LNCS 7483

Dieter Gollmann
Felix C. Freiling (Ed

Informat

15th International Confere
Passau, Germany, Septemb
Proceedings

Dieter Gollmann   Felix C. Freiling (Eds.)

Information Security

15th International Conference, ISC 2012
Passau, Germany, September 19-21, 2012
Proceedings

Springer

Volume Editors

Dieter Gollmann
Hamburg University of Technology, Institute for Security in Distributed Applications
Harburger Schlossstrasse 20, 21073, Hamburg, Germany
E-mail: diego@tu-harburg.de

Felix C. Freiling
Friedrich-Alexander-University, Department of Computer Science
Martenstrasse 3, 91058 Erlangen, Germany
E-mail: felix.freiling@cs.fau.de

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-33382-8 e-ISBN 978-3-642-33383-5
DOI 10.1007/978-3-642-33383-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012946460

CR Subject Classification (1998): E.3, E.4, D.4.6, K.6.5, C.2, J.1, K.4.4

LNCS Sublibrary: SL 4 – Security and Cryptology

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

Printed on acid-free paper

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

Foreword

Information security continues to be a topic of great relevance to society. The Information Security Conference (ISC) is an annual international conference dedicated to research on the theory and applications of information security. The 2012 conference was the 15th in the conference series that started as a workshop in 1997, changed to a conference in 2001, and has been held on five different continents since. Hosted by the University of Passau in Lower Bavaria, ISC 2012 was the first conference in this series to be held in Germany.

ISC 2012 attracted high-quality papers on all technical aspects of information security. We received 72 submissions, which were reviewed by at least three members of the program committee. We finally accepted 23 papers, which are collected in these proceedings. In addition, we invited Isabel Trancoso and her colleagues to speak on privacy issues in speech processing. A second keynote speech was given by Michael Waidner from Fraunhofer SIT and Technische Universität Darmstadt.

We wish to thank all the people who invested time and energy to make ISC 2012 a success: First and foremost come all the authors who submitted papers to ISC and presented them at the conference. The members of the program committee together with all the external reviewers worked hard in evaluating the submissions and, in some cases, in shepherding promising work. The ISC steering committee, in particular Masahiro Mambo, helped us graciously in all critical decisions. Thanks also go to the 2012 general chair Joachim Posegga and his team at Passau University for handling the local arrangements, to Eric Rothstein for maintaining the conference website, and to Isaac Agudo and Cheng-Kang Chu for their efforts as publicity chairs.

July 2012

Dieter Gollmann
Felix C. Freiling

Organization

General Chair

Joachim Posegga	University of Passau, Germany

Steering Committee Chair

Masahiro Mambo	Kanazawa University, Japan

Program Chairs

Dieter Gollmann	Hamburg University of Technology, Germany
Felix C. Freiling	Friedrich-Alexander-Universität, Erlangen, Germany

Program Committee

Magnus Almgren	Chalmers University of Technology, Sweden
Tuomas Aura	Aalto University, Finland
Joonsang Baek	KUSTAR, UAE
Alex Biryukov	University of Luxembourg, Luxembourg
Sonja Buchegger	KTH Royal Institute of Technology, Sweden
Liqun Chen	HP Laboratories Bristol, UK
Xiaofeng Chen	Xidian University, P.R. China
Chen-Mou Cheng	National Taiwan University, Taiwan
Sherman S.M. Chow	University of Waterloo, Canada
Jorge Cuellar	Siemens, Germany
Vanesa Daza	Universitat Pompeu Fabra, Spain
Claudia Diaz	KU Leuven, Belgium
Roberto Di Pietro	Università degli Studi Roma Tre, Italy
Josep Domingo-Ferrer	Universitat Rovira i Virgili, Spain
Bao Feng	Institute for Infocomm Research, Singapore
Eduardo B. Fernandez	Florida Atlantic University, USA
Josep Ferrer	University of the Balearic Islands, Spain
Sara Foresti	Università degli Studi di Milano, Italy
Stefanos Gritzalis	University of the Aegean, Greece
Thorsten Holz	Ruhr-Universität Bochum, Germany
Martin Johns	SAP, Germany
Angelos Keromytis	Columbia University, USA
Igor Kotenko	Russian Academy of Sciences, Russia
Xuejia Lai	Shanghai Jiao Tong University, P.R. China

Zhenkai Liang	National University of Singapore, Singapore
Peng Liu	Pennsylvania State University, USA
Javier López	Universidad de Málaga, Spain
Masahiro Mambo	Kanazawa University, Japan
Mark Manulis	TU Darmstadt, Germany
Atsuko Miyaji	JAIST, Japan
Jose Morales	Carnegie Mellon University, USA
Raphael C.-W. Phan	Loughborough University, UK
Frank Piessens	KU Leuven, Belgium
Christian W. Probst	Technical University of Denmark, Denmark
Vincent Rijmen	KU Leuven, Belgium
Matt Robshaw	Orange Labs, France
Kouichi Sakurai	Kyushu University, Japan
Pierangela Samarati	Università degli Studi di Milano, Italy
Jörg Schwenk	Ruhr-Universität Bochum, Germany
Jan Seedorf	NEC Laboratories Europe, Germany
Miguel Soriano	Universitat Politècnica de Catalunya, Spain
Rainer Steinwandt	Florida Atlantic University, USA
Willy Susilo	University of Wollongong, Australia
Tsuyoshi Takagi	Kyushu University, Japan
Lingyu Wang	Concordia University, Canada
Susanne Wetzel	Stevens Institute of Technology, USA
Duncan Wong	City University of Hong Kong, P.R. China
Jeff Yan	Newcastle University, UK
S.M. Yiu	University of Hong Kong, P.R. China
Jianying Zhou	Institute for Infocomm Research, Singapore
Alf Zugenmaier	Hochschule München, Germany

Additional Reviewers

Man Ho Au
Werner Backes
Oleksandr Bodriagov
Andrey Chechulin
Kai Yuen Cheong
Tat Wing Chim
Cheng-Kang Chu
Willem De Groef
Philippe De Ryck
Xinshu Dong
Ming Duan
Sebastian Gajek
Wei Gao
Stefano Guarino
Seda Gurses
Javier Herranz
Jialin Huang
Kangkook Jee
Christos Kalloniatis
Ergina Kavallieratou
Gunnar Kreitz
Po-Chun Kuo
Sebastian Lekies
Gaëtan Leurent
Wei Li
Yue-Hsun Lin
Flavio Lombardi
Nikolaos Makriyannis
Daniel A. Mayer
Kirill Morozov

Nick Nikiforakis
Evgenia Novikova
Kazumasa Omote
Kailas Patil
Constantinos Patsakis
Alfredo Rial
Panagiotis Rizomiliotis
Santi Martinez Rodriguez
Rodrigo Roman
Arnab Roy
Igor Saenko
Bagus Santoso
Theodoor Scholte
Nicolas Sendrier
Isamu Teranishi
Elmar Tischhauser
Aggeliki Tsochou
Frederik Vercauteren
José Luis Vivas
Huaqun Wang
Baodian Wei
Ralf-Philipp Weinmann
Jia Xu
Kenji Yasunaga
Eunjung Yoon
Bin Zhao
Hang Zhao
Yuanjie Zhao
Hui Zhang
Tao Zhang
Yinghui Zhang
Chen Zhong
Youwen Zhu

Publicity Chairs

Isaac Agudo	Universidad de Málaga, Spain
Cheng-Kang Chu	National Chiao Tung University, Taiwan

Local Organization

Marita Güngerich, Siglinde Böck, Eric Rothstein

Table of Contents

Privacy-Preserving Speaker Authentication

Manas Pathak[1], Jose Portelo[2], Bhiksha Raj[1], and Isabel Trancoso[2]

[1] Carnegie Mellon University, Pittsburgh, PA, USA
[2] INESC-ID/IST, Lisbon, Portugal

Abstract. Speaker authentication systems require access to the voice of the user. A person's voice carries information about their gender, nationality etc., all of which become accessible to the system, which could abuse this knowledge. The system also stores users' voice prints – these may be stolen and used to impersonate the users elsewhere. It is therefore important to develop *privacy preserving* voice authentication techniques that enable a system to authenticate users by their voice, while simultaneously obscuring the user's voice and voice patterns from the system. Prior work in this area has employed expensive cryptographic tools, or has cast authentication as a problem of exact match with compromised accuracy. In this paper we present a new technique that employs secure binary embeddings of feature vectors, to perform voice authentication in a privacy preserving manner with minimal computational overhead and little loss of classification accuracy.

1 Introduction

The number of services provided to users over the network grows daily. Many of these services require users to authenticate themselves, typically through passwords. Increasingly, these services are accessed through small, voice-enabled client devices such as smartphones and cellphones. In these cases, authenticating a user by voice is often more convenient than typing in passwords or passphrases. Even in cases where passwords may be typed conveniently, they may be combined with voice-based authentication as an added form of security. As a result, voice-based authentication systems, also often referred to as *speaker authentication systems* or *speaker verification systems* are becoming increasingly popular.

In a speaker authentication system, a user enrolls with the system by providing some "enrollment" recordings of themselves from which the system may form a voice print or model for the user. Subsequently, during operation the user announces their identity and provides an "authentication" recording to the system. The system compares this recording to the stored model for the announced identity and determines if the the user is indeed who they claim to be.

However, in the process of using speaker authentication systems, the user currently places a lot of faith in the system. A person's voice, in addition to being a biometric, carries information regarding the person's gender, their emotional state, their nationality, etc. The implicit trust is that the system will not extract this information from the signal and use it for other undesired purposes, such as to generate advertisements aimed at the user, or to sell it to third parties.

There are other risks as well. In the discussion that follows, we will not distinguish between the system and a malicious hacker who may have compromised the system and has access to all data and information it possesses. The system may manipulate and edit the recordings provided by the speaker to create fake recordings that the user never spoke. It could also use the recordings to break into other systems where the user has authenticated themselves by voice.

In addition to the privacy and security violations listed above which result from unintended use of the speaker's data, there is scope for direct privacy violation through usage of the authentication system in *intended* ways. The system could use the model it possesses for a user to uncover other recordings by the user, such as on services like YouTube, or even from recordings of audio in public places, where the user may have assumed anonymity.

Clearly then, in order to ensure the user's privacy the following conditions must ideally be satisfied: First, the system should not have clear access to the audio recorded by the user. Second, the system should also not possess a model of the user's speech that it could use to identify the speaker elsewhere. These are not entirely unreasonable requirements; similar criteria are in fact stated as being desirable for other forms of secure biometrics as well [1].

In this paper we describe two recently-developed *privacy-preserving* frameworks for speaker authentication that address these problems by enabling a user to perform authentication without revealing their actual voice patterns to the system, either during enrollment or during authentication phases.

The first framework [2] employs *secure function evaluation* (SFE) protocols [13] to implement a conventional state-of-art voice authentication algorithm [14] in a manner that enables the user and system to interact only on encrypted data which the system cannot decrypt and observe in plaintext. Both the audio transmitted by the user to the system and the models stored by it are encrypted in this manner. Thus the system (or an adversary who breaks into it) can neither manipulate the incoming audio for nefarious purposes, nor make any undesired inferences from the audio or the models it possesses.

The second approach [3] takes a different tack – it modifies the basic pattern matching algorithms employed for speaker authentication to enable them to be performed through exact and inexact match of hash strings derived from the audio obtained through stochastic locality-sensitive-hashing functions. Here too, the data transmitted by the user to the system, including both enrollment and authentication data, are transformed in a manner that does not permit the system to make undesired inferences from it, thereby protecting the user.

The SFE-based approach provides the same accuracy as the state-of-the-art in voice authentication systems, since it essentially only embeds the same computation into a privacy-preserving framework. It also generalizes to other problems in speech processing, such as the training of Gaussian mixture densities or HMMs from private speech data [15] and inference over HMMs from private data [16], both of which are component problems in many speech applications. On the other hand, it requires repeated encryption and decryption of the data, imposing high computational overhead on the process. Moreover, it remains vulnerable to imposters who may take over the user's client device, and while these can also be protected against, such protection would impose additional computational expense. The hashing-based methods, on the other hand, are relatively lightweight and fast; however by modifying the fundamental classification algorithms themselves they may compromise classification accuracy. Also, they are specific to the problem at hand – speaker authentication, and do not generalize beyond it in their current format.

In all cases we envision a client-server model, where a user employs a computation-capable client device such as a smartphone or computer, which can perform the operations necessary to protect the data. For completeness, we must also consider the possibility of an imposter who gains access to the user's client device and can manipulate the data its transmits. The proposed frameworks must ideally ensure that such an imposter cannot break into the system and pretend to be the user. We do *not* address the issue of fraudulent voice input, such as by imposters who may attempt to mimic the user's voice or criminals who may force the user to speak; these do not fall under the scope of the presented work. We will also generally assume that all communication between the user and the system is over an appropriately secured channel and hence impervious to man-in-the-middle or replay attacks by eavesdroppers.

In closing this introductory section, we note that while there has been a substantial body of work on general techniques for data processing with privacy constraints, concerns about the privacy of voice-based biometric systems, and speech applications in general have largely been ignored until recent times [17], and literature addressing the topic is sparse. Smaragdis and Shashanka [15] propose SFE protocols for training and evaluating Gaussian mixtures and hidden Markov models on speech data, with privacy constraints. Pathak, et al. [16] develop and implement an efficient protocol for privacy preserving HMM inference applied to isolated word recognition. The work reported in this article and the related citations are the only extant publications that address the issue of privacy-preserving speech biometrics, to the best of our knowledge.

In the following sections we first describe the problem of speaker authentication and common state-of-art methods for the task (Section 2). Subsequently, we describe the SFE-based procedure in Section 3 and the hashing-based approach in Section 4, followed by conclusions and discussion in Section 5.

2 Speaker Authentication

We begin by briefly outlining the actual algorithms used for speaker authentication, as a prelude to describing our privacy-preserving frameworks. Recall that a user enrolls with a speaker authentication system by providing it enrollment voice samples. The system builds a "model" for the user from these samples. Later, to authenticate himself the user announces his identity to the system and provides it with a voice sample. The system compares the sample to the model it has for the user to authenticate him. In the outline we will discuss both how the model is learned, and how it is used for authentication.

Before we proceed, it must be noted that the system does not actually work on the digitized speech signal directly. Instead, speech signals are parameterized into a sequence of *feature* vectors, typically *mel-frequency cepstral coefficients* (MFCC) vectors [6] augmented by their temporal derivatives and double derivatives, which characterize the variation in the frequency content of the signal as a function of time. Thus, when we refer to the audio signal, we are in fact only referring to the sequence of feature vectors derived from it.

Note that our discussion assumes *text-independent* speaker authentication, where the user is not required to say a specific passphrase. However the techniques are easily extended to text-dependent authentication also.

2.1 Speaker Authentication Using Likelihood Ratios from GMMs

The most common and successful technique for text-independent speaker authentication treats the problem as one of hypothesis testing [18], performed using a likelihood ratio test. To authenticate a recording $\mathbf{X}$ given by a speaker, the system computes the probability of $\mathbf{X}$ using a model λ_s for the speaker and compares it to the probability computed from a *Universal Background Model* (UBM) λ_U representing generic speech. Authentication uses the following rule:

$$\frac{P(\mathbf{X}|\lambda_s)}{P(\mathbf{X}|\lambda_U)} \begin{cases} \geq \theta & \text{accept speaker,} \\ < \theta & \text{reject speaker.} \end{cases} \tag{1}$$

Each recording $\mathbf{X}$ actually comprises a sequence of feature vectors, *i.e.* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T]$. The individual vectors $\mathbf{x}_t$ are assumed to be independent and identically distributed. The probability distribution of the individual vectors is assumed to be a Gaussian mixture [19] with the form

$$P(\mathbf{x}_t|\lambda_C) = \sum_k w_k^C \mathcal{N}(\mathbf{x}_t; \mu_k^C, \Sigma_k^C) \tag{2}$$

where C is either s or U, w_k^C represents the mixture weight of the k^{th} Gaussian in the Gaussian mixture density for C, $\mathcal{N}()$ represents a Gaussian, and μ_k^C and Σ_k^C represent the mean and variance of the k^{th} Gaussian for C.

To effectively represent the characteristics of a generic speaker, the parameters of the UBM λ_U are usually learned from a collection of speech recordings from a large number of speakers, typically using the expectation-maximization (EM) algorithm. The parameters λ_s of the model for a speaker s are obtained by adapting the UBM to the enrollmentp data from the speaker as follows.

Model Adaptation
The UBM parameters are adapted to individual speakers using *maximum a posteriori* (MAP) estimation. It has been empirically established that speaker models obtained by MAP estimation signicantly outperform the models trained directly on the enrollment data [20][18].

The MAP estimation procedure comprises estimation of a sample estimate of the parameters of the distribution for the speaker, followed by interpolation with the UBM. Given set of feature vectors $\mathbf{x}_1, \cdots, \mathbf{x}_T$ obtained from enrollment samples, we first compute the *a posteriori* probabilities of the individual Gaussians in the UBM. For the i^{th} mixture component of the UBM, the *a posteriori* probability conditioned on $\mathbf{x}_t$ is given by

$$P(i|\mathbf{x}_t) = \frac{w_i^U \mathcal{N}(\mathbf{x}_t; \mu_i^U, \Sigma_i^U)}{\sum_k w_k^U \mathcal{N}(\mathbf{x}_t; \mu_k^U, \Sigma_k^U)}. \tag{3}$$

Similarly to the M-step of EM, the *a posterior* probabilities are then used to compute the updated sample estimates of mixture weights, means, and variances.

$$w_i' = \frac{1}{T}\sum_t P(i|\mathbf{x}_t), \qquad \mu_i' = \frac{\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t}{\sum_t P(i|\mathbf{x}_t)},$$
$$\Sigma_i' = \frac{\sum_t P(i|\mathbf{x}_t)\mathrm{diag}(\mathbf{x}_t\mathbf{x}_t^T)}{\sum_t P(i|\mathbf{x}_t)}. \tag{4}$$

Finally, the parameters of the adapted model $\lambda_s = \{w_i^s, \mu_i^s, \Sigma_i^s\}$ are given by the convex combination of these new parameters and the UBM parameters:

$$w_i^s = \alpha_i w_i' + (1-\alpha_i) w_i^U, \qquad \mu_i^s = \alpha_i \mu_i' + (1-\alpha_i)\mu_i^U,$$
$$\Sigma_i^s = \alpha_i \Sigma_i' + (1-\alpha_i)\left[\Sigma_i^U + \mu_i^U {\mu_i^U}^T\right] - \mu_i^s \mu_i^{sT}. \tag{5}$$

The adaptation coefficients α_i for the mixture components control the contribution of the enrollment data to the estimate, relative to the UBM.

