chudnovsky [8] introduced the first inversion-free algorithms for performing group operations using a weighted projective point representation. Billet and Joye [2] used Jacobi quartic curves for protection against side-channel attacks with a point addition speed record for that time of $10\mathbf{M}+3\mathbf{S}+1\mathbf{D}$. In this paper, $\mathbf{M}$ stands for a field multiplication; $\mathbf{S}$ for a field squaring; $\mathbf{D}$ for a multiplication by a curve constant; $\mathbf{I}$ for a field inversion. This notation is borrowed from

C. Boyd and J. González Nieto (Eds.): ACISP 2009, LNCS 5594, pp. 452–468, 2009.

[6]. Duquesne [11] improved this operation count by $1\mathbf{M} + 1\mathbf{S}$ with a variant of Billet/Joye unified point addition algorithm. Duquesne's method converts the base point in weighted projective coordinates to a new point representation with 4 coordinates, performs the scalar multiplication within the new coordinate system, and outputs the final result in original weighted projective coordinates. Duquesne's improvement was followed by additional results in [5], [17], and [18]. However, the latter proposals tend to use more space —up to 7 coordinates per point— despite their speed advantage. Further disadvantages have already been mentioned in the abstract. We will extend our discussion on these aspects in §2.

In this paper, we carefully optimize the arithmetic of Jacobi quartic curves targeting more efficient scalar multiplication operations. Our proposal performs faster and uses less space than [11], [5], [17], and [18].

The paper is organized as follows. A review of Jacobi quartic curves is given in §2. Efficient algorithms/formulae/point-representations are introduced in §3, §4, and §5. Implementation timings are given in §6. We draw our conclusions in §7.

2 Background

This section gives definitions for Jacobi quartic curves. Some of the results involved in this section are analogous to our earlier work [19].

Let K be a field with $\mathrm{char}(K) \neq 2$. A Jacobi quartic form elliptic curve over K is defined by

$$E_{J,d,a}\colon y^2 = dx^4 + 2ax^2 + 1$$

where $a, d \in K$ with $\Delta = 256d(a^2 - d)^2 \neq 0$. The j-invariant of this curve is given by $64d^{-1}(a^2 - d)^{-2}(a^2 + 3d)^3 \in K$.

Billet and Joye remark in [2] that any elliptic curve, E/K, can be written as $E_{J,d,a}/K$ if $E(K)$ has an element of order 2 and provide the transformations between a Weierstrass elliptic curve $y^2 = x^3 + ax + b$ of even order and a projective Jacobi quartic curve.

We first review the most popular addition formulae. Let $(x_1, y_1), (x_2, y_2) \in E_{J,d,a}(K)$. Assuming that (x_3, y_3) is defined we have $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ where

$$x_3 = \frac{x_1y_2 + y_1x_2}{1 - dx_1^2x_2^2}, \tag{1}$$

$$y_3 = \frac{(y_1y_2 + 2ax_1x_2)(1 + dx_1^2x_2^2) + 2dx_1x_2(x_1^2 + x_2^2)}{(1 - dx_1^2x_2^2)^2}. \tag{2}$$

Formula (1) appears in one of Euler's historical works [13]. Formula (2) is adapted from [2]. With this selection of the algebraic expressions, the identity element becomes the point $(0, 1)$. The negative of a point (x, y) is $(-x, y)$. The point $(0, -1)$ is of order 2. $E_{J,d,a}$ is non-singular provided that $\Delta \neq 0$. On the other hand, there is a singular point at infinity —denoted by $(0\colon 1\colon 0)$— in the projective closure of $E_{J,d,a}$ if and only if d is a square in K. Resolving this singularity yields two more points of order 2 (both are written as $(0\colon 1\colon 0)$).

The following lemma shows that formulae (1) and (2) are complete if d is not a square in K. The term *complete* is used to emphasize that addition formulae are defined for all inputs, see [6].

Lemma 1. *Let $d, x_1, x_2 \in K$. Assume that d is non-square. Then $dx_1^2x_2^2 \neq 1$.*

Proof. Suppose that $dx_1^2x_2^2 = 1$. So $d, x_1, x_2 \neq 0$. But then $d = (1/(x_1x_2))^2$. □

Lemma 1 is similar to Theorem 3.3 of [6]. In the case of Jacobi quartic form, the statement of the lemma and its proof is shorter.

We can prevent $dx_1^2x_2^2 = 1$ even if d is a square in K. Lemma 2 states a sufficient condition. This lemma and its proof are similar to Corollary 1 in [19].

Lemma 2. *Let $a, d, x_1, y_1, x_2, y_2 \in K$ such that $d(a^2-d) \neq 0$. Assume that $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points of odd order on $E_{J,d,a}$. Then $1 - dx_1^2x_2^2 \neq 0$.*

We provide a proof in Appendix A. By elementary group theory, multiplying a point of even order with some power of 2 yields a point of odd order.

More formulae. Jacobi elliptic functions give rise to many addition formulae, cf. [24], [8], and [2]. The original reference is [20]. The following formulae are congruent via the algebraic relations $sn(\cdot)^2+cn(\cdot)^2 = 1$ and $k^2sn(\cdot)^2+dn(\cdot)^2 = 1$.

$$\mathrm{sn}(u_1+u_2) = \frac{\mathrm{sn}(u_1)\mathrm{cn}(u_2)\mathrm{dn}(u_2) + \mathrm{cn}(u_1)\mathrm{dn}(u_1)\mathrm{sn}(u_2)}{1-k^2\mathrm{sn}(u_1)^2\mathrm{sn}(u_2)^2} \tag{3}$$

$$= \frac{\mathrm{sn}(u_1)^2 - \mathrm{sn}(u_2)^2}{\mathrm{sn}(u_1)\mathrm{cn}(u_2)\mathrm{dn}(u_2) - \mathrm{cn}(u_1)\mathrm{dn}(u_1)\mathrm{sn}(u_2)}. \tag{4}$$

To see the congruence, either take the arithmetic cross product and write

$$\mathrm{sn}(u_1)^2\mathrm{cn}(u_2)^2\mathrm{dn}(u_2)^2 - \mathrm{cn}(u_1)^2\mathrm{dn}(u_1)^2\mathrm{sn}(u_2)^2 =$$

$$(\mathrm{sn}(u_1)^2 - \mathrm{sn}(u_2)^2)(1-k^2\mathrm{sn}(u_1)^2\mathrm{sn}(u_2)^2)$$

—the rest follows when $\mathrm{cn}(\cdot)$ is replaced with $1-\mathrm{sn}(\cdot)^2$ and $\mathrm{dn}(\cdot)$ is replaced with $1-k^2\mathrm{sn}(\cdot)$— or simply run the Maple script

```
> simplify(expand(
      JacobiSN(u1 + u2, k) - (
          (JacobiSN(u1, k)*JacobiCN(u2, k)*JacobiDN(u2, k) +
           JacobiSN(u2, k)*JacobiCN(u1, k)*JacobiDN(u1, k))/
          (1 - k^2*JacobiSN(u1, k)^2*JacobiSN(u2, k)^2))));
0
```

Formula (3) is analogous to (1) as pointed out in [2] via the relation $(x_i, y_i) = (\mathrm{sn}(u_i), \mathrm{cn}(u_i)\mathrm{dn}(u_i))$. Similarly, the analog of (4) is given by

$$x_3 = \frac{x_1^2 - x_2^2}{x_1y_2 - y_1x_2}. \tag{5}$$

This formula is not defined if $(x_1, y_1) = (x_2, y_2)$. This formula is independent of a and d. There are several other ways to derive (5). For instance, one may use

the strategy applied in [19] for the derivation of dedicated addition formulae on twisted Edwards curves. Formula (5) is of minimal total degree. Therefore, the Monagan/Pearce minimal total degree algorithm in [23] can be used to derive this same formula (or maybe an alternative formula of same total degree if there exists one) departing from (1) or any other valid formula.

Lemma 2 can be rewritten for (5). The proof is similar to the proof of Lemma 2.

Lemma 3. *Let $a, d, x_1, y_1, x_2, y_2 \in K$ such that $d(a^2 - d) \neq 0$. Assume that $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ are points of odd order on $E_{J,d,a}$. Assume that $P \neq Q$. Then $x_1y_2 - y_1x_2 \neq 0$.*

The choices for computing y_3 are abundant. For instance, each of the following formulae computes y_3 (except for a few exceptional inputs):

$$y_3 = \frac{(x_1^2 + x_2^2)(y_1y_2 - 2ax_1x_2) - 2x_1x_2(1 + dx_1^2x_2^2)}{(x_1y_2 - y_1x_2)^2}, \tag{6}$$

$$y_3 = \frac{(x_1y_2 - y_1x_2)(y_1y_2 + 2ax_1x_2) + 2(x_2y_2 - x_1y_1)}{(x_1y_2 - y_1x_2)(1 - dx_1^2x_2^2)}, \tag{7}$$

$$y_3 = \frac{x_1y_1(2 + 2ax_1^2 - y_1^2) - x_2y_2(2 + 2ax_2^2 - y_2^2)}{(x_1y_2 - y_1x_2)(1 - dx_1^2x_2^2)}. \tag{8}$$

Relevant Work. Efficient implementations often use inversion-free point doubling and point addition formulae. To the best of our knowledge all such proposals for Jacobi quartic curves reference from weighted projective coordinates which represent the points as $(X : Y : Z)_{[1,2,1]} = (\lambda X : \lambda^2 Y : \lambda Z)$ for all nonzero $\lambda \in K$.

Chudnovsky and Chudnovsky [8] proposed two inversion-free point addition and two inversion-free point doubling formulae using a slightly different quartic equation given by

$$E_{\tilde{J},a',b'} : y^2 = x^4 + a'x^2 + b'$$

and using weighted projective coordinates. The formulae in [8, (4.10) on p.418] are analogous to (5) with the minor detail that the identity element is moved to one of two points at infinity. The arithmetic of this curve is similar to that of $E_{J,a,b}$ due to the symmetry in the right hand side of the weighted projective equations $Y^2 = X^4 + a'X^2Z^2 + b'Z^4$ and $Y^2 = dX^4 + 2aX^2Z^2 + Z^4$.

Billet and Joye [2] proposed a faster inversion-free unified addition algorithm using (1) and (2). The term *unified* is used to emphasize that point addition formulae remain valid when two input points are identical, see [10, §29.1.2]. By Lemma 2, the Billet/Joye algorithm is complete if d is not a square in K and needs $10\mathbf{M} + 3\mathbf{S} + 1\mathbf{D}$. We remark that no faster way of inversion-free general point addition is known to date in $(X : Y : Z)_{[1,2,1]}$ coordinates. It remains an open question whether it is possible to speed up the addition in weighted $(X : Y : Z)_{[1,2,1]}$ coordinates. Nevertheless, the speed of the Billet/Joye algorithm was improved by Duquesne in [11] with the proposal of $(X^2 : XZ : Z^2 : Y)$ coordinates. Duquesne's variant addition algorithm needs $9\mathbf{M} + 2\mathbf{S} + 1\mathbf{D}$ saving $1\mathbf{M} + 1\mathbf{S}$ over the Billet/Joye algorithm by using slightly more space to

represent the points. Bernstein and Lange [5] extended this representation to $(X:Y:Z:X^2:2XZ:Z^2)$ and $(X:Y:Z:X^2:2XZ:Z^2:X^2+Z^2)$ saving an extra $\mathbf{M}-\mathbf{S}$ (i.e. **M**-**S** trade-off) over Duquesne's algorithm. A more detailed overview of these algorithms and operation counts can be found in the original papers or in the *Explicit-Formulas Database* (EFD) [5] which also reports $1\mathbf{M}+9\mathbf{S}+1\mathbf{D}$ doubling algorithm by Bernstein/Lange, $2\mathbf{M}+6\mathbf{S}+2\mathbf{D}$ doubling algorithm by Hisil/Carter/Dawson, and $2\mathbf{M}+6\mathbf{S}+1\mathbf{D}$ doubling algorithm by Feng/Wu in $(X:Y:Z)_{[1,2,1]}$. Duquesne coordinates $(X^2:XZ:Z^2:Y)_{[2,2,2,2]}$ use less space than redundant coordinates but need special treatment in the scalar multiplication to obtain the original coordinates $(X:Y:Z)_{[1,2,1]}$ of the final result. The original representation as $(X:Y:Z)_{[1,2,1]}$ in [2] uses even less space however this representation has to date been slower than the redundant coordinates.

Hisil, Wong, Carter, and Dawson [17] introduced new point doubling formulae together with a fast point doubling algorithm costing only $3\mathbf{M}+4\mathbf{S}$ in $(X:Y:Z:X^2:Z^2)$ if $d=1$. Roughly at the same time essentially the same formulae were independently derived by Feng and Wu, see EFD [5]. These formulae were adapted to $(X:Y:Z:X^2:2XZ:Z^2)$ coordinates with the same operation count in EFD.

Later Hisil, Wong, Carter, and Dawson [18] introduced (for the case $d=1$) new unified addition formulae which use $7\mathbf{M}+3\mathbf{S}+1\mathbf{D}$ in $(X:Y:Z:X^2:Z^2:XZ)$ and $7\mathbf{M}+4\mathbf{S}+1\mathbf{D}$ in $(X:Y:Z:X^2:Z^2)$ and newer doubling formulae which need $2\mathbf{M}+5\mathbf{S}+1\mathbf{D}$ in $(X:Y:Z:X^2:Z^2)$ and $(X:Y:Z:X^2:Z^2:XZ)$.

The redundant representations such as

$$(X:Y:Z:X^2:2XZ:Z^2:X^2+Z^2)_{[1,2,1,2,2,2,2]},$$
$$(X:Y:Z:X^2:2XZ:Z^2)_{[1,2,1,2,2,2]},$$
$$(X:Y:Z:X^2:Z^2:X^2+Z^2)_{[1,2,1,2,2,2]},$$
$$(X:Y:Z:X^2:Z^2)_{[1,2,1,2,2]},$$
$$(X:Y:Z:X^2:Z^2:XZ)_{[1,2,1,2,2,2]}$$

help in the development of faster algorithms for performing point operations and their overall performance only slightly differs from each other. However, they all share one serious drawback. They need more space for storing the points in comparison to earlier proposals. Despite the speed advantage of these coordinate systems, the large space requirement makes the practical use of Jacobi quartic curves questionable since windowing techniques in scalar multiplication algorithms precompute and store several points.

We aim to solve this disadvantage in subsequent sections. Furthermore we propose a faster doubling algorithms.

Even more formulae. All of the affine formulae given in this section involve inversions in K. In cryptographic applications K is finite and computing inverses in a finite field can be very costly in comparison to the multiplication and addition operations. In §3 and §4 we will introduce inversion-free formulae which are simply derived by the adaptation of affine formulae to a suitable projective point representation. However, formulae given so far do not necessarily lead to

the fastest inversion free algorithms to perform the basic operations; point doubling and point addition. Therefore we propose new affine point doubling and point addition formulae to assist following sections.

Let $(x_1, y_1) \in E_{J,d,a}(K)$. Assuming that (x_3, y_3) is defined we have $2(x_1, y_1) = (x_3, y_3)$ where

$$x_3 = \mu x_1, \tag{9}$$

$$y_3 = \mu(\mu - y_1) - 1 \tag{10}$$

with $\mu = 2y_1/(2 + 2ax_1^2 - y_1^2)$. In the derivations of (9) and (10) we were inspired by the results in [6] and [18]. If d is a square in K then these point doubling formulae work for all inputs i.e. (x_3, y_3) is defined for all inputs. If d is a square in K then there exist two points at infinity of order two. The double of these points is $(0, 1)$. If $(2 + 2ax_1^2 - y_1^2) = 0$ then (x_1, y_1) is a point of order 4 and the output is a point at infinity.

Further let $(x_2, y_2) \in E_{J,d,a}(K)$. Assuming that (x_3, y_3) is defined we have $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ where x_3 is defined as in (5) and

$$y_3 = \frac{(x_1 - x_2)^2}{(x_1y_2 - y_1x_2)^2}\left(y_1y_2 - 2ax_1x_2 + 1 + dx_1^2x_2^2\right) - 1. \tag{11}$$

In addition, if $s \in K$ such that $d = s^2$ then we can also write

$$y_3 = \frac{(1 + sx_1x_2)^2}{(1 - dx_1^2x_2^2)^2}(y_1y_2 + 2ax_1x_2 + sx_1^2 + sx_2^2) - sx_3^2 \tag{12}$$

where x_3 is given by (1).