A number of variations on the above scheme have also been proposed, primarily aimed at accounting for the fact the amount of “registration” data may be limited, and that the recording conditions for the registration and authentication recordings may differ. These variants typically employ various flavors of factor analysis [8][9] to assign *a priori* probabilities to the parameters of λ_s. However, the fundamental structure remains what has been described above, and we shall remain with it for the presentation in this article.

2.2 Speaker Authentication Using GMM Supervectors

An alternate equally-successful approach to likelihood ratio tests obtains a separate Gaussian mixture for each of multiple enrollment recordings from the speaker through MAP adaptation of the UBM. Thus, instead of having a single Gaussian mixture repesenting the speaker, we now have multiple Gaussian mixtures, each representing one of the enrollment recordings. For each recording, the parameters of Gaussian mixture, namely the means, covariances (which are usually assumed to be diagonal matrices) and mixture weights, are concatenated into a “supervector” representing the recording. Similar supervectors are obtained for a number of recordings from each of a large number putative imposters. The collection of supervectors from the target speaker and the imposters are used as positive and negative exemplars to train a binary classifier.

For authentication, a supervector is similarly obtained from the test recording and classified by the binary classifier.

Note that once again *a priori* probability distributions may be assigned to the parameters of the Gaussian mixtures through factor analysis as in [21]. In this case the vectors of factors may be used instead of the actual supervectors to represent the recordings.

2.3 A SVM Based Authentication System

The most common binary classifiers used in this framework are kernel-based support vector machines [10]. A number of differnt kernels may be used; however we consider only RBF kernels. Here the kernel is given by $k(\mathbf{v}_i, \mathbf{v}_j) = e^{-\gamma \cdot d^2(\mathbf{v}_i, \mathbf{v}_j)}$, where $d(\mathbf{v}_i, \mathbf{v}_j)$ refers to the Euclidean distance between vectors $\mathbf{v}_i$ and $\mathbf{v}_j$, and γ is a scaling factor. RBF kernels are observed to result in good classification accuracies for the speaker authetication task. Moreover, they are easily adapted to the hashing-based privacy-preserving speaker authentication schemes we will describe later.

2.4 *k*-Nearest-Neighbor Based Classification

As an alternative, we also consider a k-NN based classifier. k-NN classifiers are not normally employed in speaker authentication systems; however they are easily adapted to the hashing-based privacy-preserving authentication system we will describe later. Here, for each speaker S, the system retains a set of "exemplar" vectors $\mathcal{V}_S$ derived from registration recordings by the speaker. In addition, for each of a large number of *imposters* I from an imposter set $\mathbb{I}$, it retains a set of exemplar vectors $\mathcal{V}_I$. The procedure for classification is given by Algorithm 1.

Note that although we have referred to the algorithm as k-NN, we do not restrict ourselves to the k closest neighbors, but consider *all* exemplars. Also, the weights assigned to exemplars are based on mean reciprocal rank (MRR), rather than the distance to $\mathbf{y}$.

3 Privacy-Preserving Authentication as Secure Function Evaluation

Secure Function Evaluation is a formalism under which two parties with private inputs can jointly compute a function of both their inputs without exposing them to one another [13]. Under the SFE framework we treat speaker authentication as a problem where the system and user must compute the likelihood-ratio test described in Section 2.1 from their private inputs, namely the users audio and the models retained by the system, without exposing them to one another.

The general principle behind the SFE framework employed is illustrated in Figures 1 and 2. The key is to decompose the overall operation into a sequence

Algorithm 1. MRR algorithm for authentication

```
Training:
• Store the exemplar set of vectors $\mathcal{V}_S$ from the training instances for the target speaker $S$.
• For each imposter $I$ in imposter set $\mathbb{I}$, store a set of exemplar vectors $\mathcal{V}_I$. Let $\mathcal{V} = \mathcal{V}_S \cup_{I \in \mathbb{I}} \mathcal{V}_I$ be the set of all exemplars for all speakers, including target and imposters.
Testing:
• Given "test" vector $\mathbf{y}$ from claimed speaker $S$
• Rank order all exemplars $\mathbf{v} \in \mathcal{V}$ as follows:
    $$rank(\mathbf{v}) = |\mathcal{V}| - \sum_{\mathbf{u} \in \mathcal{V} \setminus \mathbf{v}} I(d(\mathbf{u}, \mathbf{y}) > d(\mathbf{v}, \mathbf{y}))$$
  where $I()$ is an indicator function.
for each speaker $C \in S \bigcup \mathbb{I}$ do
    $Score(C) = \sum_{\mathbf{v} \in \mathcal{V}_C} \frac{1}{rank(\mathbf{v})}$
end for
If $Score(S) > \max_{C \in \mathbb{I}} Score(C)$ accept $S$ else reject $S$
```

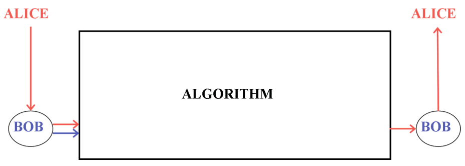


Fig. 1. In a regular computation, user "Alice" sends her data a to system "Bob", who combines it with his data b to compute the function. The output c is returned to Alice.

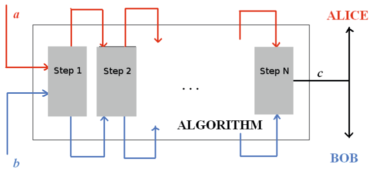


Fig. 2. In an SFE, the computation of the function is decomposed into several intermediate steps. Each step requires inputs from both parties. The output of each step is distributed to both parties as random additive shares.

of intermediate steps. Each step requires inputs from both the user and the system, and the output of each step is also distributed across both as random additive shares that individually do not reveal anything about the outcome of the step. The final outcome is received by the system. The manner in which the computation is broken down and performed in this manner is specified through detailed protocols.

We now briefly present a preliminary discussion of some of the tools we employ that enable us to decompose the computation in this manner, and subsequently outline the protocols we employ for various aspects of speaker verification, followed by the performance, security and computation caveats to consider.

3.1 Preliminaries

Homomorphic Encryption. The entire framework is predicated on the use of *homomorphic encryption* techniques that allow for arithmetic and logical operations to be performed directly on encrypted data to obtain encrypted results. If $\cdot$ and $+$ are two operators and x and y are two plaintexts, a homomorphic encryption function E satisfies $E[x] \cdot E[y] = E[x + y]$. Thus, a party who only possesses the encrypted values $E[x]$ and $E[y]$ can obtain $E[x+y]$, the encryption of the outcome of the operation $x + y$, without actually ever knowing the true values of x and y.

In this work, we use the Paillier cryptosystem [22] which is a *public-key* cryptosystem that satisfies additive homomorphism. Given two numbers x and y, the Paillier cryptosystem satisfies $E[x] \cdot E[y] = E[x + y]$, and as a consequence, $E[x]^y = E[x \cdot y]$. We refer the reader to [22] for more details on the crypto system.

Using the homomorphic encryption scheme, we can now also define a set of "primitive" protocols that enable the user and system to perform some operations. We will present these primitives as they are required by our operations. Given space constraints, we will not explain exactly how the primitives are implemented, and refer the reader to [2] [15] for details.

Gaussian Computation as a Dot Product. We make extensive use of the fact that the computation of the log of a Gaussian can be expressed as a dot product. We use the following construction from [15]. The logarithm of a multivariate Gaussian density $\mathcal{N}(x; \mu, \Sigma)$ computed on a d-dimensional vector $\mathbf{x}$ can be represented as the quadratic product $\log \mathcal{N}(\mathbf{x}; \mu, \Sigma) = \tilde{x}^T \tilde{W} \tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}$ is an extended vector obtained by concatenating 1 to $\mathbf{x}$, and $\tilde{W}$ is a $(d+1) \times (d+1)$ matrix given by.

$$\tilde{W} = \left[\begin{array}{c:c} -\frac{1}{2}\Sigma^{-1} & \Sigma^{-1}\mu \\ \hdashline 0 & w^* \end{array}\right], \tag{6}$$

where $w^* = -\frac{1}{2}\mu^T \Sigma^{-1} \mu - \frac{1}{2} \log |\Sigma|$.

We can reduce this computation further to a single inner product $\bar{\mathbf{x}}^T W$, where $\bar{\mathbf{x}}$ is a *quadratically extended* feature vector derived from $\tilde{\mathbf{x}}$ which consists of all pairwise product terms $\tilde{x}_i \tilde{x}_j$ of all components $\tilde{x}_i \in \tilde{\mathbf{x}}$, and W is obtained by unrolling $\tilde{W}$ into a vector. In this representation $\log \mathcal{N}(x; \mu, \Sigma) = \bar{\mathbf{x}}^T W$.

We can now describe the SFE protocols for the following operations: user enrollment including model adaptation to the user's data, and authentication.

In all of the description below, we assume only the user has the private key to the homomorphic encryption scheme which can be used to decrypt encrypted data, but both the user and the system possess the public encryption key and can encrypt data. Thus the user can perform both encryption $E[\cdot]$ and decryption $E^{-1}[\cdot]$, while the system can only preform $E[\cdot]$.

We assume that the user computes MFCC feature vectors from the audio on his client device. We also assume that the user has computed both the feature vectors $\mathbf{x}_t$ and their quadratic extensions $\bar{\mathbf{x}}_t$. All communication of data from the user to the system are encrypted with his private key. All references to Gaussians actually refer to the unrolled vector W derived from the $\tilde{W}$ matrices for the Gaussians.

3.2 Private Enrollment Protocol

We assume that the system has access to the UBM, λ_U trained on a collection of speech data. We propose the following protocol to adapt the system's UBM to the user's enrollment samples. We refer to individual Gaussian components in a mixture as $P(\mathbf{x}|i)$ for conciseness.
Inputs:

1. User has feature vectors $\mathbf{x}_1, \ldots, \mathbf{x}_T$ and their quadratic extensions $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_T$ derived from enrollment samples.
2. System has the UBM $\lambda_U = W_i^U$ for $i = 1, \ldots, N$, and mixing weight α.

Output: System obtains encrypted adapted model $E[\lambda_s] = \{E[w_i^s], E[\mu_i^s], E[\Sigma_i^s]\}$.

Computing the posterior probabilities:
For each t:

1. The user sends the encrypted vector $E[\bar{x}_t]$ to the system.
2. The system computes the encrypted log probabilities for each Gaussian in the UBM: $E\left[\log w_i^U P(\mathbf{x}_t|i)\right] = \sum_j E[\bar{\mathbf{x}}_{t,j}]^{W_{i,j}^U} + E[\log w_i^U]$.
3. The *logsum* protocol [15] enables a party holding $E[\log x]$ and $E[\log y]$ to collaborate with another party who holds the private encryption key to obtain $E[\log(x+y)]$ without revealing x or y. The system participates with the user's client in the logsum protocol to obtain $E\left[\log \sum_i w_i^U P(x_t|i)\right]$.
4. The system computes encrypted log posteriors: $E[P(i|\mathbf{x}_t)] = E\left[\log w_i^U P(\mathbf{x}_t|i)\right]$ - $E\left[\log \sum_i w_i^U P(\mathbf{x}_t|i)\right]$.
5. The *private exponentiation protocol* [2] enables a party holding $E[\log x]$ to collaborate with the party who holds the encryption key to obtain $E[x]$ without revealing x. The system then executes the private exponentiation protocol with the user to obtain $E\left[P(i|\mathbf{x}_t)\right]$.

Learning w_i^s.

6. The system $E[w_i']$ as $E\left[\sum_t P(i|\mathbf{x}_t)\right] = \prod_t E\left[P(i|\mathbf{x}_t)\right]$.
7. The system then computes the encrypted updated mixture weights as

$$E[w_i^s] = E[w_i']^{\alpha/T} E[w_i^U]^{1-\alpha}.$$

Learning μ_i^s.

8. The system generates subtracts a random number r homomorphically from $E[P(i|\mathbf{x}_t)]$ to obtain $E[P(i|\mathbf{x}_t) - r]$ and sends it to the user.
9. The user decrypts this value and multiplies it by feature vectors to obtain $P(i|\mathbf{x}_t)x_t - r\mathbf{x}_t$. He encyrpts it and sends $E[P(i|\mathbf{x}_t)x_t - r\mathbf{x}_t]$ and $E[\mathbf{x}_t]$ to the system.
10. The system computes $E[P(i|\mathbf{x}_t)\mathbf{x}_t] = E[P(i|\mathbf{x}_t)\mathbf{x}_t] = E[\mathbf{x}_t]^r + E[P(i|\mathbf{x}_t)\mathbf{x}_t - r\mathbf{x}_t]$. It then computes $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t] = \prod_t E[P(i|\mathbf{x}_t)\mathbf{x}_t]$.
11. The *private division protocol* [2] enables a party holding $E[x]$ and $E[y]$ to collaborate with the party holding the encryption key to obtain $E[x/y]$ without revealing x or y.The system engages the user in a private division protocol with $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t]$ and $E[w']$ as inputs to obtain $E[\mu']$.
12. The system then computes the encrypted adapted mean as.

$$E[\hat{\mu}_i^s] = E[\mu_i']^\alpha E[\mu_i^U]^{1-\alpha}.$$

Learning $\hat{\Sigma}_i^s$.

This is similar to learning $\hat{\mu}_i^s$ and continues from Step 9 of the protocol.

13. The user multiplies $P(i|\mathbf{x}_t) - r$ by $\mathbf{x}_t\mathbf{x}_t^T$ to obtain[1]. $P(i|\mathbf{x}_t)x_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T$. He sends $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T]$ and $E[\mathbf{x}_t\mathbf{x}_t^T]$ to the system.
14. The system, which knows r, computes $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T - r\mathbf{x}_t\mathbf{x}_t^T]E[\mathbf{x}_t\mathbf{x}_t^T]^r$ to obtain $E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ for each t. The system then computes $\prod_t E[P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ to obtain $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$.
15. The system engages the user in the private division protocol with $E[w']$ and $E[\sum_t P(i|\mathbf{x}_t)\mathbf{x}_t\mathbf{x}_t^T]$ as inputs to obtain $E[\Sigma_i']$.
16. The *private vector product protocol* [2] permits a user with encrypted vectors $E[\mathbf{x}]$ and $E[\mathbf{y}]$ to engage with a user who has the private decryption key to compute $E[\mathbf{x} \odot \mathbf{y}]$, where $\odot$ represents a Schur (component-wise) product. The system and user participate in the private vector product protocol with input $E[\mu_i^s]$ to obtain $E[\mu_i^s \mu_i^{sT}]$. The system then computes the encrypted updated covariance as

$$E[\Sigma_i^s] = E[\Sigma_i']^\alpha E[\Sigma_i^U + \mu_i^U \mu_i^{UT}]^{1-\alpha} - E[\mu_i^s \mu_i^{sT}].$$

The above protocol only represents the first iteration of the iterative MAP adaptation algorithm. In practice many iterations are required. The primary variation in subsequent iterations is that the user now only possesses the encrypted model learned in the previous iteration, and Step 1. of the above protocol cannot be performed. Instead this must be replaced by the protocol followed for authentication, which computes encrypted Gaussian log likelihoods from encrypted models and data.

At each iteraion (and finally) the encrypted model parameters obtained by the system cannot be directly used, since it only obtains the encrypted values of

[1] For efficiency purposes, only the diagonal terms of the product $x_t x_t^T$ can be included without having a significant impact on accuracy.

the means, mixture weights and variances; the product and logarithmic terms in W_i^s must be computed from it. This can be done through a rather simple protocol in which the system adds (or multiplies) random terms to the individual parameters of the model homomorphically to mask their true value and ships the masked values to the user. The user decrypts the data, performs the necessary operations, re-encrypts the values and returns them to the system. The system thereafter proceeds to eliminate the random maskers from the encrypted result, also homomorphically. The details of the protocol can be found in [2].

3.3 Private Verification Protocol

Once enrolment is complete, the system possesses encrypted models for the user, as well as the UBM. These are used to authenticate the user as follows:

Inputs:

1. User has extended feature vectors $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_T$ for a test sample.
2. System has $E[\lambda_s] = E[W_i^s]$, for all Gaussians $i = 1, \ldots, N$.

Output: System obtains the score $E[\log P(x_1, \ldots, x_T|\lambda_s)]$.

1. The system generates a random vector p and computes $E[W_i^s] \cdot E[-p]$ to obtain $E[\hat{W}_i^s - p]$. This is sent to the user.
2. The user decrypts it and obtains $W_i^s - p$. He computes the inner product $\bar{\mathbf{x}}_t^T(W_i^s - p)\ \forall t$, and sends $E[\bar{\mathbf{x}}_t^T(W_i^s - p)]$ and $E[\bar{\mathbf{x}}_t]\ \forall t$ to the system.
3. The system computes the inner product $E[\bar{\mathbf{x}}_t^T p] = \prod_k E[\mathbf{x}_{t,k}]^{p_k}\ \forall t$. It then computes $E[\bar{\mathbf{x}}_t^T W_i^s] = E[\bar{\mathbf{x}}_t^T(W_i^s - p)] \cdot E[\bar{x}_t^T p]\ \forall i$. The system and the user then participate in the *logsum* protocol to obtain $E[\log P(\mathbf{x}_t|\lambda_s)]$. The system finally computes $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_s)] = \prod_t E[\log P(\mathbf{x}_t|\lambda_s)]$.

The above protocol can be used to compute scores for both the speaker model and the UBM (although the system has access to the UBM in plain text, it is actually more efficient to simply encrypt it and compute the encrypted UBM score in this manner).

Once both the encrypted "user" score $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_s)]$ and the UBM score $E[\log P(\mathbf{x}_1, \ldots, \mathbf{x}_T|\lambda_U)]$ are available, the system can engage a secure comparison protocol [23] to determine which is greater.

3.4 The Efficacy of the SFE Approach

Since the SFE protocols described above essentially enable a direct privacy-preserving implementation of the likelihood-ratio test, actual authentication performance is not significantly different from that obtained with the original "unsecured" method. The only degradation comes from the fact that the encryption works over finite integer fields, and to utilize it all numbers must be converted to finite-precision integers. However, even with only 16-bit precision the performance of the algorithm is largely comparable to that obtained with floating-point valued features and models.

The larger issue arises from the computational overhead. The computations require repeated exchange of information between the user and the system, as well as repeated encryption and decryption. Table 1 shows execution times for the verification phase of the protocol using different encryption schemes – enrollment times would be several factors higher. All Gaussian mixtures employed in this experiment were mixtures of 32Gaussians. The table only evaluates the computation time, most of which is dominated by the time required for encryption and decryption. Communication overhead is negligible compared to these times. For comparison, the conventional "insecure" version of the algorithm took 14 seconds to run on the same computer, on the same utterance. Clearly the feasibility of this approach in practical terms is questionable within current computational infrastructures.

Table 1. Execution time of the verification protocol for a single 4.4second utterance

Steps	**Time (256-bit)**	**Time (1024-bit)**
Encrypting $\bar{x}_t$ $\forall t$	576.81 s	35747.82 s
Evaluating Adapted	407.21 s	7597.57 s
Evaluating UBM	same as adapted	same as adapted
Comparison	0.3 s	16.88 s
Total $=E[\bar{\mathbf{x}}_t]$ + adapted + UBM + compare	1391.53 s $\sim$ 23 min	50959.84 s $\sim$ 14 hr, 9 min

A second factor to be considered is security. The SMC protocols described above ensure that if they are correctly followed, no information is leaked whatsoever. The system learns nothing about the user's data, and the user in turn does not learn anything about the models stored by the system. The original objectives of privacy we set forth are satisfied.

However, the framework described above assumes "honest but curious" users and system. The user and the system are assumed to correctly perform the operations required of them, although they may inspect the output. Such an assumption is clearly naive. For instance, an imposter who has broken into the user's client device could modify the numbers sent out by the user's client device in only the last step of the protocol, *i.e.* the secure comparison protocol; this would result in incorrect results being obtained by the system, which result in the imposter obtaining a much-higher-than-random probability of breaking into the system. These too can be protected against, by including a stage of verification after each operation, *e.g.* using zero-knowledge proofs [24]; however such verification can increase the communication and computation overhead manifold over the current values.

On the other hand, the SMC framework not only has the benefit of being exact; it can also be generalized to incorporate other models, and perform additional tasks such as actual recognition of speech. Moreover, with appropriate verification, the procedures can be made arbitrarily secure.

4 Privacy Preserving Speaker Authentication through Hashing

The computational overhead from the SFE framework arises from two reasons. First, homomorphic encryption is expensive, as are homomorphic operations on data. Secondly, the processing is *interactive*, requiring repeated exchange of large amounts of data between the user and system.

By comparison, text-password based systems are blindingly fast, and yet highly secure. The reasons for this are not difficult to identify. Firstly, authentication requires an exact match between the password stored by the system and the string typed in by the user. This means that fast one-way hashing functions (*e.g.* DES) may be employed to encrypt the password. Since the stored password and the password transmitted by the user encrypt to the same cryptostring, comparison can be performed in the encrypted domain: a perfect match results in authentication; a mismatch provides no information about the original (unencrypted) text strings. A consequence of this is that the exchange need not be interactive: the user transmits a password, and the system determines whether the user must be authenticated in a single comparison operation; in the entire process the user's password remains secure.

On the other hand, the conventional password-based authentication scheme cannot be applied to speaker authentication for an obvious reason: speech recordings are not deterministic, and features derived from enrollment and test recordings are highly unlikely to be identical. Moreover, speech recordings tend to vary in length, making such direct exact comparisons impossible in most cases.

In the Hashing-based framework we attempt to map the authentication problem to a process similar to password matching. To do so, we employ the supervector-based mechanisms for authentication described in Section 2.2. Recall that in this formalism each recording, both from the enrollment data and the test data, is converted to a single "supervector" by adapting a UBM to the recording, and concatenating the parameters of the resultant distribution into a single vector. The length of the supervector only depends on the number of Gaussians in the UBM and is independent of the length of the recording, thereby resulting in a fixed-length parameterization of the audio. Nevertheless the supervectors are not determinsitic and will vary from recording to recording, and cannot be matched to one another.

In [3] a technique was proposed to convert these supervectors to password-like strings by hashing them using a *locality sensitive hashing* (LSH) scheme [25]. Once hashed in this manner, the supervectors of different recordings from a speaker could map onto the same hash with high probability. The problem here was that recordings from imposters too could map onto the same hashes. To reduce the incidence of such collisions, enrollment recordings were hashed multiple times using several hashes, and the collection of hash values was used as a password set for the speaker. Test recordings were hashed in the same manner, and the set of test hashes were compared to the hashes from the enrollment set. If a sufficient number of exact matches were found between the two sets, the user was authenticated.

By converting the problem of authentication to a password matching problem, privacy issues were effectively eliminated. Computation of supervectors and hashes was performed on the user's client device. The server only obtained hashes that had been encrypted by a one-way function such as SHA-256, both from the enrollment data and during authentication. Speaker authentication in this manner was as secure as text-based password matching. The problem however, was one of accuracy – by requiring only exact matches between hashes of test and enrollment data, the accuracy of the system was compromised.

The hashing-based mechanism we employ approaches the problem of conversion of audio into hashes differently – we employ a hashing mechanism based on the principle of *secure binary embeddings* (SBE) [5][4]. SBEs have an interesting property – they convert the audio into hashes that can be compared to one another in a manner that permits a small degree of mismatch between the hashes. At the same time, they provide information theoretic guarantees that the distance between hashes of vectors provides no information about the true relationship between the vectors, if the vectors are sufficiently far apart. Furthermore, they cannot be inverted to recover supervectors from their hashes, if the key to the hashing function is kept private.

In the following sections we will describe the SBE and its properties, and how we apply it to the problem of speaker authentication. Since the actual mechanism employed to authenticate users is no longer identical to existing techniques, which are based on the ability to estimate the actual distances between vectors, we also include a section with experiments demonstrating the effectiveness of the approach.

4.1 Secure Binary Embeddings

Secure binary embeddings (SBE) [5] convert vectors to bit sequences through random projections followed by band quantization. The resulting bit strings, which we will refer to as "hashes", have the following property: if the Euclidean distance between two vectors is below a threshold, the Hamming distance between their hashes is proportional to the Euclidean distance; above the threshold the Hamming distance provides no information regarding the true distance between the two vectors. Given an L-dimensional vector $\mathbf{x} \in \mathbb{R}^L$, the M-bit SBE of $\mathbf{x}$ is defined as

$$\mathbf{q}(\mathbf{x}) = Q\left(\Delta^{-1}(\mathbf{A}\mathbf{x} + \mathbf{w})\right) \tag{7}$$

where Δ is a precision parameter, $\mathbf{A} \in \mathbb{R}^{M \times L}$ is a random matrix whose elements are drawn from a normal distribution with zero mean and unit variance, and $\mathbf{w} \in \mathbb{R}^M$ is a vector composed of random numbers drawn uniformly from $[0, \Delta]$. $Q()$ is a quantization function given by $Q(\mathbf{y}) = \lfloor \mathbf{y}\%2 \rfloor$, where the floor and modulus operations are performed component wise.

The binary hash $\mathbf{q}(\mathbf{x})$ generated by Equation 7 has the following properties [4]: given two vectors $\mathbf{x}$ and $\mathbf{x}'$ that have a Euclidean distance $d(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|$, the probability that the i^{th} bits, $q_i(\mathbf{x})$ and $q_i(\mathbf{x}')$ respectively of the hashes $\mathbf{q}(\mathbf{x})$ and $\mathbf{q}(\mathbf{x}')$ are identical depends only on the distance $d(\mathbf{x}, \mathbf{x}')$ between the vectors and not on $\mathbf{x}$ and $\mathbf{x}'$ themselves. The following relationship results as a consequence [4]: with probability at most $e^{-2t^2 M}$ the normalized (per-bit) Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ between $\mathbf{q}(\mathbf{x})$ and $\mathbf{q}(\mathbf{x}')$ is bounded by:

$$\frac{1}{2} - \frac{1}{2} e^{-\left(\frac{\pi\sigma d(\mathbf{q}(\mathbf{x}))}{\sqrt{2}\Delta}\right)^2} - t \leq d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}')) \leq \frac{1}{2} - \frac{4}{\pi^2} e^{-\left(\frac{\pi\sigma d(\mathbf{q}(\mathbf{x}))}{\sqrt{2}\Delta}\right)^2} + t$$

where t is a control factor. The import of the above bound is that the Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is correlated to the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ if $d(\mathbf{x}, \mathbf{x}')$ is less than a threshold (which depends on Δ). Specifically, for small $d(\mathbf{x}, \mathbf{x}')$, $E[d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))]$, the expected Hamming distance, can be shown to be bounded from above by $\sqrt{2\pi^{-1}}\sigma\Delta^{-1}d(\mathbf{x}, \mathbf{x}')$, which is linear in $d(\mathbf{x}, \mathbf{x}')$. However, if the distance between $\mathbf{x}$ and $\mathbf{x}'$ is greater than this threshold, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is bounded by $0.5 - 4\pi^{-2} exp\left(-0.5\pi^2\sigma^2\Delta^{-2}d(\mathbf{x}, \mathbf{x}')^2\right)$, which rapidly converges to 0.5 and effectively gives us no information whatsoever about the true distance between $\mathbf{x}$ and $\mathbf{x}'$.

Figure 3 illustrates this relation through a simulation. For the simulation we randomly generated samples in a high-dimensional space ($L = 1024$) and computed their hashes. Figure 3 shows the scatter of the normalized per-dimension Euclidean distance between pairs of these vectors, and the normalized per-bit Hamming distance between their hashes. The number of bits in the hashes is also shown in the figures. We note that in all cases, once the normalized Eu-

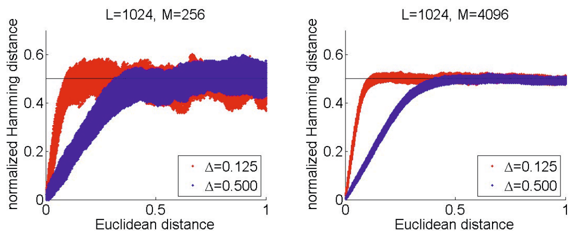


Fig. 3. Embedding behaviour for different values of Δ and different amounts of measurements M

clidean distance between two vectors exceeds Δ, the Hamming distance between their hashes no longer provides any information about the Euclidean distance. We also see that changing the value of the precision parameter Δ allows us to adjust the distance threshold until which the Hamming distance is informative. Increasing the number of bits M in the hashes also leads to a reduction of the variance of the Hamming distance.

A converse property of the embeddings is that for all $\mathbf{x}'$ except those that lie within a small radius of any $\mathbf{x}$, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ provides little information about how close $\mathbf{x}'$ is to $\mathbf{x}$ as evident from Figure 3. In fact, it can be shown that beyond this radius the embedding provides information theoretic security if the embedding parameters $\mathbf{A}$ and $\mathbf{w}$ remain unknown to a potential adversary who may try to deduce the data from their hashes. Any algorithm attempting to recover a signal $\mathbf{x}$ from its embedding $\mathbf{q}(\mathbf{x})$ or to infer anything about the relationship between two signals sufficiently far apart using only their embeddings will fail to do so. Furthermore, even in the case where $\mathbf{A}$ and $\mathbf{w}$ are known, it seems computationally intractable to derive $\mathbf{x}$ from $\mathbf{q}(\mathbf{x})$ unless one can guess a starting point very close to $\mathbf{x}$. In effect, it is infeasible to invert the SBE without strong *a priori* assumptions about $\mathbf{x}$.

4.2 Speaker Verification with Secure Embeddings

SBE hashes thus enable us to compare data that are close together, without actually being able to observe the data. This immediately enables us to apply them to the problem of privacy-preserving speaker authentication, through a rather direct modification of the algorithms described in Section 2.2. The manner in which we modify the algorithms is predicated on the fact that $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ is more-or-less linearly related to $d(\mathbf{x}, \mathbf{x}')$ for nearby vectors. We explain the actual modifications below.

In all cases we assume that the speaker possesses the embedding parameters $\mathbf{A}$ and $\mathbf{w}$. The supervectors for the speaker's recordings are computed on the user's client device. The SBE hashes of the supervectors are also computed on the user's client device using $\mathbf{A}$ and $\mathbf{w}$. For both the enrollment data and for subsequent authentication data only the hashes are transmitted to the system over a secure channel.

During the enrolment phase the system also requires hashes derived from a large collection of recordings from imposters.

We and defer the discussion of the provenance of $\mathbf{A}$ and $\mathbf{w}$ and the consequent implications on user privacy, as well as the source of the imposter recordings, to Section 4.6

SVM-Based Classifier: The SVM based classifier used for speaker authentication systems utilizes an RBF kernel defined by $k(\mathbf{x}, \mathbf{x}') = e^{-\gamma \cdot d^2(\mathbf{x}, \mathbf{x}')}$. We modify this by replacing the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ between vectors by the hamming distance between their hashes $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$. The resulting SVM utilizes the modified pseudo-kernel: $k_q(\mathbf{x}, \mathbf{x}') = e^{-\gamma \cdot d_H^2(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))}$. Note that for a given $\mathbf{A}$ and $\mathbf{w}$, the modified kernel $k_q(\mathbf{x}, \mathbf{x}')$ closely approximates the conventional RBF for small $d(\mathbf{x}, \mathbf{x}')$, but varies significantly from it at larger $d(\mathbf{x}, \mathbf{x}')$. Although it does not satisfy Mercer's conditions and cannot be considered a true kernel, in practice it is effective as we shall see in the experiments described in Section 4.3.

During enrollment, the SVM classifier is now trained using hashes of (supervectors derived from) enrollment recordings provided by the user, in conjunction with the hashes of a collection of imposter recordings. During operation, hashes of supervectors derived from test recordings are classified by the SVM.

***k*-NN Classifier:** The k-NN classifier of Section 2.4 can also be similarly modified. Here we replace the Euclidean distance $d(\mathbf{x}, \mathbf{x}')$ used to rank order the neighbors of a vector by the Hamming distance between their SBE hashes, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$. The motivation behind the design of the algorithm in Section 2.4, which was designed with the hashing-based classifiers in mind, can now be explained. Since the expected Hamming distance $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ between any the hashes of any two vectors $\mathbf{x}$ and $\mathbf{x}'$ with Euclidean distance $d(\mathbf{x}, \mathbf{x}') > M\Delta$ is $0.5M$, $d_H(\mathbf{q}(\mathbf{x}), \mathbf{q}(\mathbf{x}'))$ cannot reliably be used to choose the k closest neighbors to any $\mathbf{x}$ correctly except for small k. Hence we simply consider all exemplars to compute the score for any vector. Secondly, since the actual Hamming distance between the hashes of two vectors is a random variable and does not fall significantly above $0.5M$, particularly for larger M, weights are assigned to exemplars based on reciprocal rank, rather than the distance itself.

The actual implementation of the authentication is straight-forward. In the enrollment phase the system obtains the hashes of a set of enrolment recordings from the user, as well as those for a collection of recordings from several imposters. The k-NN algorithm is used thereafter to classify hashes derived from test recordings of the user.

4.3 Experiments in Speaker Authentication

Unlike the SFE-based framework which merely embedded a conventional algorithm within a privacy-preserving framework, the SBE-based mechanisms described above modify the actual classifier by changing the distance functions they use. Therefore, before we proceed, we first demonstrate the accuracy of the SBE-based speaker authentication.

Experiments were run on the YOHO Speaker Verification corpus [11]. It comprises of a collection of short utterances produced by 138 different speakers, 108 male and 30 female. Each utterance contains a sequence of three two-digit numbers (e.g. "26-81-57"). The recordings were sampled at 8kHz and digitized as 16-bit words. The corpus is divided into an enrollment set and a verification set. The enrollment set contains 96 utterances from each speaker, totaling 14.54 hours of audio. The verification set, which is the "test" set, contains 40 utterances from each speaker, totaling 6.24 hours of audio. We did not explicitly record imposters; instead for each of the 138 speakers in the corpus, the remaining 137 were used as imposters. All speech signals were parameterized into a sequence of MFCC feature vectors, each computed from an analysis frame of 25ms, at the rate of 100 vectors per second. The 13-dimensional MFCC vectors were augmented with the temporal differences and double-differences to result in a total of 39 features per frame.

A UBM (which, we recall is a Gaussian mixture) was trained from the data for all the speakers. The UBM was adapted to each recording from each speaker to obtain a single Gaussian supervector for each recording. The length of the supervectors depends on the number of Gaussians in the UBM: a UBM with

N Gaussians results in supervectors with $L = 39N$ dimensions. We evaluated UBMs of different sizes, with the number of Gaussian components ranging from 4 to 128 Gaussians, to find the optimal settings.

4.4 Experiments Using Supervectors

In the first experiment we directly employed the supervectors in the classifiers described in Section 2.2, to obtain reference speaker authentication results against which we could compare the performance of SBE-based authentication. We report results obtained from supervectors obtained with GMMs of different sizes. Results are reported using both the SVM and k-NN classifiers. In a binary classification task such as speaker authentication, the two key performance metrics are "precision", which records the percentage of all recordings that were "accepted" by the system that actually belonged to the target speaker, and "recall", which records the fraction of all recordings by a speaker that were correctly accepted. Ideally, both precision and recall must be close to 100%. We report performances in terms of "F-measure", which is the harmonic mean of precision and recall and encapsulates both. If either of them is low, the F-measure will be low. Note that for the k-NN classifier which actually obtains separate scores for each speaker in the corpus, the F-measure is also identical to the accuracy of a multi-class classifier that attempts to identify the speaker in each recording.

The obtained results averaged over all the speakers and are presented in Tables 2 and 3 respectively. In both cases the classifier achieves a close to optimal

Table 2. Speaker authentication F-measure (%age) using SVMs

#Gaussians	4	8	16	32	64	128
F-measure	76.8	89.2	92.8	94.0	94.1	94.4

Table 3. Speaker authentication F-measure (%age) using k-NN

#Gaussians	4	8	16	32	64	128
F-measure	71.8	86.7	92.4	94.8	91.5	85.4

performance with supervectors derived from mixtures of 32 Gaussians – for the k-NN classifier the performance falls off with increasing Gaussians; for the SVMs, although further increase in suprevector size results in improved performance, the improvements are marginal. Consequently, for experiments with the SBE we will consider only SBE hashes of supervectors obtained from mixtures of 32 Gaussians.

4.5 Experiments Using SBE Hashes

The secure binary embeddings have two parameters that can be varied: the quantization step size Δ and the number of bits M. The value of M by itself is not informative, since increasing L (dimensionality of the supervector) requires increasing values of M to retain the same resolution. Hence we report our results as a function of *bits per coefficient* (*bpc*), computed as M/L. The results obtained for the SVM and k-NN classifiers are presented in Tables 5 and 4 respectively.