Formulae (11) and (12) compute the same result as (2), (6), (7), and (8). Formula (12) is defined if $(x_1, y_1) = (x_2, y_2)$. Formula (11) is not defined if $(x_1, y_1) = (x_2, y_2)$. Both formulae are incomplete, i.e. (x_3, y_3) is not defined for a few special inputs.

3 Homogeneous Projective Coordinates, $\mathcal{Q}$

Projective coordinates are used as basic tools in designing inversion-free algorithms to carry out group arithmetic on elliptic curves. In the case of Jacobi quartic curves, we consider homogenous projective coordinates $(X : Y : Z)_{[1,1,1]}$ for efficiency purposes for the first time. From now on we omit the informative subscript $[1, 1, 1]$ for these coordinates.

In homogeneous projective coordinates each point (x, y) is represented by the triplet $(X : Y : Z)$ which satisfies the projective equation $Y^2Z^2 = dX^4 + 2aX^2Z^2 + Z^4$ and corresponds to the affine point $(X/Z, Y/Z)$ with $Z \neq 0$. The identity element is represented by $(0 : 1 : 1)$. The negative of $(X : Y : Z)$ is $(-X : Y : Z)$. In the following subsection, we provide efficient point doubling formulae.

3.1 Point Doubling in $\mathcal{Q}$

We remark that the fastest-so-far three-coordinate point doubling algorithm in [5, `dbl-2007-fw-2`] costs $2\mathbf{M} + 6\mathbf{S} + 1\mathbf{D}$ in $(X : Y : Z)_{[1,2,1]}$. We also remark that this algorithm assumes $d = 1$.

In this section we introduce new efficient doubling formulae. Given $(X_1 : Y_1 : Z_1)$ with $Z_1 \neq 0$ the point doubling can be performed as $2(X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ where

$$\begin{aligned} X_3 &= 2X_1Y_1(2Z_1^2 + 2aX_1^2 - Y_1^2), \\ Y_3 &= 2Y_1^2(Y_1^2 - 2aX_1^2) - (2Z_1^2 + 2aX_1^2 - Y_1^2)^2, \\ Z_3 &= (2Z_1^2 + 2aX_1^2 - Y_1^2)^2. \end{aligned} \tag{13}$$

We obtained these formulae from (9). With these formulae a point doubling takes $2\mathbf{M} + 5\mathbf{S} + 1\mathbf{D}$ where $1\mathbf{D}$ is multiplication with a. These formulae do not depend on d. Therefore keeping d arbitrary has no effect on the cost of (13).

If $a = \pm -1/2$ then a point doubling takes $2\mathbf{M} + 5\mathbf{S}$. Note $2a$ can be rescaled to -1 via the map $(x, y) \mapsto (x/\sqrt{-2a}, y)$ provided that $\sqrt{-2a} \in K$. This map transforms the curve $y^2 = dx^4 + 2ax^2 + 1$ to $y^2 = (d/(4a^2))x^4 - x^2 + 1$. Alternatively, a curve having $a = -1/2$ can be selected without rescaling. We comment that similar arguments apply to the case $a = 1/2$.

For justifications and more on operation counts see `DBL-Q-x` in the Appendix B. The proposed algorithm(s) are faster than other three-coordinate point doubling algorithms for Jacobi quartic curves.

3.2 Point Addition in $\mathcal{Q}$

It would be convenient to give an efficient point addition algorithm for the projective coordinates. However, the fastest point addition algorithms that we could design were quite uncompetitive in comparison to the previous proposals in other coordinate systems. Therefore, we leave this as an open question. As a remedy to this, we will introduce fast point addition algorithms on a new coordinate system in the next section and show that the new point addition algorithms can be efficiently combined with the fast doubling algorithms from §3.1.

4 Extended Homogeneous Projective Coordinates, $\mathcal{Q}^e$

Jacobi quartic curves not only have a rich body of formulae but also allow us to use various efficient point representations. We have already given a detailed review in §2.

This section introduces a new representation of points on Jacobi quartic curves and provides efficient algorithms to perform group operations on Jacobi quartic form elliptic curves. Some of the results in this section are analogous to our earlier work [19].

In the new system a point $(x, y) \in E_{J,d,a}(K)$ is represented by $(X : Y : T : Z)$ where $T = X^2/Z$ and $(X : Y : T : Z)_{[1,1,1,1]} = (\lambda X : \lambda Y : \lambda T : \lambda Z) = (x : y : x^2 : 1)$ for all nonzero $\lambda \in K$. From now on we omit the informative subscript $[1, 1, 1, 1]$ for these coordinates. Each quadruplet $(X : Y : T : Z)$ simultaneously satisfy the homogeneous projective equations

$$\begin{cases} X^2 - TZ = 0 \\ Y^2 - dT^2 - 2aX^2 - Z^2 = 0 \end{cases} \tag{14}$$

or simply the homogeneous projective equation

$$Y^2Z^2 = dX^4 + 2aX^2Z^2 + Z^4 \tag{15}$$

where T is omitted in the latter case. A point representation $(X : Y : Z)$ satisfying (15) can be converted to the new coordinates by computing $(XZ : YZ : X^2 : Z^2)$ with $Z \neq 0$ in $1\mathbf{M} + 3\mathbf{S}$. This coordinate system will be denoted by $\mathcal{Q}^e$ in the rest of the paper. The identity element is represented by the quadruplet $(0 : 1 : 0 : 1)$. The negative of $(X : Y : T : Z)$ is $(-X : Y : T : Z)$.

4.1 Dedicated Point Doubling in $\mathcal{Q}^e$

Given $(X_1 : Y_1 : T_1 : Z_1)$ with $Z_1 \neq 0$ satisfying (15), point doubling can be performed as $2(X_1 : Y_1 : T_1 : Z_1) = (X_3 : Y_3 : T_3 : Z_3)$ where X_3, Y_3, and Z_3 are the same as (13) and

$$T_3 = (2X_1Y_1)^2. \tag{16}$$

If $a = -1/2$ then the operations can be performed with a $0\mathbf{M} + 8\mathbf{S}$ algorithm. Again the formulae do not depend on d. Therefore keeping d arbitrary has no effect on the cost of (13). There are many $\mathbf{M}/\mathbf{S}$ trade-offs possible for doubling in $\mathcal{Q}^e$ when a is arbitrary or when $a = -1/2$. For justifications and more on operation counts see `DBL-Qe-x` in Appendix B.

In §5, we will mix $\mathcal{Q}^e$ with $\mathcal{Q}$ to benefit from faster doubling algorithms proposed in §3.1. In §5, we will use point doubling from this section to develop a double-and-add algorithm.

4.2 Dedicated Point Addition in $\mathcal{Q}^e$

Given $(X_1 : Y_1 : T_1 : Z_1)$ and $(X_2 : Y_2 : T_2 : Z_2)$ with $Z_1 \neq 0$ and $Z_2 \neq 0$ and $(X_1 : Y_1 : T_1 : Z_1) \neq (X_2 : Y_2 : T_2 : Z_2)$, a dedicated addition can be performed as $(X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2) = (X_3 : Y_3 : T_3 : Z_3)$ where

$$\begin{aligned}
X_3 &= (X_1Y_2 - Y_1X_2)(T_1Z_2 - Z_1T_2),\\
Y_3 &= (Y_1Y_2 - 2aX_1X_2)(T_1Z_2 + Z_1T_2) - 2X_1X_2(Z_1Z_2 + dT_1T_2),\\
T_3 &= (T_1Z_2 - Z_1T_2)^2,\\
Z_3 &= (X_1Y_2 - Y_1X_2)^2.
\end{aligned} \tag{17}$$

We derived these formulae using (5) and (6) in §2. Without any assumptions on the curve constants, Y_3 can alternatively be written as

$$Y_3 = (T_1Z_2 + Z_1T_2 - 2X_1X_2)\big(Y_1Y_2 - 2aX_1X_2 + Z_1Z_2 + dT_1T_2\big) - Z_3. \tag{18}$$

We obtained these formulae from (11). If $a = -1/2$ then the dedicated addition costs $7\mathbf{M}+3\mathbf{S}+2\mathbf{D}$ with the use of (18). For justifications and more on operation counts see `ADD-Qe-x` in Appendix B.