Table 4. Speaker authentication F-measure (%age) using k-NN + SBE

Δ	13.5	14.0	14.5	15.0	15.5
bpc=4	72.4	78.9	82.8	85.6	87.3
bpc=8	85.0	88.6	89.9	90.8	91.6
bpc=16	90.2	91.8	92.7	93.1	93.2

Table 5. Speaker authentication F-measure (%age) using SVMs + SBE

Δ	13.5	14.0	14.5	15.0	15.5
bpc=4	85.4	87.9	89.0	90.9	91.6
bpc=8	90.7	92.1	93.0	93.7	93.9
bpc=16	94.0	94.0	94.6	94.6	94.9

From Figure 3 and the discussion in Section 4.1 we can infer that smaller Δ values result in smaller neighborhoods over which the distances between adjacent vectors can be deduced from their hashes. Thus, smaller Δ values result in hashes $\mathbf{q}(\mathbf{x})$ that better obscure the value of the vectors $\mathbf{x}$. By the same token, however, they are also likely to result in poorer classification accuracy since $\mathbf{x}$ cannot reliably be compared to a model.

Tables 5 and 4 report performance as a function of Δ and *bpc*. As expected, increasing Δ generally improves classification performance, but then so does increasing *bpc*. For small *bpc* small Δ values result in unacceptably poor classification. However increasing *bpc* improves the classification performance in all cases, and at *bpc*=16, the performance stabilizes quickly and is thereafter largely independent of Δ. At these settings we notice little, if any degradation when we work from SBE hashes instead of the original supervectors.

4.6 Privacy and Other Practical Issues

Let us now analyze the actual privacy afforded by the SBE-based authentication.

When the User Possesses A and w as Private Keys: In the most secure case, the *user* generates $(\mathbf{A}, \mathbf{w})$ and retains it as his private key. Since the system only obtains the hashes and does not know $\mathbf{A}$ or $\mathbf{w}$, it provably has no means of recovering vectors from their hashes. Moreover, it also has no way of *computing* hashes given a vector. As a result, the process is totally secure from abuse by the system. Moreover, this also protects the user from imposters: the user's private key is tied to her client device(s), and if these are secured, *e.g.* by a password, the system cannot be broken into by an imposter who poses as the user.

The problem here is that since the system has no means of computing hashes, the *user* is in charge of downloading imposter data and generating hashes from them for the system to train its classifiers from, an onerous responsibility. One alternative is to use a "single-class" classifier, which only requires positive instances from the user; however the performance of such classifiers tends to be poor [3]. A more practical solution is for the user to generate his set of negative samples from the positive samples. This can be achieved by using the method proposed in [12], in which the user iteratively estimates SVM classifiers and generates negative instances that lie on the wrong side of the decision boundary. The authors of this technique state that after a few iterations the classification results are comparable to the ones obtained using sets of genuine positive and negative samples; however whether this is true for supervectors of speech recordings remains to be seen.

The *System* Provides A and w: Recall that the SBE is essentially not invertible without strong *a priori* assumptions about the data. Under the assumption of non-invertibility, we can propose the following: the system generates $(\mathbf{A}, \mathbf{w})$ once, and transmits it to the speaker. The system is now in charge of generating imposter hashes. Users only send the SBE hashes of their enrollment data to the system. Even if the hashes are not invertible, this process is not secure: since the system can generate hashes on its own, it does not prevent the system from searching public sources such as YouTube for instances of the speaker's voice. This may not be an unacceptable compromise in many situations; however, it is still unknown whether the system can recover a vector from its hash given *a priori* information that it can obtain.

Finally, we note that one of the key advantages of the SFE approach was its generalizability – the same framework could also be employed for a variety of other tasks such as speaker *identification*, speech recognition, etc. Since the hashing-based approach can only be guaranteed to be secure if the *user* retains the hashing parameters, it does not lend itself to such generalization.

5 Discussion and Conclusions

With increasing use of speech-based services, the issue of the privacy of the users and their speech data is only just beginning to be considered. These issues go beyond the mere problem of biometrics. Even for speech-based biometrics, the proposed frameworks are not complete – while they enable some operations, specifically speaker authentication, to be performed securely, they cannot provide the same security to closely related tasks such as speaker identification. In the larger scenario, users who use speech recognition and speech mining services, social media sites etc. remain exposed.

The presented work can thus only be considered a first step towards establishing a truly secure framework for speech processing applications in general. We continue to investigate these issues and hope researchers around the world recognize the importance of the problem and take it up as well.

Acknowledgements. The authors would like to thank Petros Boufounos for many helpful suggestions. José Portêlo and Isabel Trancoso were supported by Portuguese funds through FCT – Fundação para a Ciência e a Tecnologia, under PhD grant SFRH/BD/71349/2010 and project PTDC/EIA-CCO/122542/2010. Bhiksha Raj was partially supported by NSF Grant 1017256.

References

1. Adler, A.: Biometric System Security. In: Jain, A.K., Flynn, P., Ross, A. (eds.) Handbook of Biometrics. Springer (2007)
2. Pathak, M., Raj, B.: Privacy Preserving Speaker Verification using adapted GMMs. In: Proc. Interspeech (2011)
3. Pathak, M., Raj, B.: Privacy-Preserving Speaker Verification as Password Matching. In: Proc. ICASSP (2012)
4. Boufounos, P., Rane, S.: Secure Binary Embeddings for Privacy Preserving Nearest Neighbors. In: Proc. Workshop on Information Forensics and Security, WIFS (2011)
5. Boufounos, P.: Universal Rate-Efficient Scalar Quantization. IEEE Trans. on Information Theory 58(3), 1861–1872 (2012)
6. Davis, S.B., Mermelstein, P.: Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. IEEE Transactions on Acoustics, Speech, and Signal Processing 28(4), 357–366 (1980)
7. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker Verification using Adapted Gaussian Mixture Models. Digital Signal Processing 10(1-3), 19–41 (2000)
8. Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P.: Joint Factor Analysis Versus Eigenchannels in Speaker Recognition. IEEE Trans. Audio, Speech and Language Processing 15(4), 1435–1447 (2007)
9. Sennoussaoui, M., Kenny, P., Brummer, N., de Villiers, E., Dumouchel, P.: Mixture of PLDA Models in I-Vector Space for Gender-Independent Speaker Recognition. In: Proc. Interspeech 2011, Florence, Italy (August 2011)
10. Campbell, W.M., Campbell, J.R., Reynolds, D.A., Singer, E., Torres-Carrasquillo, P.A.: Support Vector Machines for Speaker and Language Recognition. Computer Speech and Language 20, 210–229 (2006)
11. Campbell, J.P.: Testing with the YOHO CD-ROM voice verification corpus. In: Proc. ICASSP (1995)
12. Yu, H., Han, J., Chang, K.C.C.: PEBL: Positive Example based Learning for Web Page Classification using SVM. In: Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 239–248. ACM (2002)
13. Lindell, Y., Pinkas, B.: An Efficient Protocol for Secure Two-Party Computation in the Presence of Malicious Adversaries. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 52–78. Springer, Heidelberg (2007)
14. Kinnunena, T., Li, H.: An overview of text-independent speaker recognition: From features to supervectors. Speech Communication 52(1), 12–40 (2010)
15. Smaragdis, P., Shashanka, M.: A Framework for Secure Speech Recognition. IEEE Transactions on Audio, Speech, and Language Processing 15(4), 1404–1413
16. Pathak, M., Rane, S., Sun, W., Raj, B.: Privacy Preserving Probabilistic Inference with Hidden Markov Models. In: Proc. ICASSP, Prague, Czech Republic (May 2011)
17. Prabhakar, S., Pankanti, S., Jain, A.K.: Biometric recognition: security and privacy concerns. IEEE Security & Privacy 1(2), 33–42

18. Reynolds, D.A., Quatieri, T.F., Dunn, R.B.: Speaker Verification Using Adapted Gaussian Mixture Models. Digital Signal Processing 10(1-3), 19–41 (2000)
19. Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-García, J., Petrovska-Delacrétaz, D., Reynolds, D.A.: A Tutorial on Text-Independent Speaker Verification. EURASIP Journal on Advances in Signal Processing 4, 430–451 (2004)
20. Reynolds, D.A.: Comparison of Background Normalization Methods for Text-Independent Speaker Verification. In: Proceedings of the European Conference on Speech Communication and Technology (September 1997)
21. Shou-Chun, Rose, R., Kenny, P.: Adaptive score normalization for progressive model adaptation in text independent speaker verification. In: Proc. ICASSP, Las Vegas, Nevada, USA (April 2008)
22. 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)
23. Kershbaum, F., Biswas, D., de Hoogh, S.: Performance Comparison of Secure Comparison Protocols. In: Proceedings of the 2009 20th International Workshop on Database and Expert Systems Application, pp. 133–136 (2009)
24. Quisquater, J.-J., Guillou, L.C., Berson, T.A.: How to Explain Zero-Knowledge Protocols to Your Children. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 628–631. Springer, Heidelberg (1990)
25. Gionis, A., Indyk, P., Motwani, R.: Similarity Search in High Dimensions via Hashing. In: Proceedings of the 25th Very Large Database (VLDB) Conference (1999)

Differential Attacks on Reduced RIPEMD-160

Florian Mendel[1], Tomislav Nad[2], Stefan Scherz, and Martin Schläffer[2]

[1] Katholieke Universiteit Leuven, ESAT/COSIC and IBBT, Belgium
[2] Graz University of Technology, IAIK, Austria

Abstract. In this work, we provide the first security analysis of reduced RIPEMD-160 regarding its collision resistance with practical complexity. The ISO/IEC standard RIPEMD-160 was proposed 15 years ago and may be used as a drop-in replacement for SHA-1 due to their same hash output length. Only few results have been published for RIPEMD-160 so far and most attacks have a complexity very close to the generic bound. In this paper, we present the first application of the attacks of Wang et al. on MD5 and SHA-1 to RIPEMD-160. Due to the dual-stream structure of RIPEMD-160 the application of these attacks is nontrivial and almost impossible without the use of automated tools. We present practical examples of semi-free-start near-collisions for the middle 48 steps (out of 80) and semi-free-start collisions for 36 steps of RIPEMD-160. Furthermore, our results show that the differential characteristics get very dense in RIPEMD-160 such that a full-round attack seems unlikely in the near future.

Keywords: hash functions, cryptanalysis, semi-free-start collisions.

1 Introduction

In the last decade, several significant advances have been made in the field of hash function research. Especially the collision attacks [16–18] on the MD4 family of hash functions, in particular on MD5 and SHA-1, have weakened the security assumptions of many commonly used hash functions. As a consequence, NIST is organizing the SHA-3 competition to evaluate alternative hash function designs and choose a new hash function standard in 2012 [11]. During this ongoing evaluation, not only the three classical security requirements (preimage resistance, second preimage resistance and collision resistance) are considered. Researchers also analyze (semi-) free-start collisions, near-collisions, and any other non-random behavior of a hash function or its building blocks. Commonly, also simplified or round-reduced variants are studied to get new insights in the design and strength of a cryptographic primitive.

RIPEMD-160 is a cryptographic hash function which was designed by Dobbertin et al. in 1996 [5] and standardized by ISO/IEC in 1997 [6]. As a part of the ISO/IEC 10118-3 standard on dedicated hash functions, RIPEMD-160 is used in many applications and is part of several standards, e.g. OpenSSL, OpenPGP. Furthermore, RIPEMD-160 is often recommended as a drop-in replacement for

SHA-1 due to their same output length. Even though RIPEMD-160 relies on the same design principles as MD5 and SHA-1, the dual-stream structure makes RIPEMD-160 more secure against recent attacks on other members of the MD4 family. For this reason, only few results on RIPEMD-160 have been published to date.

The only work regarding the collision resistance of RIPEMD-160 has been published by Mendel et al. [9]. In this work, the application of the differential attacks on RIPEMD by Dobbertin [4] and Wang et al. [16] has been studied. However, due to the increased number of steps and the two streams are more different than in RIPEMD, they concluded that RIPEMD-160 might be secure against these types of attacks. The best currently known attack on the hash function RIPEMD-160 is a preimage attack for 31 (out of 80) steps by Ohtahara et al. [12]. However, the complexity of the attack is very close to the generic complexity of 2^{160}. Recently, Sasaki and Wang [14] have shown non-random properties for up to 51 steps when starting from round 2. However, the complexity of the attack is very high 2^{158} and the attack setting is much weaker than in a collision attack.

In this paper, we provide the first analysis of unmodified RIPEMD-160 against collision attacks. We show how the collision attacks of Wang et al. can be applied on up to 3 rounds of the RIPEMD-160 compression function. We present semi-free-start collisions for 36 steps and semi-free-start near-collisions for 48 steps when starting at round 2. Contrary to all previous results, our results have a very low complexity and we are able to show practical examples in all cases. Although we were not able to attack the reduced round hash function, our results provide a significant improvement in the analysis of the collision resistance of RIPEMD-160 and gives new insights in its security.

The paper is structured as follows. In Sect. 2 we briefly describe the dual-stream hash function RIPEMD-160. In Sect. 3 we present different strategies to construct collisions for round-reduced RIPEMD-160 using local collisions in both streams. In Sect. 4, we show in detail how to find high-probability differential characteristics and confirming inputs using an automatic search tool. Finally, we conclude in Sec. 5.

2 Description of RIPEMD-160

RIPEMD-160 was designed by Dobbertin, Bosselaers and Preneel in 1996 as a replacement for RIPEMD [5] and is part of the international standard ISO/IEC 10118-3:2004 on dedicated hash functions. It is an iterative hash functions based on the Merkle-Damgård design principle [2,10] and produces a 160-bit hash value by processing message blocks of 512 bits. Like its predecessor RIPEMD, the compression function of RIPEMD-160 consists of two parallel streams. The two streams of RIPEMD-160 are designed more differently than those of RIPEMD. In each stream the expanded message block is used to update the state variables.

After the computations the results of both streams are combined with the chaining input, which is illustrated in Fig. 1 and defined as follows:

$$h_0 = B_{-1} + B_{78} + (B'_{77} \lll 10) \qquad h_3 = B_{-4} + (B_{75} \lll 10) + B'_{79}$$
$$h_1 = B_{-2} + (B_{77} \lll 10) + (B'_{76} \lll 10) \qquad h_4 = B_{-5} + B_{79} + B'_{78}$$
$$h_2 = B_{-3} + (B_{76} \lll 10) + (B'_{75} \lll 10)$$

The final values of one iteration $h_0, \ldots, h_4$ are either the final hash value or the chaining input for the next message block.

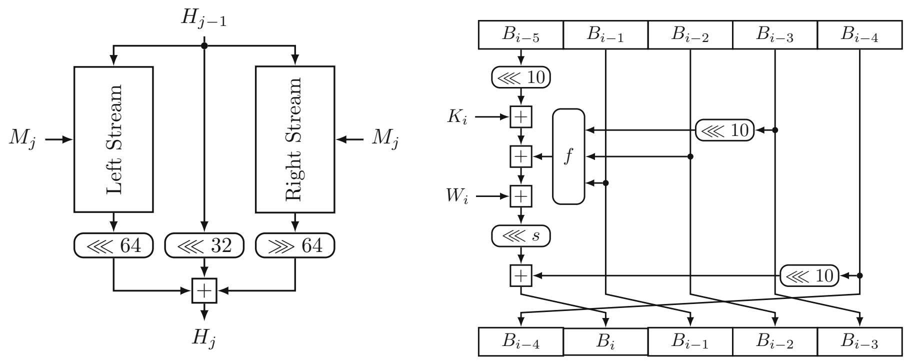


Fig. 1. The compression function (left) and state update transformation (right) of RIPEMD-160.

State Update Transformation. The five 32-bit input state variables of both streams $h_0 = B_{-5} = B'_{-5}$ and $h_i = B_{-i} = B'_{-i}$ with $1 \le i \le 4$ are initialized with the initial value (first block) or the previous chaining values. The state update transformation updates the five state variables in five rounds of 16 steps each using one expanded message word W_i in each step.

Note that he Boolean functions f and rotation values s are different in each stream and each round. f_r is used for the r-th round in the left stream and f_{6-r} for the r-th round in the right stream with $1 \le r \le 5$. For the rotation values s and the constants K_i we refer to [5].

$$f_1(X,Y,Z) = X \oplus Y \oplus Z \qquad f_4(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$
$$f_2(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z) \qquad f_5(X,Y,Z) = X \oplus (Y \vee \neg Z)$$
$$f_3(X,Y,Z) = (X \vee \neg Y) \oplus Z$$

Message Expansion. The message expansion of RIPEMD-160 is a round-wise permutation of the 16 message block words. For the left and the right stream different permutations are used.

	Step	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	Round 1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Left	Round 2	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
	Round 3	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
Stream	Round 4	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
	Round 5	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13
	Round 1	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
Right	Round 2	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
	Round 3	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
Stream	Round 4	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
	Round 5	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

3 Constructing (Local) Collisions for RIPEMD-160

In this section, we provide a general outline how to construct collisions for RIPEMD-160. The idea is based on the recent differential attacks on the MD4 family of hash functions [16, 18] and its application to the dual stream hash function RIPEMD-128 in [8]. The high-level strategy is basically the same in all attacks and can be summarized as follows:

1. Find a characteristic for the hash function that holds with high probability after the first round of the hash function.
2. Find a characteristic (not necessary with high probability) for the first round of the hash function.
3. Use message modification techniques to fulfill conditions imposed by the characteristic in the first round to increase the probability of the characteristic.
4. Use random trials to find values for the remaining free message bits such that the message follows the characteristic.

The most difficult and important part of the attack is to find a good differential characteristic for both the first round and the remaining rounds of the hash function, since this defines the final attack complexity.

3.1 Constructing High-Probability Characteristics for RIPEMD-160

In a differential attack on hash functions, we first need to construct differential characteristics that hold with high probability. In general, a characteristic has a high probability if the number of differences and conditions imposed by the differential characteristic is small. We refer to such a characteristic to be sparse. For single-stream hash functions, a characteristic does not need to be sparse in the first round, since we can use basic message modification [18] to deterministically construct conforming message pairs. However, in the case of dual-stream hash functions, a single message word is used to update two streams in each round which complicates message modification. Therefore, we try to construct sparse characteristics also in at least one stream of the first round such that the message modification part can be carried out more efficiently.

For MD5 a very sparse characteristic with only differences in the MSB of the chaining variable exists, which can be used to construct long high-probability characteristics [1]. RIPEMD-160 consists of two MD5-like step update transformations. However, due to the additional rotation of the state variable B_{i-4} this high-probability characteristic does not exist in RIPEMD-160. Moreover, because of the additional rotation also differences spread rather quickly such that differential characteristics get dense easily. Therefore, also using a linearized approximation of the hash function and algorithms from coding theory (as it is done for instance in the attacks on SHA-1 [13]) does not result in sparse characteristics [9]. The best choice is to use local collisions, which result in large areas which do not contain any differences at all. This strategy is usually advantageous for hash functions using permuted message words in the message expansion.

We start by constructing a very sparse (high probability) characteristic for the hash function after the first round. Then a suitable characteristic for the first round needs to be constructed. The goal is to use one or more message words to construct short local collisions within only a few steps in both streams of the later rounds. This is obviously more difficult for dual-stream hash functions, since more constraints have to be fulfilled. In particular, the different message word permutations and rotation values in each stream of RIPEMD-160 make the construction of many short local collisions difficult.

3.2 Local Collisions

In MD4-like hash functions, a local collision has to start and end by a difference in a message word. We basically have three options to construct local collisions for two rounds which are shown in Fig. 2. First, we can use differences in two message words, to construct local collisions within each round. Second, we can use differences in a single message word, to construct local collisions spanning over two rounds. Thirdly, we can combine the two approaches or use even more message word differences.

Remember that we aim for a high probability differential characteristics after the first round in both streams. This can be achieved by using short local collisions in the second round. Another possibility is to use local collisions spanning over two rounds and cancel the differences very early in the second round in each stream.

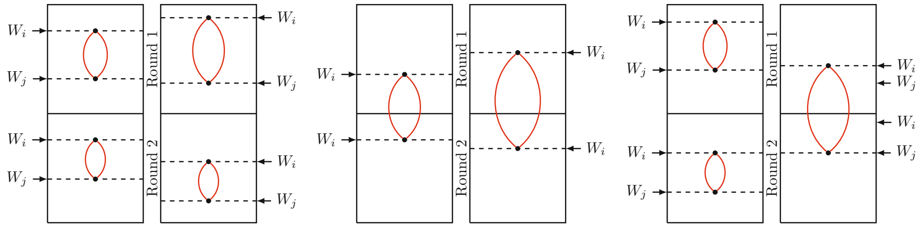


Fig. 2. Three options to construct local collisions for two rounds

To get high-probability differential characteristics, we aim for local collisions over as few steps as possible. The minimum number of steps for a local collision in RIPEMD-160 is five since a difference has to pass through all 5 state words. A single 5-step local collision can easily be constructed using differences in at least three message words depending on the Boolean function f_r. Assuming that the differences at the input of the Boolean function can always be absorbed, one needs only differences in the message words that are used in step one, four and five of the local collision. Note that if the differences can not be absorbed also differences in the other message words might be needed.

However, even if we have differences in three message words, we need to construct up to four (short) local collisions. This places several constraints on the message words. Possible candidates for message words which contain differences are given in Sect. 3.3.

If differences in a single message word are used, the local collisions have to be constructed over two rounds. The advantage is that we only need two local collisions instead of four. In general, this places less constraints on the message words and may also lead to sparser characteristics in the third round. However, this approach has consequences on the minimum number of steps of a local collision:

Observation 1. *The shortest local collision, which uses difference(s) in a single message word, has to be constructed over six steps.*

The reason for this is the update process of RIPEMD-160. A 5-step local collision only allows non-zero differences in a single state variable, more precisely if a message word W_j is introducing a difference in step i, only the state variable B_i contains a difference. This difference can be canceled five steps later using the same message word. However, at the input of step $i+4$ of the 5-step local collision we get the following setting:

$$\Delta B_{i-1} = 0, \Delta B_{i+3} = 0, \Delta B_{i+2} = 0, \Delta B_{i+1} = 0 \text{ and } \Delta B_i \neq 0 \text{ and } \Delta W_k = 0$$

Then, the following step update transformation has to lead to a zero difference in $\Delta B_{i+4} = 0$ in order to produce a 5-step local collision:

$$\overbrace{B_{i+4}}^{\Delta=0} = \overbrace{(((B_{i-1} \lll 10) + f(B_{i+3}, B_{i+2}, (B_{i+1} \lll 10)) + W_k + K_{i+4}) \lll s)}^{\Delta=0} + \underbrace{(B_i \lll 10)}_{\Delta\neq 0}$$

However, this equation leads to a contradiction since only one term contains a non-zero difference. Hence, a 5-step local collision cannot be constructed using a difference in a single message word. Note that this restriction also applies to local collisions constructed within one round using only differences in two distinct message words.

Since RIPEMD-160 consists of two streams with different permutations of message words, both streams need to be considered concurrently. The first step in the attack is to determine those message words, which may contain differences in order to lead to local collisions. We have several constraints regarding those local collisions such that the whole attack can be carried out efficiently.

3.3 Choosing Message Word Differences

In this section, we describe the different options to construct sparse local collision in RIPEMD-160. The best result and the semi-free-start near-collision on 48 steps (see Sect. 4) was obtained using a difference in a single message word. However, we have also analyzed the use of differences in two or three message words and present local collisions for these cases as well.

Using a single message word difference, we need to construct local collisions between round 1 and 2 of RIPEMD-160. If we consider only a difference in a single message word, we get a sparse differential characteristic in the second round if the local collisions end as early as possible in both streams. Using W_7, we can construct local collisions which end in the first and fourth step of the second round in the left and right stream, respectively. The candidates, which can construct local collisions that end in the first half of round 2 in both streams are given in Table 1.

Table 1. Local collision candidates (single message word)

Message	Local Collision Lengths	
Word	Left Stream	Right Stream
W_7	9 steps (step 7 to 16)	17 steps (step 2 to 19)
W_6	15 steps (step 6 to 21)	7 steps (step 9 to 16)
W_{10}	10 steps (step 10 to 20)	10 steps (step 13 to 23)

Unfortunately, it is very hard to find a corresponding differential characteristic for the first round (also see Section 4). Due to the XOR-function used in round 1 of the left stream and the rather short local collision (9 steps), we did not succeed in finding a corresponding differential characteristic. However, due to the repeating pattern in the message expansion of RIPEMD-160, we can use a local collision of the same length and position between round 2 and 3 using message word W_3 by skipping the first round. Note that this setting is used for the main attack of this paper and is also shown in Fig. 3.

Note that a single 5-step local collision can be constructed easily using differences in multiple message words. Since the Boolean functions in the second round of both streams can absorb the differences, three message words are sufficient to construct 5-step local collisions. However, due to the message permutation, it is not possible to construct 5-step local collisions in the second round of both streams concurrently. We list the shortest local collision for the second round

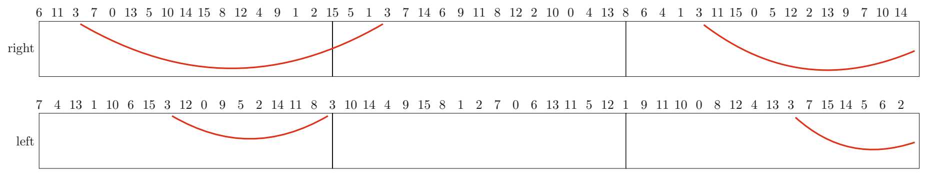


Fig. 3. Using message word W_3 and round 2-4

and the resulting local collisions for the first round in Table 2. Unfortunately, we did not find high-probability characteristics using differences in three message words.

Table 2. Local collision candidates (triples of message words)

Triples of Message Words	Local Collision Lengths			
	Left Stream		Right Stream	
	Round 1	Round 2	Round 1	Round 2
(W_4, W_{10}, W_{12})	8 steps (4 - 12)	7 steps (17 - 24)	8 steps (7 - 15)	5 steps (23 - 28)
(W_5, W_9, W_{15})	10 steps (5 - 15)	5 steps (22 - 27)	10 steps (0 - 10)	7 steps (22 - 29)
(W_4, W_{10}, W_{12})	20 steps (4 - 24)		8 steps (7 - 15)	5 steps (23 - 28)

Another possibility is to use differences in two distinct message words. Only three message pairs can construct 6-step local collision in round 2 of both streams concurrently and suitable local collisions in round 1. Those pairs of message words are presented in Table 3 as well as some candidates for a combined approach. In such 6-step local collisions, two state variables have to contain differences. Due to different rotation values and the additional modular addition in the state update process, it is not possible to use a single bit difference in both state variables concurrently. Moreover, we need a long carry expansion to cancel differences in the modular addition which results in not so sparse characteristics.

Table 3. Local collision candidates (pairs of message words)

Pairs of Message Words	Local Collision Lengths			
	Left Stream		Right Stream	
	Round 1	Round 2	Round 1	Round 2
(W_0, W_8)	8 steps (0 - 8)	6 steps (25 - 31)	8 steps (3 - 11)	6 steps (20 - 26)
(W_7, W_{15})	8 steps (7 - 15)	6 steps (16 - 22)	8 steps (2 - 10)	6 steps (19 - 25)
(W_3, W_{14})	11 steps (3 - 14)	6 steps (23 - 29)	13 steps (1 - 14)	6 steps (18 - 24)
(W_{12}, W_{13})	12 steps (12 - 24)		7 steps (8 - 15)	6 steps (21 - 27)
(W_7, W_{15})	15 steps (7 - 22)		8 steps (2 - 10)	6 steps (19 - 25)
(W_9, W_{10})	17 steps (9 - 26)		9 steps (4 - 13)	6 steps (23 - 29)

4 Collision Attacks on RIPEMD-160

In this section, we show how to find semi-free-start near-collisions for 48 steps and semi-free-start collisions for 36 steps of RIPEMD-160 using an automated search tool. The results are obtained using the middle 3 rounds of RIPEMD-160 and a single bit difference in message word W_3.

4.1 Automatic Search Tool

To find a differential characteristic and confirming inputs after fixing the message words which contain differences requires an advanced set of techniques and tools. Due to the increased complexity of RIPEMD-160 compared to RIPEMD, finding good (high probability) differential characteristics by hand is almost impossible. Hence, we have used an automatic tool which can be used for finding complex nonlinear differential characteristics as well as for solving nonlinear equations involving conditions on state words and message words. The tool is based on the approach of Mendel et al. to find both complex nonlinear differential characteristics and conforming message pairs for RIPEMD-128 [8] and SHA-256 [7].

The basic idea is to consider differential characteristics which impose arbitrary conditions on pairs of bits using generalized conditions [3]. Generalized conditions are inspired by signed-bit differences and take all 16 possible conditions on a pair of bits into account. Table 4 lists all these possible conditions and introduces the notation for the various cases.

Table 4. Notation for possible generalized conditions on a pair of bits [3].

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

By considering the propagation of these generalized conditions in a bit sliced way we can construct differential characteristics efficiently. The basic idea of the search algorithm is to randomly pick a bit from a set of bit positions with predefined conditions, impose a more restricted condition and compute how this new condition propagates. This is repeated until an inconsistency is found or all unrestricted bits from the set are eliminated. Note that this general approach can be used for both, finding differential characteristics and conforming message pairs.

For example, the search strategy for finding nonlinear characteristics works as follows (for a more detailed description of the search algorithm or how the conditions are propagated we refer to [7]):

1. Define a set of unrestricted bits (`?`) and differences (`x`).
2. Pick a random bit from the set.
3. Impose a zero-difference (`-`) on unrestricted bits (`?`), or randomly choose a sign (`u` or `n`) for differences (`x`).
4. Check how the new conditions propagate.
5. If an inconsistency occurs, remember the last bit and jump back until this bit can be restricted without leading to a contradiction.
6. Repeat from step 2 until all bits from the set have been restricted.

Note that in RIPEMD-160 we have two modular additions (separated by a rotation operation) within one state update. Therefore, two different carry expansions may occur and by only picking random bits of B_i respectively B'_i in the search, the conditions propagate very slowly. Hence, we also consider output bits of the first modular addition and impose more restrictions on these conditions. This way contradictions are detected much earlier, which improves the search significantly.

We use the same strategy to find conforming input pairs for a given differential characteristic. Instead of picking an unrestricted bit (`?`) we pick an undetermined bit without difference (`-`) and assign randomly a value (`0` or `1`) until a solution is found:

1. Define a set of undetermined bits without difference (`-`).
2. Pick a random bit from the set.
3. Randomly choose the value of the bit (`0` or `1`).
4. Check how the new conditions propagate.
5. If an inconsistency occurs, remember the last bit and jump back until this bit can be restricted without leading to a contradiction.
6. Repeat from step 2 until all bits from the set have been restricted.

Note that the efficiency of finding a conforming message pair can be increased if the undetermined bits without difference (`-`) are picked in a specific order. The order strongly depends on the specific hash function. In general, fully determining word after word turns out to be a good approach. It can be used to find solutions without the need for hand-tuned advanced message modification techniques.

4.2 Finding a Differential Characteristic

The starting point for the search tool to find a semi-free-start near-collision on 48 steps is given in Table 6. Note that we do not fix the message difference prior to the search to allow the tool to find a good solution. In order to get a differential characteristics resulting in a low attack complexity, we aim for a low Hamming weight difference in the message words, and hence, also in the state

Table 5. Semi-free-start near-collision for round 2-4 (48 steps) of RIPEMD-160

H_0	b23f78a3 7775d378 20806ef8 8d6b662d 4f669598
M_1	2e3d54df e568a9cd d5e45e10 52f4e41a bb1bcda9 ffd073a6 ffe9b7f6 bfe436a9
	1273b786 b4ce0002 254a969e 359b7260 817f9eda ef3fff6d bc5068f5 2c3fc390
M_1^*	2e3d54df e568a9cd d5e45e10 52f4f41a bb1bcda9 ffd073a6 ffe9b7f6 bfe436a9
	1273b786 b4ce0002 254a969e 359b7260 817f9eda ef3fff6d bc5068f5 2c3fc390
ΔM_1	00000000 00000000 00000000 00001000 00000000 00000000 00000000 00000000
	00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
H_1	d98051ed e2a12c89 a16b3753 e8764785 1cb36d97
H_1^*	d98053ed e3a16c89 a16b3753 e8764785 1cb36d97
ΔH_1	00000200 01004000 00000000 00000000 00000000

words after the first 16 steps, i.e. after step 32. Hence, we first search for a good characteristic in this area.

We start by searching for a sparse differential characteristic only in state words B'_{24} and B'_{30} in the right stream. This way the search space gets reduced significantly on one hand, but also the resulting differential characteristics get sparser after step 24 in the right stream. This in turn simplifies finding conforming inputs later. We are able to find a differential characteristics with a single bit differences in W_3 (see Table 7) and only few conditions after step 32, which results in an overall low attack complexity.

We continue the search for the remaining parts of the differential characteristic. Using our search tool, we are able find many differential characteristic for the left and right stream. The differential characteristic for round 2-4 (48 steps) of RIPEMD-160 is given in Table 8.

4.3 Finding a Confirming Message Pair

To fulfill all conditions imposed by the differential characteristic in the first 16 steps (steps 16-31), we need to apply message modification techniques. Since we have many conditions in the first steps of the left and right stream this may not be an easy task. However, using our tool and generalized conditions, we can do message modification for the first 16 steps quite efficiently. Of course, by hand-tuning basic message modification the complexity might be significantly improved, but using our tool this phase of the message search can be automated and still be done quite efficiently. It can be summarized as follows.

– Since the first steps in the right stream are very dense, we start with guessing the remaining free bits in the state words B'_{16} to B'_{22}. This determines the message words W_5 and W_{13}.
– Next we guess the state words B_{26} to B_{16} in the left stream. This determines B_{-1}, B_{-2}, B_{-3} and most of the message words, except W_4, W_{11} and large parts of W_2, W_8.
– Finally, we guess all free bits in the remaining message words to determine the remaining state variables and to find a confirming message pair.

The resulting semi-free-start near-collision for three rounds (steps 16-63) of RIPEMD-160 is given in Table 5. Note that the same input also leads to a collision for 36 steps of RIPEMD-160. Furthermore, one can combine two semi-free-start near-collision as given in Table 5 to construct a second-order differential collision (4-sum) with practical complexity for 48 steps of RIPEMD-160, which also improves the result of Sasaki and Wang in [15] by 8 steps.

5 Conclusions and Future Work

In this work, we have presented new results on the dual-stream hash function RIPEMD-160 standardized by ISO/IEC. To be more precise, we show how the collision attacks of Wang et al. on MD5 and SHA-1, and the recent attack on RIPEMD-128 by Mendel et al. can be extended to RIPEMD-160. We have presented practical semi-free-start near-collisions for 48 out of 80 steps and semi-free-start collisions for 36 steps by skipping the first round. Our results improve upon previous results in a number of ways. First, we have increased the number of steps for which (near-) collisions for the compression function of RIPEMD-160 can be found. Second, our attacks have a very low complexity and we are even able to show practical examples.

Unfortunately, it is very hard to find a similar attack including the first round of RIPEMD-160. Due to the XOR-function used in round 1 of the left stream we did not succeed in finding a corresponding differential characteristic. It is part of future work to apply the attack also to round 1-3 which will probably require many improvements in the automated tool. Furthermore, to find (near-) collisions also for the reduced hash function of RIPEMD-160 where the chaining value is fixed, we need to construct sparser local collisions with more freedom to perform message modification.

The ideas and techniques in this paper may also be used in attacks on other hash functions, which update more state variables using a single message word, like SHA-2 or the SHA-3 candidates Blake and Skein.

Acknowledgments. This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007), 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. In addition, this work was supported by the Research Fund KU Leuven, OT/08/027 and by the Austrian Science Fund (FWF, project P21936).