4.3 Unified Point Addition in $\mathcal{Q}^e$

Given $(X_1 : Y_1 : T_1 : Z_1)$ and $(X_2 : Y_2 : T_2 : Z_2)$ with $Z_1 \neq 0$ and $Z_2 \neq 0$, a unified addition can be performed as $(X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2) = (X_3 : Y_3 : T_3 : Z_3)$ where

$$\begin{aligned} X_3 &= (X_1Y_2 + Y_1X_2)(Z_1Z_2 - dT_1T_2), \\ Y_3 &= (Y_1Y_2 + 2aX_1X_2)(Z_1Z_2 + dT_1T_2) + 2dX_1X_2(T_1Z_2 + Z_1T_2), \\ T_3 &= (X_1Y_2 + Y_1X_2)^2, \\ Z_3 &= (Z_1Z_2 - dT_1T_2)^2. \end{aligned} \tag{19}$$

These formulae are analogous to (11) and (12) hence complete[1] by Lemma 1 if d is not a square in K.

Let $s \in K$ such that $d = s^2$. Alternatively, we can write

$$Y_3 = (Z_1Z_2 + dT_1T_2 + 2sX_1X_2)(Y_1Y_2 + 2aX_1X_2 + sT_1Z_2 + sZ_1T_2) - sT_3. \tag{20}$$

We obtained this formula from (12) and following the derivation notes in [18, §2.1]. In this case, the addition is still unified. However, the completeness is lost. Nevertheless, logical checks can be eliminated if the inputs are selected as indicated in Lemma 2. As indicated before these algorithms do not strictly require $d = 1$.

For justifications and more on operation counts see `UADD-Qe-x` in Appendix B. The new representation is solidly faster than the representation in [2]. The new representation can be equally fast as (or even faster than) the representation [11]. The special treatment in [11] for obtaining the original coordinates is also removed since $(X_3 : Y_3 : T_3 : Z_3)$ satisfies the homogeneous projective Jacobi quartic curve. The new representation can be equally fast as the representations in [5], [17], and [18]. However this is achieved by using only 4 coordinates rather than 5, 6, or 7 coordinates.

5 Mixed Homogeneous Projective Coordinates, $\mathcal{Q} + \mathcal{Q}^e$

The construction in this section is the same as [19, §4.3] and is closely linked with [9]. Therefore, we only give a brief outline of the technique. The details can be extracted from the original papers.

Most of the efficient scalar multiplication implementations are based on a suitable combination of signed integer recoding (such as NAF, MOF), fast precomputation and left-to-right sliding fractional-windowing techniques. The resulting algorithm is doubling intensive. Roughly for each bit of the scalar one doubling is performed. Additions are accessed less frequently. Excluding the additions used in the precomputation phase, approximately $l/(w+1)$ additions are needed where l is the number of bits in the scalar and w is the window length. w is used to control space consumption and optimize the total running time.

[1] If d is not a square in K then the point $(0 : 1 : 0)$ is not defined over K and should be omitted though it seems to satisfy the curve equation (15).

An abstract view of the scalar multiplication is composed of several repeated-doublings each followed by a single addition. In our specific case, these operations are performed in the following way:

(i) If a point doubling is followed by another point doubling, use $\mathcal{Q} \leftarrow 2\mathcal{Q}$.
(ii) If a point doubling is followed by a point addition, use
   1. $\mathcal{Q}^e \leftarrow 2\mathcal{Q}$ for the point doubling step; followed by,
   2. $\mathcal{Q} \leftarrow \mathcal{Q}^e + \mathcal{Q}^e$ for the point addition step.

Suppose that a repeated-doubling phase is composed of m doublings. In (i), $m-1$ successive doublings in $\mathcal{Q}$ are performed with the fastest `DBL-Q-x` algorithm explained in §3.1 and given in Appendix B. In (ii), the remaining doubling is merged with the single addition phase to yield a combined double-and-add step; a similar approach to [12]. To perform the double-and-add operation we first compute the doubling step with the fastest `DBL-QtoQe-x` algorithm explained in §4.1 and given in Appendix B. This algorithm is suitable to compute $\mathcal{Q}^e \leftarrow 2\mathcal{Q}$ since the inputs are only composed of the coordinates X, Y, Z and the output is still produced in $\mathcal{Q}^e$. We then perform the addition in $\mathcal{Q}^e$ using `ADD-Qe-2` which is explained in §4.2 and given in Appendix B but output only the coordinates of $\mathcal{Q}$. Note that the last operation of `ADD-Qe-2` (i.e. $T_3 \leftarrow T_3^2$) can be confidently removed to save $1\mathbf{S}$ since the result is in $\mathcal{Q}$ (not $\mathcal{Q}^e$).

If we use `DBL-Q-1` for repeated doubling operations and a combination of `DBL-QtoQe-1` and `ADD-Qe-2` for double-and-add operations then we need only $2\mathbf{M} + 5\mathbf{S}$ for each doubling and we effectively need $((8\mathbf{S}) + (8\mathbf{M} + 2\mathbf{S} + 2\mathbf{D} - 1\mathbf{S})) - (2\mathbf{M} + 5\mathbf{S}) = 6\mathbf{M} + 4\mathbf{S} + 2\mathbf{D}$ for each addition step.

We should note that the precomputed points are kept in $\mathcal{Q}^e$ which is composed of 4 coordinates rather than 3. On the other hand, we do not need 5, 6, or 7 coordinates as is the case in [5], [17], and [18].

By lemma 3, all logical checks to handle exceptional inputs can be eliminated if the base point is of odd order.

6 Experimental Results

This section provides implementation timings for single-variable-point-single-variable-scalar elliptic curve scalar multiplication. We have used a single core of Intel Core 2 Duo (E6550) processor in our experimentations.

Finite field operations. Following the implementation notes from [15] and [14], we have written a hand-crafted finite field layer using `x86-64` instruction set and GCC extended inline assembly. We have designed our field arithmetic layer to serve for fields $\mathbb{F}_p$ where p is of the form $2^{256} - c$ such that c is smaller than or equal to 64 bits. In our experimentations we have fixed $c = 587$.

Elliptic curve operations. We have selected `Q-DBL-2` as the doubling algorithm and a combination of `Q-DBL-2` and `Qe-ADD-2` as the double-and-add algorithm. This decision is due to the fact that the cost of additions is not so negligible in 64 bit applications. This was previously discussed in [15].

Scalar multiplication algorithm. We have implemented Algorithm 3.38 in [16] by modifying Steps 4.3 and Steps 4.4 as we discussed in §5.

Integer recoding. We have used Avanzi's w-LtoR integer recoding algorithm [1]. We have determined $w = 5$ to be the optimal window length in our implementation. We have not incorporated fractional windowing techniques [22] to our implementation following the comments in [14].

Lookup table. To accommodate the 5-LtoR technique $3P, 5P, \ldots, 15P$ are precomputed by the sequence of operations $2P, 2P + P, 2P + 3P, \ldots, 2P + 13P$. A new precomputation strategy in [21] is of interest for implementation. We have not implemented this approach yet. In our implementation $\mathbf{I}/\mathbf{M} \approx 121$. Therefore we have not normalized the precomputed values following the analysis [3]. Also following the same reference, we have derived and implemented double and add algorithms in $\mathcal{Q}^e$ with $Z = 1$. These special operations save time in the precomputation.

Table 1 summarizes measured average clock cycles for primitive operations for a single-variable-point-single-variable-scalar multiplication on $E_{J,-1/2,d}$ for some fixed d.

Table 1. 256-bit scalar multiplication on Intel Core 2 Duo (E6550)

Operation	Cycles
Precomputation	17,000
Main loop	345,000
Normalization	14,000
Total	376,000

We should warn the reader that we have detected these cycle counts with our local benchmarking tools. Unfortunately, we have not yet integrated our implementation to the commonly accepted toolkit SUPERCOP, a benchmarking framework within eBACS, the benchmarking project of ECRYPT II [4]. Therefore we do not claim verifiability at this stage.