References

1. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD-5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
2. Damgård, I.B.: A Design Principle for Hash Functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)

3. De Cannière, C., Rechberger, C.: Finding SHA-1 Characteristics: General Results and Applications. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 1–20. Springer, Heidelberg (2006)
4. Dobbertin, H.: RIPEMD with Two-Round Compress Function is Not Collision-Free. J. Cryptology 10(1), 51–70 (1997)
5. 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)
6. International Organization for Standardization: ISO/IEC 10118-3:2004. Information technology – Security techniques – Hash-functions – Part 3: Dedicated hash-functions (2004), http://www.iso.org/
7. Mendel, F., Nad, T., Schläffer, M.: Finding SHA-2 Characteristics: Searching through a Minefield of Contradictions. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 288–307. Springer, Heidelberg (2011)
8. Mendel, F., Nad, T., Schläffer, M.: Collision Attacks on the Reduced Dual-Stream Hash Function RIPEMD-128. In: Canteaut, A. (ed.) FSE 2012. LNCS, vol. 7549, pp. 226–243. Springer, Heidelberg (2012)
9. 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)
10. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
11. National Institute of Standards and Technology: Announcing Request for Candidate Algorithm Nominations for a New Cryptographic Hash Algorithm (SHA-3) Family. Federal Register 27(212), 62212–62220 (November 2007), http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf
12. Ohtahara, C., Sasaki, Y., Shimoyama, T.: Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160. In: Lai, X., Yung, M., Lin, D. (eds.) Inscrypt 2010. LNCS, vol. 6584, pp. 169–186. Springer, Heidelberg (2011)
13. Pramstaller, N., Rechberger, C., Rijmen, V.: Exploiting Coding Theory for Collision Attacks on SHA-1. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 78–95. Springer, Heidelberg (2005)
14. Sasaki, Y., Wang, L.: 2-Dimension Sums: Distinguishers Beyond Three Rounds of RIPEMD-128 and RIPEMD-160. Cryptology ePrint Archive, Report 2012/049 (2012), http://eprint.iacr.org/
15. Sasaki, Y., Wang, L.: Distinguishers beyond Three Rounds of the RIPEMD-128/-160 Compression Functions. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 275–292. Springer, Heidelberg (2012)
16. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the Hash Functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
17. Wang, X., Yin, Y.L., Yu, H.: Finding Collisions in the Full SHA-1. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 17–36. Springer, Heidelberg (2005)
18. Wang, X., Yu, H.: How to Break MD5 and Other Hash Functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Differential Characteristics

Table 6. Starting characteristic for the attack on round 2-4 of RIPEMD-160

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	--------------------------------			
-4	--------------------------------			
-3	--------------------------------			
-2	--------------------------------			
-1	--------------------------------			
16	--------------------------------	--------------------------------	0	--------------------------------
17	--------------------------------	--------------------------------	1	--------------------------------
18	--------------------------------	????????????????????????????????	2	--------------------------------
19	--------------------------------	????????????????????????????????	3	????????????????????????????????
20	--------------------------------	????????????????????????????????	4	--------------------------------
21	--------------------------------	????????????????????????????????	5	--------------------------------
22	--------------------------------	????????????????????????????????	6	--------------------------------
23	????????????????????????????????	????????????????????????????????	7	--------------------------------
24	????????????????????????????????	????????????????????????????????	8	--------------------------------
25	????????????????????????????????	????????????????????????????????	9	--------------------------------
26	????????????????????????????????	????????????????????????????????	10	--------------------------------
27	????????????????????????????????	????????????????????????????????	11	--------------------------------
28	--------------------------------	????????????????????????????????	12	--------------------------------
29	--------------------------------	????????????????????????????????	13	--------------------------------
30	--------------------------------	????????????????????????????????	14	--------------------------------
31	--------------------------------	--------------------------------	15	--------------------------------
32	--------------------------------	--------------------------------		
33	--------------------------------	--------------------------------		
34	--------------------------------	--------------------------------		
35	--------------------------------	--------------------------------		
36	--------------------------------	--------------------------------		
37	--------------------------------	--------------------------------		
38	--------------------------------	--------------------------------		
39	--------------------------------	--------------------------------		
40	--------------------------------	--------------------------------		
41	--------------------------------	--------------------------------		
42	--------------------------------	--------------------------------		
43	--------------------------------	--------------------------------		
44	--------------------------------	--------------------------------		
45	--------------------------------	--------------------------------		
46	--------------------------------	--------------------------------		
47	--------------------------------	--------------------------------		
48	--------------------------------	--------------------------------		
49	--------------------------------	--------------------------------		
50	--------------------------------	--------------------------------		
51	--------------------------------	--------------------------------		
52	--------------------------------	????????????????????????????????		
53	--------------------------------	????????????????????????????????		
54	--------------------------------	????????????????????????????????		
55	--------------------------------	????????????????????????????????		
56	--------------------------------	????????????????????????????????		
57	????????????????????????????????	????????????????????????????????		
58	????????????????????????????????	????????????????????????????????		
59	????????????????????????????????	????????????????????????????????		
60	????????????????????????????????	????????????????????????????????		
61	????????????????????????????????	????????????????????????????????		
62	????????????????????????????????	????????????????????????????????		
63	????????????????????????????????	????????????????????????????????		
	????????????????????????????????			
	????????????????????????????????			
	????????????????????????????????			
	????????????????????????????????			
	????????????????????????????????			

Table 7. Characteristic for the attack on round 2-4 of RIPEMD-160 once the differential characteristic after step 24 in the right stream has been fixed

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	--------------------------------			
-4	--------------------------------			
-3	--------------------------------			
-2	--------------------------------			
-1	--------------------------------			
16	--------------------------------	--------------------------------	0	--------------------------------
17	--------------------------------	--------------------------------	1	--------------------------------
18	--------------------------------	????????????????????????????????	2	--------------------------------
19	--------------------------------	????????????????????????????????	3	-------------------n------------
20	--------------------------------	????????????????????????????????	4	--------------------------------
21	--------------------------------	????????????????????????????????	5	--------------------------------
22	--------------------------------	????????????????????????????????	6	--------------------------------
23	????????????????????????????????	????????????????????????????????	7	--------------------------------
24	????????????????????????????????	-1--nu-u----n------n------------	8	--------------------------------
25	????????????????????????????????	-1-un-0----------0----0------u--	9	--------------------------------
26	?????????x??????????????????????	------0u---------1----1n-0--00--	10	--------------------------------
27	----------???????????????????x--	-------n----u---------0--1--n0--	11	--------------------------------
28	--------------------------------	-------0----------un--1------u--	12	--------------------------------
29	--------------------------------	-------00---n-----1----------1--	13	--------------------------------
30	--------------------------------	--1-----1---0----------------u--	14	--------------------------------
31	--------------------------------	--0--------------------------0--	15	--------------------------------
32	--------------------------------	--------------------------------		
33	--------------------------------	--------------------------------		
34	--------------------------------	--------------------------------		
35	--------------------------------	--------------------------------		
36	--------------------------------	--------------------------------		
37	--------------------------------	--------------------------------		
38	--------------------------------	--------------------------------		
39	--------------------------------	--------------------------------		
40	--------------------------------	--------------------------------		
41	--------------------------------	--------------------------------		
42	--------------------------------	--------------------------------		
43	--------------------------------	--------------------------------		
44	--------------------------------	--------------------------------		
45	--------------------------------	--------------------------------		
46	--------------------------------	--------------------------------		
47	--------------------------------	--------------------------------		
48	--------------------------------	--------------------------------		
49	--------------------------------	--------------------------------		
50	--------------------------------	--------------------------------		
51	--------------------------------	--------------------------------		
52	--------------------------------	-----x--------------------------		
53	--------------------------------	--------------------------------		
54	--------------------------------	--------------------------------		
55	--------------------------------	--------------------------------		
56	--------------------------------	---------------------------x----		
57	-----x--------------------------	--------------------------------		
58	--------------------------------	--------------------------------		
59	--------------------------------	--------------------------------		
60	--------------------------------	-----------------x--------------		
61	---------------------------x----	--------------------------------		
62	----------------------x---------	--------------------------------		
63	--------------------------------	--------------------------------		
	----------------------x---------			

	-------x---------x--------------			

Table 8. Differential Characteristic for round 2-4 of RIPEMD-160 resulting in a semi-free-start near-collision

i	∇B_i	$\nabla B'_i$	j	∇W_j
-5	--------------------------------			
-4	--------------------------------			
-3	--------------------------------			
-2	--------1000000-----------------			
-1	--------------------------------			
16	--------------------------------	00000000000000000000000000000000	0	--------------------------------
17	--------------------------------	00001010111111-11111000100011111	1	--------------------------------
18	--------------------------------	uuuu1uuuuuuuuuuuuuuuuuuuuuuuuuuu	2	--------------------------------
19	--------------------------------	01-01101001110101001100011-100--	3	-------------------n------------
20	---000--------------------------	nu-nu01000uuuu0u0u01101011-unn--	4	--------------------------------
21	-------------1------------------	11-000u0uu10--u1-111-11--10-00nn	5	-----11111--000-------------01--
22	---0----------------------------	uu-0nu01------11-11n110--n1-10uu	6	--------------------------------
23	---nu---------------------------	----1-01---u-0-1-0n01111-1uuuu10	7	--------------------------------
24	---00-------nuuuuuuuuuuuu000u---	-1--nu0u0---n011-00n0000--01----	8	-------------------1------------
25	000-----10--111111111111111-0000	-10un-0-00-------011000---101u-0	9	--------------------------------
26	--101101nu111111--1111111-------	--1---0u0111-----100--1n-00100--	10	----0101--------------------11--
27	--------00----unnnnnnnnnnnnnnn--	-------n11--u--1---1--0--10-n0--	11	--------------------------------
28	--------------100010000000000011	-------0----------un--1------u--	12	--------------------------------
29	----0000000001111111------------	-------00---n----11----------1--	13	----11110-----------------------
30	-------------------------------0	--1-----1--------------------u--	14	-------------------------1---10-
31	--------------------------------	--0--------------------------0--	15	-------------------------00-----
32	--------------------------------	--------------------------------		
33	--------------------------------	--------------------------------		
34	--------------------------------	--------------------------------		
35	--------------------------------	--------------------------------		
36	--------------------------------	--------------------------------		
37	--------------------------------	--------------------------------		
38	--------------------------------	--------------------------------		
39	--------------------------------	--------------------------------		
40	--------------------------------	--------------------------------		
41	--------------------------------	--------------------------------		
42	--------------------------------	--------------------------------		
43	--------------------------------	--------------------------------		
44	--------------------------------	--------------------------------		
45	--------------------------------	--------------------------------		
46	--------------------------------	--------------------------------		
47	--------------------------------	--------------------------------		
48	--------------------------------	--------------------------------		
49	--------------------------------	--------------------------------		
50	--------------------------------	--------------------------------		
51	--------------------------------	--------------------------------		
52	--------------------------------	-----x--------------------------		
53	--------------------------------	--------------------------------		
54	--------------------------------	--------------------------------		
55	--------------------------------	--------------------------------		
56	--------------------------------	---------------------------x----		
57	-----x--------------------------	--------------------------------		
58	--------------------------------	--------------------------------		
59	--------------------------------	--------------------------------		
60	--------------------------------	-----------------x--------------		
61	---------------------------x----	--------------------------------		
62	----------------------x---------	--------------------------------		
63	--------------------------------	--------------------------------		
	----------------------x---------			

	-------x---------x--------------			

Revisiting Difficulty Notions for Client Puzzles and DoS Resilience

Bogdan Groza[1] and Bogdan Warinschi[2]

[1] Faculty of Automatics and Computers, Politehnica University of Timisoara
bogdan.groza@aut.upt.ro
[2] Computer Science Department, University of Bristol
bogdan@cs.bris.ac.uk

Abstract. Cryptographic puzzles are moderately difficult problems that can be solved by investing non-trivial amounts of computation and/or storage. Devising models for cryptographic puzzles has only recently started to receive attention from the cryptographic community as a first step towards rigorous models and proofs of security of applications that employ them (e.g. Denial-of-service (DoS) resistance). Unfortunately, the subtle interaction between the complex scenarios for which cryptographic puzzles are intended and typical difficulties associated with defying concrete security easily leads to flaws in definitions and proofs. Indeed, as a first contribution we exhibit shortcomings of the state-of-the-art definition of security of cryptographic puzzles and point out some flaws in existing security proofs. The main contribution of this paper are new security definitions for puzzle difficulty. We distinguish and formalize two distinct flavors of puzzle security (which we call optimal and ideal) and in addition properly define the relation between solving one puzzle vs. solving multiple ones. We demonstrate the applicability of our notions by analyzing the security of two popular puzzle constructions. In addition, we briefly investigate existing definitions for the related notion of DoS security. We demonstrate that the only rigorous security notions proposed to date is not sufficiently demanding (as it allows to prove secure protocols that are clearly not DoS resilient) and suggest an alternative definition. Our results are not only of theoretical interest. We show that our better characterization of hardness for puzzles and DoS resilience allows establishing formal bounds on the effectiveness of client puzzles which confirm previous empirical observations.

1 Introduction

Background. Cryptographic puzzles are moderately difficult problems that can be solved by investing non-trivial amounts of computation and/or memory. A typical use for puzzles is to balance participants costs during the execution of some protocols. For examples, many papers addressed their use against resource depletion in SSL/TLS [7], TCP/IP [14], general authentication protocols [3,10], spam combat [9], [8], [11]. The use of puzzles reaches beyond balancing resources: they can be used as proof-of-work in other applications (like timestamping) or through a clever application in encryption into the future [17]. Puzzles are accounted under various names: cryptographic puzzles, client puzzles, computational puzzles or proofs of work, we prefer the first one since the puzzles that we study are intrinsically based on cryptographic functions.

Most of the puzzle-related literature concentrates on providing constructions, often with additional, innovative properties. For example puzzles that are non-parallelizable prevent an adversary from using distributed computations to solve them. Examples of constructions include the well known time-lock puzzle [17], the constructions proposed by Tritilanunt et al. in [21] and later by Jeckmans [12], Ghassan and Čapkun [15], Tang and Jeckmans [20], Jerschow and Mauve [13]. All of these constructions can ensure that a puzzle-solver spends computation cycles before a server engages in any expensive computation. To alleviate computational disparities between solvers, Abadi et al. [1] build puzzles that rely on memory usage rather than on CPU speed, this leading to a more uniform behaviour between devices.

Given the wide-range of applications for puzzles and the number of proposed constructions it is probably surprising that devising formal security notions for puzzles has received rather little attention so far, with only two notable exceptions. Chen et al. [6] initiate the formal study of security properties for puzzles. They identify two such properties. *Puzzle difficulty* requires that no adversary can solve *a single puzzle* faster than some prescribed bound, whereas *puzzle unforgeability* requires that no adversary can produce a valid-looking puzzle. While this latter property is not required by all scenario usages for puzzles, the former one is critical. In a recent paper, Stebila et al. [18] notice that single-puzzle difficulty may not suffice to guarantee security when puzzles are used in real applications, since here it may be needed that an adversary does not solve *multiple puzzles* faster than some desired bound, and the relation between single-puzzle difficulty and multi-puzzle difficulty is unclear at best, and completely inexistent at worst.

To fix this, Stebila et al. [18] propose a notion of puzzle difficulty that accounts for multiple puzzles being solved at once and prove that two existing constructions HashInversion (initially used by Juels and Brainard [14]) and HashTrail (initially used in the hashcash system [4]) meet this notion. The main motivation for the work in this paper is that the proposed security definition is problematic: the notion defined is incomplete since it does not account for the tightness of the bounds and, strictly speaking, it cannot be met by any existing scheme. This does not contradict the security proofs mentioned above as the claims rely on faulty analysis: the difficulty bound provided for the HashInversion puzzle is wrong while for HashTrail is largely overestimated.

Our results. The main contribution of our paper are new security notions for puzzle difficulty. We distinguish between two different flavors of puzzle difficulty. The first property demands that no adversary can solve the puzzle faster than by using the "prescribed" algorithm (i.e. the puzzle-solving algorithm that is associated to the puzzle). We call such puzzles *optimal*. We call a puzzle *ideal* if on the average the puzzle is as hard to solve as in the worst case. These notions have already appeared in the literature but have never been formalized and previous work does not seem to make a clear distinction between them. For example, [2] introduces informally the notion of *computation guarantee* which requires that a malicious party cannot solve the puzzle significantly faster than honest clients. This is what we call optimality. Other papers [20] require that solving the puzzle be done via deterministic computation – this seems to be what we call an *ideal* puzzle. The formulations for both of these notions are in the multi-puzzle setting which, as correctly observed in [18], is the case relevant for most practical applications. While it is not true in general that for a puzzle construction solving n puzzles takes n times the resources needed for solving one puzzle, this is clearly a desirable property. We capture this intuition through a property that we call *difficulty preserving*. Having fixed the definitions we move to the analysis of two popular puzzle systems HashTrail and HashInversion. We prove that, in the random oracle model, these puzzles are optimal and difficulty preserving for concrete difficulty bounds that we derive. Finally, we look at existing work on using puzzles for provable DoS resistance. Unfortunately, we discovered that the formal definition for DoS resilience proposed by [18] is not strong enough as it allows for clear attacks against protocols that are provably secure according to the definition. We then design and justify a new security definition that does not suffer from the problems that we have identified.

Before we move on, we note that getting the security definitions for puzzles and DoS security right is quite important as more and more works in this direction have appeared (a book chapter in [5] and also [16] and [19]) and all seem to have inherited the weaknesses in the definition of [18].

2 Shortcomings of Existing Definitions and Proofs

The first attempt to formalize puzzle properties, and in particular puzzle difficulty, was by Chen et al. in [6]. Recently, Stebila et al. [18], motivated by the observation that the security notion of [6] does not guarantee that solving n puzzles is n times harder than solving one, introduced a new definition of puzzle difficulty. In brief, a puzzle is deemed $\epsilon_{k,d,n}(\cdot)$-strongly difficult if the success probability of an adversary is less or equal to $\epsilon_{k,d,n}(\cdot)$ and $\epsilon_{k,d,n}(t) \leq \epsilon_{k,d,1}(t/n)$ (this later condition enforcing stronger difficulty w.r.t. n puzzles). Here k is a security parameter, d is the difficulty level and n denotes the number of solved puzzles.

Shortcomings of existing definitions. There are several weak points in the difficulty definition outlined above. Perhaps the most problematic one is that the property of a puzzle of being "strongly" difficult is in fact a property of the function ϵ that upperbounds the success of the adversary. However, ϵ is an upper bound on the hardness of the puzzle, but not necessarily the tightest possible (for example if one sets $\epsilon_{k,d,n} = 1$ any puzzle is $\epsilon_{k,d,n}$-strongly difficult). A natural question is then what if one can find a bound that deems the puzzle strongly difficult, while for some other tighter bounds this property does not hold anymore. Should we consider such a puzzle strongly difficult or not? Note that in contrast, Chen et al. in [6] clearly state that any puzzle that is ϵ difficult is $\epsilon + \mu$ difficult and the most accurate difficulty bound is the *infimum* of ϵ. The point is not that one would find such a bound on purpose, but rather as security reductions are not trivial one could find a good bound with respect to which the puzzle is strongly difficult, just to turn out that the puzzle is not strongly difficult for a tighter bound.