We should also warn the reader that our implementation is a variable-single-point-variable-single-scalar multiplication. In the case where the base point is fixed the timings can be dramatically improved by using Algorithm 3.44 or Algorithm 3.45 in [16]. Indeed such an approach was used in [14] for Diffie-Hellman key pair generation where the base point is fixed. Note also that our implementation does not incorporate the Galbraith-Lin-Scott (GLS) homomorphism [14] which has been recently shown to yield faster results.

In our implementation, a scalar multiplication on $E_{J,-1/2,d}$ takes approximately $1162\mathbf{M} + 1110\mathbf{S} + 102\mathbf{D}$. In addition, there are approximately 1796 calls to faster field operations (such as addition, subtraction, division by 2, etc.).

As the comparative part of our work, we have also covered Weierstrass ($a = -3$) and twisted Edwards ($a = -1$) curves in our implementation. For the twisted Edwards implementation we have followed [19, §4.3]. We have used doubling

formulae from [7]. For the Weierstrass implementation we have collected the most efficient formulae from EFD [5]. In our implementation, a scalar multiplication on the Weierstrass curve $y^2 = x^3 - 3x + b$ for some fixed b with $w = 5$ —optimum— takes approximately $1598\mathbf{M} + 1156\mathbf{S} + 0\mathbf{D}$. In addition, there are approximately 2896 calls to faster field operations (such as addition, subtraction, division by 2, etc.). In our implementation, a scalar multiplication on the twisted Edwards curve $-x^2 + y^2 = 1 + dx^2y^2$ for some fixed d with $w = 6$ —optimum— takes approximately $1202\mathbf{M} + 969\mathbf{S} + 0\mathbf{D}$. In addition, there are approximately 2025 calls to faster field operations (such as addition, subtraction, division by 2, etc.). We have not tested the performance of formulae in [18] yet.

Table 2 summarizes measured average clock cycles for a single-variable-point-single-variable-scalar multiplication on different representations of elliptic curves.

Table 2. 256-bit scalar multiplication on Intel Core 2 Duo (E6550)

Curve	Cycles
Weierstrass ($a = -3$), Jacobian	418,000
Jacobi quartic ($a = -1/2$), $\mathcal{Q} + \mathcal{Q}^e$	376,000
Twisted Edwards ($a = -1$), $\mathcal{E} + \mathcal{E}^e$	362,000

In this implementation Jacobi quartic curves runs significantly faster than Weierstrass curves and slightly slower than twisted Edwards curves.

7 Conclusion

We introduced new results for performing arithmetic on Jacobi quartic curves.

In §2, we proved that earlier formulae (1) and (2) in the literature are complete provided that d is a not a square in the underlying field K. We explored several affine formulae some being known to date and some being new.

In §3 and §4, we carefully selected the most suitable affine formulae and then with suitable point representations converted them to projective form. In this context, for the first time we used homogeneous projective coordinates on Jacobi quartic curves for efficiency purposes and introduced new and faster doubling algorithms. In addition, we introduced a new point representation namely extended homogeneous projective coordinates, $\mathcal{Q}^e$, for Jacobi quartic curves. This coordinate system allows very efficient point addition operations using fewer coordinates in comparison to recent proposals for Jacobi quartic curves.

In §6 we reported our experimental results using state-of-art formulae for Jacobi quartic, twisted Edwards, and Weierstrass curves. In our implementation Jacobi quartic curves are approximately 20% faster than Weierstrass curves.

With our proposal Jacobi quartic curves can provide similar speeds to twisted Edwards curves in variable-point-and-variable-scalar multiplications. The point doubling on a Jacobi quartic curve is slightly faster than that of Edwards curves. Note that doubling is the most frequently accessed elliptic curve group operation in variable-point-and-variable-scalar multiplications. On the other hand,

Jacobi quartic curves seem to be slower on the double-and-add operation (§5) in comparison to twisted Edwards curves. In overall timings for variable-single-point-variable-single-scalar multiplication Jacobi quartic curves are competitive with (but slightly slower than) twisted Edwards curves. It should be noted here that there are elliptic curves of order 2-times-a-prime where our methods are applicable with the use of the Jacobi quartic curves with $a = -1/2$. However, such curves cannot be written (over the base field) in twisted Edwards form.

References

1. Avanzi, R.M.: A note on the signed sliding window integer recoding and its left-to-right analogue. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 130–143. Springer, Heidelberg (2004)
2. Billet, O., Joye, M.: The Jacobi model of an elliptic curve and side-channel analysis. In: Fossorier, M.P.C., Høholdt, T., Poli, A. (eds.) AAECC 2003. LNCS, vol. 2643, pp. 34–42. Springer, Heidelberg (2003)
3. Bernstein, D.J., Lange, T.: Analysis and optimization of elliptic-curve single-scalar multiplication. In: Finite Fields and Applications Fq8. Contemporary Mathematics, American Mathematical Society, vol. 461, pp. 1–18 (2008)
4. Bernstein, D.J., Lange, T.: eBACS: ECRYPT benchmarking of cryptographic systems (2008), http://bench.cr.yp.to
5. Bernstein, D.J., Lange, T.: Explicit-formulas database, http://www.hyperelliptic.org/EFD
6. Bernstein, D.J., Lange, T.: Faster addition and doubling on elliptic curves. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 29–50. Springer, Heidelberg (2007)
7. Bernstein, D.J., Birkner, P., Joye, M., Lange, T., Peters, C.: Twisted Edwards curves. In: Vaudenay, S. (ed.) AFRICACRYPT 2008. LNCS, vol. 5023, pp. 389–405. Springer, Heidelberg (2008)
8. Chudnovsky, D.V., Chudnovsky, G.V.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. Advances in Applied Mathematics 7(4), 385–434 (1986)
9. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In: Ohta, K., Pei, D. (eds.) ASIACRYPT 1998. LNCS, vol. 1514, pp. 51–65. Springer, Heidelberg (1998)
10. Cohen, H., Frey, G. (eds.): Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, Boca Raton (2005)
11. Duquesne, S.: Improving the arithmetic of elliptic curves in the Jacobi model. Information Processing Letters 104(3), 101–105 (2007)
12. Eisenträger, K., Lauter, K., Montgomery, P.L.: Fast elliptic curve arithmetic and improved Weil pairing evaluation. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 343–354. Springer, Heidelberg (2003)
13. Euler, L.: De integratione aequationis differentialis $m\,dx/\sqrt{1-x^4} = n\,dy/\sqrt{1-y^4}$. Novi Commentarii Academiae Scientiarum Petropolitanae 6, 37–57 (1761); Translated from the Latin by Langton, S.G. On the integration of the differential equation $m\,dx/\sqrt{1-x^4} = n\,dy/\sqrt{1-y^4}$, http://home.sandiego.edu/~langton/eell.pdf
14. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: EUROCRYPT 2009. LNCS, vol. 5479, pp. 518–535. Springer, Heidelberg (2009)

15. Gaudry, P., Thomé, E.: The mpFq library and implementing curve-based key exchanges. In: SPEED 2007, pp.49–64 (2007), http://www.loria.fr/~gaudry/publis/mpfq.pdf
16. Hankerson, D., Menezes, A.J., Vanstone, S.A.: Guide to Elliptic Curve Cryptography. Springer, New York (2003)
17. Hisil, H., Carter, G., Dawson, E.: New formulae for efficient elliptic curve arithmetic. In: Srinathan, K., Rangan, C.P., Yung, M. (eds.) INDOCRYPT 2007. LNCS, vol. 4859, pp. 138–151. Springer, Heidelberg (2007)
18. Hisil, H., Wong, K.K., Carter, G., Dawson, E.: Faster group operations on elliptic curves. In: Australasian Information Security Conference (AISC 2009), Wellington, New Zealand (January 2009); Conferences in Research and Practice in Information Technology (CRPIT), vol. 98, pp. 7–19 (2009)
19. Hisil, H., Wong, K.K., Carter, G., Dawson, E.: Twisted Edwards curves revisited. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 326–343. Springer, Heidelberg (2008)
20. Jacobi, C.G.J.: Fundamenta nova theoriae functionum ellipticarum. Sumtibus Fratrum Borntræger (1829)
21. Longa, P., Gebotys, C.: Novel precomputation schemes for elliptic curve cryptosystems. In: ACNS 2009. LNCS. Springer, Heidelberg (to appear, 2009)
22. Möller, B.: Improved techniques for fast exponentiation. In: Lee, P.J., Lim, C.H. (eds.) ICISC 2002. LNCS, vol. 2587, pp. 298–312. Springer, Heidelberg (2003)
23. Monagan, M., Pearce, R.: Rational simplification modulo a polynomial ideal. In: ISSAC 2006, pp. 239–245. ACM, New York (2006)
24. Whittaker, E.T., Watson, G.N.: A Course of Modern Analysis. Cambridge University Press, Cambridge (1927)

A Appendix

Proof (Lemma 2). Points at infinity (over the extension of K where they exist) are of order 2. Assume that P and Q are of odd order. Thus, P, Q and $P+Q$ cannot be the points at infinity. Since the formulae (1) and (2) are complete provided that the points at infinity are not involved, the denominator $1-dx_1^2x_2^2$ must be nonzero.

An algebraic approach is as follows. Suppose that $1-dx_1^2x_2^2=0$. Then $x_1, x_2 \neq 0$ and we can write $x_1^2 = 1/(dx_2^2)$.

Suppose that $P = \pm Q$. Then $1-dx_1^2x_2^2 = 1-dx_1^4 = 1-dx_2^4 = 0$. It follows that $R = 2P$ and $S = 2Q$ are points at infinity. Therefore P and Q must be of order 4 which contradicts the hypothesis. From now on we assume $P \neq \pm Q$.

We now have $1-dx_1^4 \neq 0$ and $1-dx_2^4 \neq 0$. Using the relations $x_1^2 = 1/(dx_2^2)$, $y_2^2 = dx_2^4 + 2ax_2^2 + 1$ and formulae (1) and (2) we get

$$x(R)^2 = \frac{(2x_1y_1)^2}{(1-dx_1^4)^2} = \frac{2(1/(dx_2^2))(d(1/(dx_2^2))^2 + 2a(1/(dx_2^2)) + 1)}{(1-d(1/(dx_2^2))^2)^2} = \frac{(2x_2y_2)^2}{(1-dx_2^4)^2} = x(S)^2,$$

$$\begin{aligned} y(R) &= \frac{(1+dx_1^4)(y_1^2+2ax_1^2)+4dx_1^4}{(1-dx_1^4)^2} \\ &= \frac{(1+d(1/(dx_2^2))^2)((d(1/(dx_2^2)^2+2a(1/(dx_2^2))+1)+2a(1/(dx_2^2)))+4d(1/(dx_2^2))^2}{(1-d(1/(dx_2^2))^2)^2} \\ &= \frac{(1+dx_2^4)(y_2^2+2ax_2^2)+4dx_2^4}{(1-dx_2^4)^2} = y(S). \end{aligned}$$

Hence, $R = \pm S$. But then $R \mp S = 2P \mp 2Q = 2(P \mp Q) = (0, 1)$. It follows that $P \mp Q$ is a point of order 2 since $P \neq \pm Q$.

Now either P is a point of even order or Q is a point of even order or both P and Q are points of even order. All conditions contradict the hypothesis. In conclusion $1 - dx_1^2 x_2^2 \neq 0$. □

B Appendix

These algorithms are optimized in a way that X_1-X_2-X_3, Y_1-Y_2-Y_3, T_1-T_2-T_3, and Z_1-Z_2-Z_3 are allowed to be the same registers. The algorithm uses t_i as temporary registers. **a** stands for an addition or a subtraction or a multiplication by 2 or a division by 2.

$2\mathbf{M} + 5\mathbf{S} + 7\mathbf{a}$, `DBL-Q-1`, assumes $a = -1/2$, $(X_3 : Y_3 : Z_3) = 2(X_1 : Y_1 : Z_1)$; $t_1 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $t_1 \leftarrow t_1^2$, $X_3 \leftarrow X_3 + Y_3$, $t_1 \leftarrow t_1 - X_3$, $Z_3 \leftarrow 2Z_3$, $Y_3 \leftarrow X_3 \cdot Y_3$, $Y_3 \leftarrow 2Y_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow t_1 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$.

$3\mathbf{M} + 4\mathbf{S} + 4\mathbf{a}$, `DBL-Q-2`, assumes $a = -1/2$, $(X_3 : Y_3 : Z_3) = 2(X_1 : Y_1 : Z_1)$; $t_1 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Y_3 \leftarrow Y_3 \cdot X_3$, $X_3 \leftarrow Z_3 - X_3$, $Z_3 \leftarrow X_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow t_1 \cdot X_3$.

$2\mathbf{M} + 5\mathbf{S} + 1\mathbf{D} + 8\mathbf{a}$, `DBL-Q-3`, $(X_3 : Y_3 : Z_3) = 2(X_1 : Y_1 : Z_1)$; $t_1 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $t_1 \leftarrow t_1^2$, $t_1 \leftarrow t_1 - X_3$, $t_1 \leftarrow t_1 - Y_3$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow Y_3 + X_3$, $Z_3 \leftarrow 2Z_3$, $Y_3 \leftarrow X_3 \cdot Y_3$, $Y_3 \leftarrow 2Y_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow t_1 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$.

$3\mathbf{M} + 4\mathbf{S} + 1\mathbf{D} + 4\mathbf{a}$, `DBL-Q-4`, $(X_3 : Y_3 : Z_3) = 2(X_1 : Y_1 : Z_1)$; $t_1 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Y_3 \leftarrow Y_3 \cdot X_3$, $X_3 \leftarrow Z_3 - X_3$, $Z_3 \leftarrow X_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow t_1 \cdot X_3$.

$0\mathbf{M} + 8\mathbf{S} + 13\mathbf{a}$, `DBL-Qe-1`, `DBL-QtoQe-1`, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $T_3 \leftarrow T_3^2$, $X_3 \leftarrow X_3 + Y_3$, $T_3 \leftarrow T_3 - X_3$, $Z_3 \leftarrow 2Z_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow T_3 + Z_3$, $T_3 \leftarrow T_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3 - Z_3$, $Z_3 \leftarrow 2Z_3$, $Y_3 \leftarrow 2Y_3$, $Y_3 \leftarrow Y_3^2$, $Y_3 \leftarrow Y_3 + T_3$, $Y_3 \leftarrow Y_3 - Z_3$, $T_3 \leftarrow 2T_3$.

$1\mathbf{M} + 7\mathbf{S} + 9\mathbf{a}$, `DBL-Qe-2`, `DBL-QtoQe-2`, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $T_3 \leftarrow T_3^2$, $X_3 \leftarrow X_3 + Y_3$, $T_3 \leftarrow T_3 - X_3$, $Z_3 \leftarrow 2Z_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow T_3 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $T_3 \leftarrow T_3^2$, $Y_3 \leftarrow 2Y_3$, $Y_3 \leftarrow Y_3^2$, $Y_3 \leftarrow Y_3 + T_3$, $Y_3 \leftarrow Y_3/2$, $Y_3 \leftarrow Y_3 - Z_3$.

$2\mathbf{M} + 6\mathbf{S} + 6\mathbf{a}$, `DBL-Qe-3`, `DBL-QtoQe-3`, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow T_3 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $T_3 \leftarrow T_3^2$, $Y_3 \leftarrow Y_3^2$, $Y_3 \leftarrow Y_3 + T_3$, $Y_3 \leftarrow Y_3/2$, $Y_3 \leftarrow Y_3 - Z_3$.

$3\mathbf{M} + 5\mathbf{S} + 4\mathbf{a}$, `DBL-Qe-4`, `DBL-QtoQe-4`, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Y_3 \leftarrow Y_3 \cdot X_3$, $X_3 \leftarrow Z_3 - X_3$, $Z_3 \leftarrow X_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow X_3 \cdot T_3$, $T_3 \leftarrow T_3^2$.

$0\mathbf{M} + 8\mathbf{S} + 2\mathbf{D} + 14\mathbf{a}$, `DBL-Qe-5`, `DBL-QtoQe-5`, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $T_3 \leftarrow T_3^2$, $T_3 \leftarrow T_3 - X_3$, $t_1 \leftarrow T_3 - Y_3$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $Z_3 \leftarrow 2Z_3$, $Z_3 \leftarrow Z_3 - X_3$, $T_3 \leftarrow t_1^2$, $X_3 \leftarrow t_1 + Z_3$, $X_3 \leftarrow X_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3 - Z_3$, $Z_3 \leftarrow 2Z_3$, $t_1 \leftarrow kT_3$, $T_3 \leftarrow 2T_3$, $Y_3 \leftarrow 2Y_3$, $Y_3 \leftarrow Y_3^2$, $Y_3 \leftarrow Y_3 + t_1$, $Y_3 \leftarrow Y_3 - Z_3$.