To show that the tightness of the bound matters, take for example the case of the time-lock puzzles. We skip the formalism as we want to keep this example as intuitive as possible. Set m to be an RSA-like modulus (sufficiently large to rule out any insecurity) and assume that solving one puzzle means given $x \in_R [0..2^{k-1}]$ to compute $x^{2^d} \bmod m$. We assume the usual hypothesis that this computation cannot be done faster than d squarings unless one knows the factorization of the modulus. Suppose the adversary can get 1 or 2 fresh values x and has to compute $x^{2^d} \bmod m$ for each of them with no prior knowledge of the modulus. We can say that the success probability of the adversary is upper bounded by $\epsilon_{k,d,n}(t) = \frac{t}{n \cdot d}, \forall n \in \{1, 2\}$. To check for correctness, indeed, if $n = 1$ the probability to find the output for less than d steps (one step means one squaring) is almost 0 assuming a sufficiently large modulus and 1 at d steps. While for $n = 2$, for less than d steps the probability is 0, at d steps the adversary has solved the first puzzle, while the probability that the second is also solved is 2^{-k} due to the possibility of colliding x_1, x_2, and 2^{-k} is lower than $1/2$ claimed by the upper bound. Thus the bound holds and one can also verify that $\epsilon_{k,d,1}(t/2) = \epsilon_{k,d,2}(t)$ so the puzzle is $\epsilon_{k,d,n}(t)$-strongly difficult. We set some artificially small parameters just to easily exhibit some calculation. Let $k = 16$ and $d = 2^{16}$ (the bound holds for these values as well). One would expect that solving the two puzzles requires $2 \times 2^{16} = 131072$ steps. However, due to the possibility of colliding inputs the average number of steps is actually $2^{16} - 1 = 131071$, that is, one step is missing. The numbers given here are artificially small and the variation is not very relevant, but it has the sole purpose to show that the criterion has some deficiencies. The problem here is that the bound is not tight enough. More precise bounds that should have been used are: $\epsilon_{k,d,1} = 0$ if $t \in [0, d)$, $\epsilon_{k,d,1} = 1$ if $t = d$ and $\epsilon_{k,d,2} = 0$ if $t \in [0, d)$, $\epsilon_{k,d,2} = 2^{-k}$ if $t \in [d, 2d)$ and $\epsilon_{k,d,2} = 1$ if $t = d$. For these bounds indeed $\epsilon_{k,d,1}(t/2) \leq \epsilon_{k,d,2}(t)$ which shows that in fact the puzzle is not strongly difficult. These bounds are also informal and we used them just as an intuition, indeed for any $t < d$ the adversary can still guess the solution with negligible (but non-zero) probability.

We can prove, and we specify this in a remark that follows, that if the bound is tight then the condition from [18] is sufficient to make a puzzle difficulty preserving. But, one may further ask if this condition is really necessary. The answer is negative. In fact, quite surprisingly, the HashTrail puzzle does not satisfy it and neither does the HashInversion puzzle (while both of them can be proved to be difficulty preserving). We call HashInversion the generic puzzle which consists in the partial inversion of a hash function, that is given $x'', H(x'||x'')$ find x'. Also, we refer HashTrail as the generic puzzle which consists in finding an input to $H(r||\cdot)$ such that the result has a fixed number of trailing zeros. Both these constructions are frequently used in many proposals. The first one is used by Jules and Brainard in [14] and the second by Back in the Hashcash system [4]. We prefer the generic names HashInversion and HashTrail as these suggest better what means to solve the puzzle as well as we are not interested in the specific details for the construction of the puzzles used in [4], [14].

Moreover, and this is another weakness for the definition of [18], the criterion $\epsilon_{k,d,n}(t) \leq \epsilon_{k,d,1}(t/n)$, can never hold in general. The reason is that in the game that defines security of multiple puzzle it is possible with some (negligible) probability that the challenge puzzles contain *two identical* puzzles. In this case solving n puzzles should *always* require less effort than n times the effort required to solve a single puzzle, at least up to negligible factors. The definition should therefore allow for this kind of slack, i.e. it should require that $|\epsilon_{k,d,n}(t) - \epsilon_{k,d,1}(t/n)| \leq k^{-\omega(1)}$. The time-lock puzzle seems to satisfy such a criterion, but note that this is certainly not the case for the hash-based puzzles above which are the most commonly employed solution in practice.

Flaws in existing proofs. In light of the above comments, it is natural to ask how tight are the bounds obtained in [18]. By inspecting the security proofs it turns out that beside the conceptual shortcoming in judging the hardness of n puzzle instances, the bound used for the HashTrail puzzle is extremely loose while the bound for the HashInversion puzzle is wrong (these puzzles are difficulty preserving as we show later in the paper, but unfortunately the proofs provided in [18] are wrong). Figure 1 depicts the loose bound in (i) and the wrong bound in (ii) for the case of $n = 3$ puzzles of difficulty $d = 8$ bits. Note that in (ii) the adversary advantage is well underestimated.

We give a short numerical example to illustrate this. Informally, the HashInversion puzzle requires that given $H(x'||x''), x''$ find random $x' \in_R [0..2^d)$. The difficulty bound claimed for this puzzle is $\epsilon_{d,k,n} = (\frac{q+n}{n \cdot 2^d})^n$ and the puzzle is deemed strongly difficult with respect to this bound. Just to show that this bound is wrong consider the trivial case of $n = 2$, $d = 3$, i.e., the case of solving 2 puzzles each having 3 bits. Consider an adversary running at most 11 steps. According to the aforementioned bound, one would expect that the advantage of the adversary is less than $(\frac{11+2}{2 \cdot 2^3})^2 = (\frac{13}{16})^2 \approx 0.66$. Consider the naive (yet the best) algorithm that successively walks trough the set $\{0, 1, 2, ..., 7\}$ in order to solve each puzzle. The success probability of this algorithm is actually bigger than 0.66 as one can easily show. The naive algorithm can solve two puzzles in 11 steps if, given x'_1 and x'_2 the two solutions, it holds that $x'_1 + x'_2 \leq 9$. That is, there exists 1 solutions for 2 steps (the pair $\{(0, 0)\}$), 2 solutions for 3 steps (the pairs $\{(0, 1), (1, 0)\}$) and so on, $k - 1$ solutions for k steps up to $k = 9$ steps. From there on, one can note that for 10 steps given the set of pairs $\{(0, 8), (1, 7), (2, 6), (3, 5), (4, 4), (5, 3), (6, 2), (7, 1), (8, 0)\}$ one must discard the first and the last pair (since 8 is not a valid value for the 3 bit guess) while for 11 steps one must discard the first 2 and the last 2 pairs. Summing up, the naive algorithm succeeded in $1+2+3+...+8+7+6 = 36+13 = 49$ out of the obvious $2^3 \times 2^3$ variants which gives a success probability of $\frac{49}{64} \approx 0.76$. Thus the naive algorithm does better than the success probability of the adversary considered in [18] and the discrepancy is due to the flawed security proof. The difference is not big in this example, but obviously it gets significant when one increases the values of n and d.

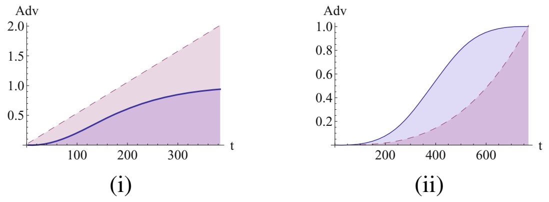


Fig. 1. Adversary advantage at $n = 3$, $d = 8$ for HashTrail (i) and HashInversion (ii) puzzles according to Stebila et al. (dotted line) and in this paper (continuous line)

3 Puzzle Difficulty

To formalize puzzle difficulty and related notions we proceed as follows. First we define the usual game of solving multiple puzzles and bound the adversary advantage by $\epsilon_{k,d,n}(t)$. Then, we define puzzle optimality which means that, up to some negligible factor, there is no adversary that can solve one or more puzzles with better advantage than the solving algorithm that comes with the puzzle. This property was generally ignored, we consider it to be the most relevant, since if an adversary can solve puzzles in less steps than the puzzle solving algorithm, then such a construction may have no use at all. Further, if the puzzle is optimal, assuming the usual way of solving more puzzles by running the solving algorithm on each of the puzzles, the puzzle is difficulty preserving and solving n puzzles is n times as hard as solving one. For completeness, we also define ideal puzzles as puzzles that can not be solved faster than the average number of steps, this again up to some negligible factor.

3.1 Syntax for Cryptographic Puzzles

Our definition of a puzzle system follows in spirit the definition from [6], with several differences. One is that we do not consider arbitrary strings as inputs together with keys to the puzzle generation, but instead we group these in what we call the attribute space. This ensures a more general setting, since strings and long term secrets are part of puzzles that assure additional properties, such as unforgeability, etc. Thus in the simpler case where one does not want to ensure any additional property, the attributes can be set to null. We use the symbol $\perp$ to indicate the null attribute. The attributes can also be used to simulate secret keys, if these are used in the construction of the protocol.

Definition 1 (Cryptographic puzzle). *Let* dSpace *denote the space of difficulty levels,* pSpace *the puzzle space,* sSpace *the solution space and* aSpace *the attribute space. A cryptographic puzzle, or alternatively client puzzle,* CPuz *is a quadruple of algorithms,* $(\mathsf{Setup}, \mathsf{Gen}, \mathsf{Find}, \mathsf{Ver})$*, with the following descriptions:*

- $\mathsf{Setup}(1^k)$ *is the setup algorithm that takes as input a security parameter* 1^k *and outputs* dSpace, pSpace, sSpace *and* aSpace,
- $\mathsf{Gen}(d, \mathsf{atr})$ *is the puzzle generation algorithm, it takes as input the difficulty of the puzzle to be created* $d \in \mathsf{dSpace}$ *and a list of attributes* $\mathsf{atr} \in \mathsf{aSpace}$ *then outputs a puzzle instance* $\mathsf{puz} \in \mathsf{pSpace}$,
- $\mathsf{Find}(\mathsf{puz}, t)$ *is the solving algorithm that takes as input a puzzle* $\mathsf{puz} \in \mathsf{pSpace}$ *and the maximum number of steps* t *that is allowed to perform, then outputs a solution* $\mathsf{sol} \in \mathsf{sSpace} \cup \{\perp\}$ *(where* $\perp$ *is for the case when a solution could not be found in* t *steps),*
- $\mathsf{Ver}(\mathsf{puz}, \mathsf{sol}')$ *is the verification algorithm that takes as input a potential solution* $\mathsf{sol}' \in \mathsf{sSpace}$ *and a puzzle* $\mathsf{puz} \in \mathsf{pSpace}$ *and outputs* 1 *if and only if* sol' *is a correct solution for puzzle* puz *and* 0 *otherwise.*

For soundness, we require that puz is the input necessary and sufficient to run the Find algorithm and that for any sol that is output of Find the verification holds. By this, we force that one cannot produce a puzzle construction that is impossible to solve either because the information is not sufficient or the puzzle has no solution.

Remark 1. In [18] Ver also takes the secret master key s and is also responsible for verifying if the solution is authentic, also embedding the functionality of VerAuth from [6]. Here we choose to keep the puzzle description close to that from [6], thus Ver is responsible just for verifying the correctness of the solution, and not the authenticity of the puzzle.

Remark 2. The puzzle is generic and can be further augmented with other algorithms to ensure additional properties. For example one can add the Auth algorithm to verify authenticity for the case of unforgeable puzzles as in [6], etc.

Remark 3. On purpose, we did not specify any detail on the runtime of Gen, Find and Ver algorithms. This is because we wanted to keep the definition as generic as possible as it addresses puzzle in general. For practical purposes, one can request that all four algorithms work in probabilistic polynomial time.

3.2 Optimal, Ideal, and Difficulty Preserving Puzzles

We formalize puzzle difficulty using a game in which the adversary $\mathcal{A}$ is allowed to get as many puzzles and their solutions from the challenger $\mathcal{C}$ and later needs to find solutions for one or more puzzles generated by the challenger.

PUZZLE SOLVING GAME. We define the puzzle game $\mathsf{Exec}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(t)$ as the following four stage game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$:

1. challenger $\mathcal{C}$ runs Setup on input 1^k to get dSpace, pSpace, sSpace, aSpace and sets $d \in$ dSpace as the difficulty level of the game,
2. adversary $\mathcal{A}$ is allowed to make q_{Gen} queries to GenSolvePuz which returns each time a puzzle and its corresponding solution, i.e., $\{\mathsf{puz}, \mathsf{sol}\}$, and n queries to a Test oracle which on each invocation generates and returns a target puzzle $\mathsf{puz}^{\diamondsuit}$,
3. after t steps adversary $\mathcal{A}$ outputs the solutions $\{\mathsf{sol}^{\diamondsuit}_1, \mathsf{sol}^{\diamondsuit}_2, ..., \mathsf{sol}^{\diamondsuit}_n\}$ for puzzles $\{\mathsf{puz}^{\diamondsuit}_1, \mathsf{puz}^{\diamondsuit}_2, ..., \mathsf{puz}^{\diamondsuit}_n\}$ that were returned by Test,
4. challenger $\mathcal{C}$ queries Ver on all puzzles and solutions output from adversary $\mathcal{A}$ and returns 1 if all solutions are correct else returns 0.

The *winning* event, denoted by $\mathsf{Win}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(t)$, is the event in which the adversary outputs a correct solution for the puzzles and the game returns 1, i.e.,

$$\mathsf{Win}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(q_{\mathsf{Gen}}, t) = \Pr\left[\mathsf{Exec}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(q_{\mathsf{Gen}}, t) = 1\right]$$

Remark 4. We did not stress whether the adversary $\mathcal{A}$ runs GenSolvePuz on its own or these are simulated by the challenger $\mathcal{C}$ as we did not make distinction between interactive and non-interactive puzzles (puzzles that are generated by the solver or the challenger). We defer such specific details for the security proof of each particular puzzle that we analyze.

Remark 5. In addition to previous hardness definitions we allow collision in the generation algorithm, that is, we did not exclude that the same puzzle can be outputted more than once. Generally, collisions appear as a negligible factor in the hardness bound, but this factor is relevant as the examples from the previous section showed.

Remark 6. In $\mathsf{Exec}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(t)$ we assumed puzzles of the same difficulty level. But it is easy to extend this definition to puzzles of various difficulty levels as well. Such an extension to puzzles of multiple difficulty levels does not appear to be possible with the definition from related work [18] since multiple puzzle difficulty is linked inextricably to single puzzle difficulty, but for precisely the same difficulty parameter d.

Definition 2 (Difficulty bound). *For $\epsilon_{k,d} : \mathrm{N} \rightarrow [0,1]$ a family of functions indexed by parameters k, d and n, we say that $\epsilon_{k,d}(\cdot)$ is a difficulty bound for puzzle system* CPuz *if:* $\mathsf{Win}^{\mathsf{CPuz},k}_{\mathcal{A},d,n}(q_{\mathsf{Gen}}, t) \leq \epsilon_{k,d,n}(q_{\mathsf{Gen}}, t)$.

Before formally defining the different properties for puzzles, we need to introduce the average and the maximum solving time that one should expect from an honest client, that is a client who simply executes the Find algorithm that defines the puzzle. Below, we write $\mathsf{Exec}^{\mathsf{CPuz},k}_{\mathsf{Find},d,n}(t)$ for the random variable obtained by executing the experiment defined above with a "benign" adversary who for each puzzle that it obtains as challenge it solves it using the Find algorithm. The following definitions captures the average probability of solving n puzzles of difficulty d in time t.

Definition 3 (Find bound). *For a given* CPuz *we denote by $\zeta_{k,d,n}(t)$ the probability that* Find *correctly finishes in at most t steps, i.e., $\zeta_{k,d,n}(t) = \Pr\left[\mathsf{Exec}^{\mathsf{CPuz},k}_{\mathsf{Find},d,n}(t) = 1\right]$.*

For a puzzle system, the next definition identifies the maximum number of steps needed by the Find algorithm to solve n puzzles with probability 1.

Definition 4 (Maximum solving time). *For a given* CPuz *the maximum solving time of* CPuz *is $\mathrm{t}_{\max}$ if $\mathrm{t}_{\max}$ is the minimum number of steps at which $\zeta_{k,d,n}(t)$ is 1, i.e., $\zeta_{k,d,n}(\mathrm{t}_{\max}) = 1, \zeta_{k,d,n}(\mathrm{t}'_{\max}) < 1, \forall \mathrm{t}'_{\max} < \mathrm{t}_{\max}$.*

Definition 5 (Average solving time). *For a given* CPuz *we define the average solving time as the average number of steps required by* Find*, i.e., $\mathrm{t}_{\mathrm{avr}}(k,d,n) = \sum_{i=1,\mathrm{t}_{\max}} i \cdot [\zeta_{k,d,n}(i) - \zeta_{k,d,n}(i-1)]$.*

Remark 7. In Definition 1 we assumed that Find can solve at most one puzzle at a time, thus whenever Find is used to solve more than one puzzle we consider the usual way in which one repeatedly uses Find for each of the puzzles. In the case when Find behaves differently on more than one puzzle, one can extend the input and output of Find to a vector of puzzle instances and solutions.

Remark 8. There are puzzles for which $\mathrm{t}_{\max}$ is infinite while $\mathrm{t}_{\mathrm{avr}}$ is finite. Consider for example the trivial HashTrail puzzle, which we analyze in the next section, that consists in finding an input for a hash function such that the output ends with d consecutive zeros. Obviously, if one considers the hash function simulated by a random oracle, we have $\mathrm{t}_{\max} = \infty$ and $\mathrm{t}_{\mathrm{avr}} = 2^d$.

Definition 6 (Optimal puzzle). *We say that* CPuz *is optimal if at any number of steps and any number of puzzles the success probability of the adversary is upper bounded by the success probability of the solving algorithm of the puzzle plus some negligible factor in the difficulty level and security parameter, i.e., $\forall t, n, \epsilon_{k,d,n}(t) \leq \zeta_{k,d,n}(t) + \nu_n(k, d)$.*

Definition 7 (Difficulty preserving puzzle). *We say that an optimal* CPuz *is difficulty preserving if the average solving time for* n *puzzles of difficulty* d *equals* n *times the average solving time of a puzzle of difficulty 1 up to some negligible factor, i.e.,* $\forall n, d, |\mathrm{t_{avr}}(k, d, n) - n \cdot \mathrm{t_{avr}}(k, d, 1)| \leq \nu_n(k, d)$.