$1\mathbf{M} + 7\mathbf{S} + 1\mathbf{D} + 12\mathbf{a}$, `DBL-Qe-6`, `DBL-QtoQe-6`, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $t_1 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $t_1 \leftarrow t_1^2$, $t_1 \leftarrow t_1 - X_3$, $t_1 \leftarrow t_1 - Y_3$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $Z_3 \leftarrow 2Z_3$, $Y_3 \leftarrow X_3 \cdot Y_3$, $Z_3 \leftarrow Z_3 - X_3$, $T_3 \leftarrow t_1^2$, $X_3 \leftarrow t_1 + Z_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow 2Y_3$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow X_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3 - Z_3$, $X_3 \leftarrow X_3/2$.

$1\mathbf{M} + 7\mathbf{S} + 2\mathbf{D} + 10\mathbf{a}$, `DBL-Qe-7`, `DBL-QtoQe-7`, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $t_1 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $t_1 \leftarrow t_1^2$, $t_1 \leftarrow t_1 - X_3$, $t_1 \leftarrow t_1 - Y_3$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $Z_3 \leftarrow 2Z_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow t_1 \cdot Z_3$, $T_3 \leftarrow t_1^2$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow 2Y_3$, $Y_3 \leftarrow Y_3^2$, $t_1 \leftarrow kT_3$, $Y_3 \leftarrow Y_3 + t_1$, $Y_3 \leftarrow Y_3/2$, $Y_3 \leftarrow Y_3 - Z_3$.

$2\mathbf{M} + 6\mathbf{S} + 1\mathbf{D} + 8\mathbf{a}$, DBL-Qe-8, DBL-QtoQe-8, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 + Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $T_3 \leftarrow T_3^2$, $T_3 \leftarrow T_3 - X_3$, $T_3 \leftarrow T_3 - Y_3$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow Y_3 + X_3$, $Z_3 \leftarrow 2Z_3$, $Y_3 \leftarrow X_3 \cdot Y_3$, $Y_3 \leftarrow 2Y_3$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow T_3 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $T_3 \leftarrow T_3^2$.

$2\mathbf{M}+6\mathbf{S}+2\mathbf{D}+6\mathbf{a}$, DBL-Qe-9, DBL-QtoQe-9, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $t_1 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Z_3 \leftarrow Z_3 - X_3$, $X_3 \leftarrow t_1 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$, $T_3 \leftarrow t_1^2$, $Y_3 \leftarrow Y_3^2$, $t_1 \leftarrow kT_3$, $Y_3 \leftarrow Y_3 + t_1$, $Y_3 \leftarrow Y_3/2$, $Y_3 \leftarrow Y_3 - Z_3$.

$3\mathbf{M} + 5\mathbf{S} + 1\mathbf{D} + 4\mathbf{a}$, DBL-Qe-10, DBL-QtoQe-10, $(X_3 : Y_3 : T_3 : Z_3) = 2(X_1 : Y_1 : T_1 : Z_1)$; $T_3 \leftarrow X_1 \cdot Y_1$, $X_3 \leftarrow X_1^2$, $Y_3 \leftarrow Y_1^2$, $Z_3 \leftarrow Z_1^2$, $X_3 \leftarrow kX_3$, $X_3 \leftarrow X_3 + Y_3$, $X_3 \leftarrow X_3/2$, $Y_3 \leftarrow Y_3 \cdot X_3$, $X_3 \leftarrow Z_3 - X_3$, $Z_3 \leftarrow X_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow T_3 \cdot X_3$, $T_3 \leftarrow T_3^2$.

$7\mathbf{M} + 3\mathbf{S} + 2\mathbf{D} + 19\mathbf{a}$, ADD-Qe-1, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow dT_2$, $t_2 \leftarrow t_2 + Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_1 \cdot T_2$, $T_3 \leftarrow T_1 \cdot Z_2$, $t_1 \leftarrow t_1 - T_3$, $t_3 \leftarrow dt_2$, $t_1 \leftarrow t_1 - t_3$, $t_3 \leftarrow T_3 + t_2$, $T_3 \leftarrow T_3 - t_2$, $Z_3 \leftarrow X_1 - Y_1$, $t_2 \leftarrow X_2 + Y_2$, $Z_3 \leftarrow Z_3 \cdot t_2$, $t_2 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $Z_3 \leftarrow Z_3 - t_2$, $Z_3 \leftarrow Z_3 + Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2t_2$, $t_1 \leftarrow t_3 - t_1$, $Y_3 \leftarrow Y_3 + t_2$, $Y_3 \leftarrow t_1 \cdot Y_3$, $X_3 \leftarrow Z_3 + T_3$, $X_3 \leftarrow X_3^2$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow X_3 - Z_3$, $T_3 \leftarrow T_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3/2$.

$8\mathbf{M} + 2\mathbf{S} + 2\mathbf{D} + 15\mathbf{a}$, ADD-Qe-2, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow dT_2$, $t_2 \leftarrow t_2 + Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_1 \cdot T_2$, $T_3 \leftarrow T_1 \cdot Z_2$, $t_1 \leftarrow t_1 - T_3$, $t_3 \leftarrow dt_2$, $t_1 \leftarrow t_1 - t_3$, $t_3 \leftarrow T_3 + t_2$, $T_3 \leftarrow T_3 - t_2$, $Z_3 \leftarrow X_1 - Y_1$, $t_2 \leftarrow X_2 + Y_2$, $Z_3 \leftarrow Z_3 \cdot t_2$, $t_2 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $Z_3 \leftarrow Z_3 - t_2$, $Z_3 \leftarrow Z_3 + Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2t_2$, $t_1 \leftarrow t_3 - t_1$, $Y_3 \leftarrow Y_3 + t_2$, $Y_3 \leftarrow t_1 \cdot Y_3$, $X_3 \leftarrow Z_3 \cdot T_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $T_3 \leftarrow T_3^2$.

$7\mathbf{M} + 3\mathbf{S} + 3\mathbf{D} + 19\mathbf{a}$, ADD-Qe-3, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow dT_2$, $t_2 \leftarrow t_2 + Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_1 \cdot T_2$, $T_3 \leftarrow T_1 \cdot Z_2$, $t_1 \leftarrow t_1 - T_3$, $t_3 \leftarrow dt_2$, $t_1 \leftarrow t_1 - t_3$, $t_3 \leftarrow T_3 + t_2$, $T_3 \leftarrow T_3 - t_2$, $Z_3 \leftarrow X_1 - Y_1$, $t_2 \leftarrow X_2 + Y_2$, $Z_3 \leftarrow Z_3 \cdot t_2$, $t_2 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $Z_3 \leftarrow Z_3 - t_2$, $Z_3 \leftarrow Z_3 + Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2t_2$, $t_1 \leftarrow t_3 - t_1$, $t_2 \leftarrow kt_2$, $Y_3 \leftarrow Y_3 + t_2$, $Y_3 \leftarrow t_1 \cdot Y_3$, $X_3 \leftarrow Z_3 + T_3$, $X_3 \leftarrow X_3^2$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $X_3 \leftarrow X_3 - Z_3$, $T_3 \leftarrow T_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3/2$.

$8\mathbf{M} + 2\mathbf{S} + 3\mathbf{D} + 15\mathbf{a}$, ADD-Qe-4, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow dT_2$, $t_2 \leftarrow t_2 + Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_1 \cdot T_2$, $T_3 \leftarrow T_1 \cdot Z_2$, $t_1 \leftarrow t_1 - T_3$, $t_3 \leftarrow dt_2$, $t_1 \leftarrow t_1 - t_3$, $t_3 \leftarrow T_3 + t_2$, $T_3 \leftarrow T_3 - t_2$, $Z_3 \leftarrow X_1 - Y_1$, $t_2 \leftarrow X_2 + Y_2$, $Z_3 \leftarrow Z_3 \cdot t_2$, $t_2 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $Z_3 \leftarrow Z_3 - t_2$, $Z_3 \leftarrow Z_3 + Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2t_2$, $t_1 \leftarrow t_3 - t_1$, $t_2 \leftarrow kt_2$, $Y_3 \leftarrow Y_3 + t_2$, $Y_3 \leftarrow t_1 \cdot Y_3$, $X_3 \leftarrow Z_3 \cdot T_3$, $Z_3 \leftarrow Z_3^2$, $Y_3 \leftarrow Y_3 - Z_3$, $T_3 \leftarrow T_3^2$.

$7\mathbf{M} + 3\mathbf{S} + 1\mathbf{D} + 18\mathbf{a}$, UADD-Qe-1, assumes $d = 1$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $t_1 \leftarrow t_1 - t_2$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $t_3 \leftarrow T_3 - X_3$, $t_3 \leftarrow t_3 - Y_3$, $T_3 \leftarrow t_3^2$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2X_3$, $t_1 \leftarrow t_1 + t_2$, $t_2 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - t_2$, $Y_3 \leftarrow Y_3 \cdot t_1$, $Y_3 \leftarrow Y_3 - T_3$, $X_3 \leftarrow t_3 + Z_3$, $X_3 \leftarrow X_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3 - Z_3$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3/2$.

$8\mathbf{M} + 2\mathbf{S} + 1\mathbf{D} + 14\mathbf{a}$, UADD-Qe-2, assumes $d = 1$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $t_1 \leftarrow t_1 - t_2$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $t_3 \leftarrow T_3 - X_3$, $t_3 \leftarrow t_3 - Y_3$, $T_3 \leftarrow t_3^2$, $Y_3 \leftarrow Y_3 + t_1$, $t_1 \leftarrow 2X_3$, $t_1 \leftarrow t_1 + t_2$, $t_2 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - t_2$, $Y_3 \leftarrow Y_3 \cdot t_1$, $Y_3 \leftarrow Y_3 - T_3$, $X_3 \leftarrow t_3 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$.

$7\mathbf{M} + 3\mathbf{S} + 5\mathbf{D} + 18\mathbf{a}$, UADD-Qe-3, assumes $d = s^2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow sT_1$, $t_2 \leftarrow sT_2$, $T_3 \leftarrow t_1 t_2$, $t_1 \leftarrow t_1 + Z_1$, $t_2 \leftarrow t_2 + Z_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $t_1 \leftarrow t_1 - t_2$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $t_3 \leftarrow T_3 - X_3$, $t_3 \leftarrow T_3 - Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $T_3 \leftarrow t_3^2$, $t_1 \leftarrow 2s$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow t_1 + t_2$, $t_2 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - t_2$, $Y_3 \leftarrow Y_3 \cdot t_1$, $t_1 \leftarrow sT_3$, $Y_3 \leftarrow Y_3 - t_1$, $X_3 \leftarrow t_3 + Z_3$, $X_3 \leftarrow X_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3 - Z_3$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3/2$,

$8\mathbf{M} + 2\mathbf{S} + 5\mathbf{D} + 14\mathbf{a}$, UADD-Qe-4, assumes $d = s^2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow sT_1$, $t_2 \leftarrow sT_2$, $T_3 \leftarrow t_1 \cdot t_2$, $t_1 \leftarrow t_1 + Z_1$, $t_2 \leftarrow t_2 + Z_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $t_1 \leftarrow t_1 - t_2$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $T_3 \leftarrow T_3 - X_3$, $t_3 \leftarrow T_3 - Y_3$, $Y_3 \leftarrow Y_3 + t_1$, $T_3 \leftarrow t_3^2$, $t_1 \leftarrow 2s$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow t_1 + t_2$, $t_2 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - t_2$, $Y_3 \leftarrow Y_3 \cdot t_1$, $t_1 \leftarrow sT_3$, $Y_3 \leftarrow Y_3 - t_1$, $X_3 \leftarrow t_3 \cdot Z_3$, $Z_3 \leftarrow Z_3^2$.

$8\mathbf{M} + 3\mathbf{S} + 2\mathbf{D} + 17\mathbf{a}$, UADD-Qe-5, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_1 \leftarrow t_1 - T_3$, $t_1 \leftarrow t_1 - Z_3$, $T_3 \leftarrow dT_3$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$,

$X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $T_3 \leftarrow T_3 - X_3$, $T_3 \leftarrow T_3 - Y_3$, $t_1 \leftarrow t_1 d$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow 2t_1$, $Y_3 \leftarrow Y_3 - X_3$, $Y_3 \leftarrow Y_3 \cdot t_2$, $Y_3 \leftarrow Y_3 + t_1$, $X_3 \leftarrow Z_3 + T_3$, $X_3 \leftarrow X_3^2$, $T_3 \leftarrow T_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3 - Z_3$, $X_3 \leftarrow X_3/2$.

$9\mathbf{M} + 2\mathbf{S} + 2\mathbf{D} + 13\mathbf{a}$, `UADD-Qe-6`, assumes $a = -1/2$, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_1 \leftarrow t_1 - T_3$, $t_1 \leftarrow t_1 - Z_3$, $T_3 \leftarrow dT_3$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $T_3 \leftarrow T_3 - X_3$, $T_3 \leftarrow T_3 - Y_3$, $t_1 \leftarrow dt_1$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow 2t_1$, $Y_3 \leftarrow Y_3 - X_3$, $Y_3 \leftarrow Y_3 \cdot t_2$, $Y_3 \leftarrow Y_3 + t_1$, $X_3 \leftarrow Z_3 \cdot T_3$, $T_3 \leftarrow T_3^2$, $Z_3 \leftarrow Z_3^2$.

$8\mathbf{M} + 3\mathbf{S} + 3\mathbf{D} + 17\mathbf{a}$, `UADD-Qe-7`, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_1 \leftarrow t_1 - T_3$, $t_1 \leftarrow t_1 - Z_3$, $T_3 \leftarrow dT_3$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $T_3 \leftarrow T_3 - X_3$, $T_3 \leftarrow T_3 - Y_3$, $t_1 \leftarrow dt_1$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow 2t_1$, $X_3 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - X_3$, $Y_3 \leftarrow Y_3 \cdot t_2$, $Y_3 \leftarrow Y_3 + t_1$, $X_3 \leftarrow Z_3 + T_3$, $X_3 \leftarrow X_3^2$, $T_3 \leftarrow T_3^2$, $Z_3 \leftarrow Z_3^2$, $X_3 \leftarrow X_3 - T_3$, $X_3 \leftarrow X_3 - Z_3$, $X_3 \leftarrow X_3/2$.

$9\mathbf{M} + 2\mathbf{S} + 3\mathbf{D} + 13\mathbf{a}$, `UADD-Qe-8`, $(X_3 : Y_3 : T_3 : Z_3) = (X_1 : Y_1 : T_1 : Z_1) + (X_2 : Y_2 : T_2 : Z_2)$; $t_1 \leftarrow T_1 + Z_1$, $t_2 \leftarrow T_2 + Z_2$, $T_3 \leftarrow T_1 \cdot T_2$, $Z_3 \leftarrow Z_1 \cdot Z_2$, $t_1 \leftarrow t_1 \cdot t_2$, $t_1 \leftarrow t_1 - T_3$, $t_1 \leftarrow t_1 - Z_3$, $T_3 \leftarrow dT_3$, $t_2 \leftarrow Z_3 + T_3$, $Z_3 \leftarrow Z_3 - T_3$, $T_3 \leftarrow X_1 + Y_1$, $t_3 \leftarrow X_2 + Y_2$, $X_3 \leftarrow X_1 \cdot X_2$, $Y_3 \leftarrow Y_1 \cdot Y_2$, $T_3 \leftarrow T_3 \cdot t_3$, $T_3 \leftarrow T_3 - X_3$, $T_3 \leftarrow T_3 - Y_3$, $t_1 \leftarrow dt_1$, $t_1 \leftarrow t_1 \cdot X_3$, $t_1 \leftarrow 2t_1$, $X_3 \leftarrow kX_3$, $Y_3 \leftarrow Y_3 - X_3$, $Y_3 \leftarrow Y_3 \cdot t_2$, $Y_3 \leftarrow Y_3 + t_1$, $X_3 \leftarrow Z_3 \cdot T_3$, $T_3 \leftarrow T_3^2$, $Z_3 \leftarrow Z_3^2$.

Author Index