Remark 9. The optimality condition $\epsilon_{k,d,n}(t) \leq \zeta_{k,d,n}(t) + \nu_n(k, d)$ ensures that the bound from the puzzle solving game, i.e., $\epsilon_{k,d,n}(t)$, is $\nu_n(k, d)$ tight.

Remark 10. The condition $\epsilon_{k,d,n}(t) \leq \epsilon_{k,d,1}(t/n)$ is enough to assure that an optimal puzzle, i.e., a puzzle for which $\forall n, d, |\epsilon_{k,d,n}(t) - \zeta_{k,d,n}(t)| \leq \nu_n(k, d)$, is difficulty preserving. This is trivial to prove, but it seems that the condition $\epsilon_{k,d,n}(t) \leq \epsilon_{k,d,1}(t/n)$ is not so trivial since none of the puzzles that we analyze next satisfies it (one could easily plot the difficulty bounds to verify this).

Remark 11. To assure that a puzzle is difficulty preserving for puzzles of various difficulty levels, one must enforce that $\mathrm{t_{avr}}(k, \overline{d}, n)$ is the sum of the difficulty levels, i.e., $\mathrm{t_{avr}}(k, \overline{d}, n) = \mathrm{t_{avr}}(k, d_1, 1) + \mathrm{t_{avr}}(k, d_2, 1) + ... + \mathrm{t_{avr}}(k, d_n, 1)$. Here $\overline{d}$ denotes an array of the difficulty levels.

Definition 8 (Ideal puzzle). *We say that an optimal puzzle* CPuz *is ideal if the average solving time equals the maximum solving time up to some negligible value in the the security parameter* k *and difficulty level* d*, i.e.,* $\forall n, d, |\mathrm{t_{avr}}(k, d, n) - \mathrm{t_{max}}(k, d, n)| \leq \nu_n(k, d)$*. Alternatively, having an optimal puzzle, i.e.,* $\epsilon_{k,d,n}(t) \leq \zeta_{k,d,n}(t) + \nu_n(k, d)$*,* $\forall n, d$*, the puzzle is ideal if* $\epsilon_{k,d,n}(t) = \nu_n(k, d), \forall n, d, \forall t < \mathrm{t_{max}}$.

4 New Difficulty Bounds for HashTrail and HashInversion

We now examine the HashInversion and HashTrail puzzles and establish tight security bounds for each of them.

HASHTRAIL PUZZLE. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a publicly known hash function. The $\mathsf{HashTrail}$ puzzle is a quadruple of algorithms:

- $\mathsf{Setup}(1^k)$ is the setup algorithm that on input 1^k outputs $\mathsf{dSpace} = [1, k]$, $\mathsf{pSpace} = \{0,1\}^* \times \{0,1\}^k$, $\mathsf{sSpace} = \{0,1\}^*$,
- $\mathsf{Gen}(d)$ is the generation algorithm which on input d randomly chooses $r \in \{0,1\}^k$ and outputs puzzle instance $\mathsf{puz} = \{d, r\}$,
- $\mathsf{Find}(\mathsf{puz}, t)$ is the solving algorithm that on input puz and the number of steps t iteratively samples $\mathsf{sol} \in [0, t)$ until $\mathcal{H}(r||\mathsf{sol})_{1..d} = 0$,
- $\mathsf{Ver}(\mathsf{puz}, \mathsf{sol})$ is the algorithm that takes $\mathsf{puz}, \mathsf{sol}$ as input and returns 1 if $\mathcal{H}(r||\mathsf{sol})_{1..d} = 0$ and 0 otherwise.

Theorem 1. *In the random oracle model, the HashTrail puzzle is optimal and difficulty preserving with* $\mathrm{t_{avr}}(k, d, 1) = 2^d$*,* $\mathrm{t_{max}}(k, d, 1) = \infty$ *and solving and difficulty bounds:* $\zeta^{HT}_{k,d,n}(t) = \sum_{i=n,t} \binom{i-1}{n-1} \cdot \frac{1}{2^{nd}} \cdot \left(1 - \frac{1}{2^d}\right)^{i-n}$, $\epsilon^{HT}_{k,d,n}(t) \leq \zeta^{HT}_{k,d,n}(t) + \frac{1}{2^d - 1} + \frac{q^2_{\mathsf{Gen}}}{2^{k+1}}$.

Remark 12. For HashTrail, in [18] the advantage is upper bounded by $\frac{q+n}{n \cdot 2^d}$ using Markov inequality - obviously, $\frac{q}{2^d}$ is a bound for the probability to solve 1 puzzle in q queries and dividing it with n gives a bound of the probability for n instances. While such a bound is easy to prove, Figure 1 shows how loose this is compared to the advantage from the previous theorem for a small numerical example. In section 2 we showed that loose bounds cannot say much about the difficulty of solving multiple puzzles.

HASHINVERSION PUZZLE. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a publicly known hash function. The $\mathsf{HashInv}$ puzzle is the quadruple of algorithms:

- $\mathsf{Setup}(1^k)$ is the setup algorithm that on input 1^k outputs $\mathsf{dSpace} = [1,k]$, $\mathsf{pSpace} = \{0,1\}^* \times \{0,1\}^k$, $\mathsf{sSpace} = \{0,1\}^*$,
- $\mathsf{Gen}(d)$ is the puzzle generation algorithm which on input d randomly chooses $x \in \{0,1\}^k$, computes $\mathcal{H}(x)$, sets x' as the first d bits of x and x'' as the remaining bits and outputs puzzle instance $\mathsf{puz} = \{d, x'', \mathcal{H}(x)\}$,
- $\mathsf{Find}(\mathsf{puz})$ is the solving algorithm that on input puz and the number of steps t iteratively samples at most t values $\mathsf{sol} \in \{0,1\}^d$ until $\mathcal{H}(\mathsf{sol}||x'') = \mathcal{H}(x)$,
- $\mathsf{Ver}(\mathsf{puz}, \mathsf{sol})$ be the algorithm that takes puz, sol as input and returns 1 if $\mathcal{H}(\mathsf{sol}||x'') = \mathcal{H}(x)$ and 0 otherwise.

Theorem 2. *Let $[z^i]P(z)$ denote the coefficient of z^i in the expansion of polynomial $P(z)$. In the random oracle model, the HashInversion puzzle is optimal and difficulty preserving with $\mathrm{t_{avr}}(k,d,1) = 2^{d-1}$, $\mathrm{t_{max}}(k,d,1) = 2^d$ and solving and difficulty bounds: $\zeta^{HI}_{k,d,n}(t) = \sum_{i=n,t}[z^i]\left(z \cdot \frac{1-z^{2^d}}{1-z}\right)^n \cdot \frac{1}{2^{nd}}$, $\epsilon^{HI}_{k,d,n}(t) \le \zeta^{HI}_{k,d,n}(t) + \frac{n}{2^d} + \frac{q^2_{\mathsf{Gen}}}{2^{k-d+1}}$.*

The proof of Theorem 1 can be found in Appendix A.1, due to space limitations we defer a proof of Theorem 2 for the extended version of this paper.

Remark 13. In [18] the advantage of HashInversion is upper bounded by $\left(\frac{q+n}{n \cdot 2^d}\right)^n$. As Figure 1 shows for a small numerical example, the advantage of the solving algorithm from the previous theorem is much bigger, thus the bound in [18] is wrong.

5 DoS Resilience

Defining resilience against resource exhaustion DoS is a non-trivial task that requires subtle analysis of the costs incurred by the steps done on the server side. In practice, choosing the right amount of work that needs to be done in order to gain access to a particular resource on the server side is a matter of protocol engineering, rather than cryptography. Notably, as the server resources are always limited, when the number of honest clients exceeds the total amount of resources, resource exhaustion is unavoidable. Thus from the protocol design, the best one could do is to hinder an adversary from claiming resources in the name of potentially many honest participants - this is were proof-of-work comes into action.

AN ATTACK ON THE APPROACH OF STEBILA ET AL. The security definition for DoS resilience of Stebila et al. [18] builds directly on the difficulty of puzzle systems of [18] and, essentially, requires that an adversary cannot claim more resources than the number of puzzles he is able to solve in the running time of the adversary. The problem with this definition is that it disregards an important aspect of puzzle defense against DoS, namely puzzle management. Puzzles used for DoS resilience come with an expiration time to avoid what we call a *next day attack* where an adversary first spends large amounts of resources to solve a large amount of puzzles and later it uses these puzzles with solutions to claim the corresponding resources in a much shorter interval. The definition of [18] allows for next day attacks as the execution that is considered looks directly at how many puzzles an adversary can solve in time t (and this amount is bounded by puzzle difficulty), but does not account for the possibility that the puzzles sent to claim resources could have been solved earlier.

OUR APPROACH. To prevent such attacks we take two measures: first we introduce a fixed lifetime for the puzzles, then we define resilience as a condition that must hold in any time interval $[t_2, t_1]$ and not just for an adversary having runtime t. By $\Pi_{t_{puz}}(\mathsf{CPuz})$ we denote a protocol Π that is protected by puzzles generated by CPuz and with lifetime t_{puz}, i.e., the protocol deems as invalid any solution received later than t_{puz} after the generation of the puzzle. We stress that we do not consider the detailed cost of running the server program, etc., and we consider as a premise that puzzles of difficulty d from CPuz are enough to protect the server.

PROTOCOL ATTACK GAME. We define the attack game $\mathsf{Exec}_{\mathbf{Adv}}^{\Pi_{t_{puz}}(\mathsf{CPuz}_{d,n}^k)}$ based on a two stage adversary. First adversary $\mathbf{Adv}_1$ is allowed to interact with the server and honest clients via: (1) $\mathsf{RequestPuz}(\mathsf{str})$ on which the server answers with a new fresh instance puz, (2) $\mathsf{SolvePuz}(\mathsf{puz})$ on which any client answers with a solution sol.

Then $\mathbf{Adv}_1$ outputs state information $\mathsf{state}_{\mathbf{Adv}_1}$ to $\mathbf{Adv}_2$ which is allowed to do the same actions subject only to one restriction: t_1 marks the time at which $\mathbf{Adv}_1$ has send its state information and at $t_2 + t_{puz}$ he must output the solutions to n puzzles created no sooner than t_1. The game returns 1 if the adversary has returned correct solutions for all n puzzles, i.e.,

$$\mathsf{Win}_{\mathbf{Adv}}^{\Pi_{t_{puz}}(\mathsf{CPuz}_{d,n}^k)}(t_2 - t_1, n) = \Pr\left[\mathsf{Exec}_{\mathbf{Adv}}^{\Pi_{t_{puz}}(\mathsf{CPuz}_{d,n}^k)} = 1\right]$$

Definition 9 (DoS Resilience). *Let* $\mathsf{CPuz}_{d,n}^k$ *be an unforgeable, difficulty preserving puzzle. Protocol* $\Pi_{d,t_{puz}}(\mathsf{CPuz}_{d,n}^k)$ *is* $\epsilon_{d,n}^k$-DoS resilient *if for any* $t_1, t_2 \in [0, t_\Pi]$ *with* $t_1 < t_2$*, having an adversary* $\mathbf{Adv}$ *that can perform at most* $t_{\mathbf{Adv}} : t_2 - t_1 + t_{puz}$ *computations in time* $t_2 - t_1 + t_{puz}$ *it holds:*

$$\Pr\left[\mathsf{Win}_{\mathbf{Adv}}^{\Pi_{t_{puz}}(\mathsf{CPuz}_{d,n}^k)}(t_2 - t_1, n)\right] \leq \epsilon_{d,n}^k(t_{\mathbf{Adv}} : t_2 - t_1 + t_{puz}) + \nu(k)$$

PRACTICAL APPLICABILITY. We sketch the practical applicability of our security notions. The next theorem links the efficacy of a puzzle-based DoS defense system with the parameters of the underlying puzzle. Informally, the theorem states that a puzzle scheme can protect a protocol only when the ratio between the computational power of the adversary and that of the client does not exceed service time (note that paradoxically this is independent on the hardness of the puzzle, an aspect that to best of our knowledge is overlooked in previous work on client puzzles). In practice, DoS is usually analyzed by means of queuing theory and the main parameter is service time $\theta_{service}$ which gives the maximum input rate that can be handled by the system. For example, if service time is $\theta_{service} = 10ms$ then the server can handle a maximum input rate $\lambda = 100$ connections each second and beyond this the systems gets saturated (leading to a waiting queue than can grow without bound). While previous definition is of theoretical interest, it can be easily translated to practical systems where resource exhaustion occurs as soon as the requests of an adversary exceed the inverse of the service time. In the proof of the following theorem, note that the lifetime of the puzzle t_{puz} from the $\epsilon^k_{d,n}$*-DoS resilience* is used to derive a practical bound that depends strictly on the computational resources and maximum acceptable load of the server $\theta^{-1}_{service}$.

Theorem 3. *Consider protocol $\Pi_{d,t_{puz}}(\mathsf{CPuz}^k_{d,n})$ runs on a server side with service time $\theta_{service}$ for each connection and the computational resources of the adversary and clients are $\pi_{\mathcal{A}}$ and $\pi_{\mathcal{C}}$ respectively. Protocol $\Pi_{d,t_{puz}}(\mathsf{CPuz}^k_{d,n})$ can provide DoS protection only if $\pi_{\mathcal{A}} < \pi_{\mathcal{C}} \cdot \theta^{-1}_{service}$ and the maximum level of protection is reached at $d = \pi_{\mathcal{A}}$.*

This bound seems to justify existing empirical results. Dean and Stubblefield [7] provided the first positive results for protecting SSL/TLS by using client puzzles. In the performance related section, the authors of [7] note that *20-bit puzzles* seem to offer the optimal level of protection. While this observation is only empirical, it is supported by the result of Theorem 3 which shows $d = \pi_{\mathcal{A}}$ as the maximum difficulty level and indeed in practice the computational power of an adversary is the order of 2^{20} hashes per second. For distributed DoS attacks these values must be scaled up with the size of the bot-net that the adversary controls.

6 Conclusion

We refined difficulty notions for puzzles, making a clear distinction between optimal, difficulty preserving, and ideal puzzles. Also we provided new difficulty bounds for two hash based puzzles. We showed that these bounds are tight enough to ensure optimality and that the puzzles are difficulty preserving. Finally, we introduced a stronger definition for DoS resilience motivated by the observation that previous definitions may still allow an adversary to mount a successful attack. As this is the third paper proposing rigorous difficulty notions for client puzzles and showing that previous definitions fail, it is clear that formalizing puzzles properties is not as easy as it may appear on first sight. Our definition opens the avenue of studying puzzles and their use in DoS defense in more detail than was possible in the past (e.g., by introducing new security notions and including an explicit puzzle management mechanism in the puzzle protocol). Previously, choosing puzzle difficulty in practice was only based on empirical observations, here we provided a clear upper bound for this as well as a bound on the usefulness of client puzzles against DoS. Namely, puzzles will work only if $\pi_{\mathcal{A}} < \pi_{\mathcal{C}} \cdot \theta^{-1}_{service}$ which places the computational power of the adversary and clients in a clear, crisp relation with network service time.

Acknowledgement. First author was partially supported by national research grant CNCSIS UEFISCDI, project number PNII IDEI 940/2008-2011 and by the strategic grant POSDRU/21/1.5/G/13798, inside POSDRU Romania 2007-2013, co-financed by the European Social Fund - Investing in People.

References

1. Abadi, M., Burrows, M., Manasse, M., Wobber, T.: Moderately hard, memory-bound functions. ACM Transactions on Internet Technology 5, 299–327 (2005)
2. Abliz, M., Znati, T.: A guided tour puzzle for denial of service prevention. In: Proceedings of the 2009 Annual Computer Security Applications Conference, ACSAC 2009, pp. 279–288. IEEE Computer Society (2009)
3. 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)
4. Back, A.: Hashcash - a denial of service counter-measure. Technical report (2002)
5. Boyd, C., Gonzalez-Nieto, J., Kuppusamy, L., Narasimhan, H., Rangan, C., Rangasamy, J., Smith, J., Stebila, D., Varadarajan, V.: An investigation into the detection and mitigation of denial of service (Dos) attacks: Critical information infrastructure protection. In: Cryptographic Approaches to Denial-of-Service Resistance, p. 183 (2011)
6. Chen, L., Morrissey, P., Smart, N.P., Warinschi, B.: Security Notions and Generic Constructions for Client Puzzles. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 505–523. Springer, Heidelberg (2009)
7. Dean, D., Stubblefield, A.: Using client puzzles to protect tls. In: Proceedings of the 10th Conference on USENIX Security Symposium, SSYM 2001, vol. 10, p. 1. USENIX Association, Berkeley (2001)
8. Dwork, C., Goldberg, A., Naor, M.: On Memory-Bound Functions for Fighting Spam. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 426–444. Springer, Heidelberg (2003)
9. 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)
10. Gao, Y., Susilo, W., Mu, Y., Seberry, J.: Efficient trapdoor-based client puzzle against DoS attacks. Network Security, 229–249 (2010)
11. Jeckmans, A.: Computational puzzles for spam reduction in SIP (draft) (July 2007)
12. Jeckmans, A.: Practical client puzzle from repeated squaring. Technical report (August 2009)
13. Jerschow, Y.I., Mauve, M.: Non-parallelizable and non-interactive client puzzles from modular square roots. In: Sixth International Conference on Availability, Reliability and Security, ARES 2011, pp. 135–142 (2011)
14. Juels, A., Brainard, J.: Client puzzles: A cryptographic countermeasure against connection depletion attacks. In: Proceedings of NDSS 1999 (Networks and Distributed Security Systems), pp. 151–165 (1999)
15. Karame, G.O., Čapkun, S.: Low-Cost Client Puzzles Based on Modular Exponentiation. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. LNCS, vol. 6345, pp. 679–697. Springer, Heidelberg (2010)
16. Rangasamy, J., Stebila, D., Boyd, C., Gonzalez Nieto, J.: An integrated approach to cryptographic mitigation of denial-of-service attacks. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 114–123. ACM (2011)
17. Rivest, R., Shamir, A., Wagner, D.: Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA (1996)

18. Stebila, D., Kuppusamy, L., Rangasamy, J., Boyd, C., Gonzalez Nieto, J.: Stronger Difficulty Notions for Client Puzzles and Denial-of-Service-Resistant Protocols. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 284–301. Springer, Heidelberg (2011)
19. Suriadi, S., Stebila, D., Clark, A., Liu, H.: Defending web services against denial of service attacks using client puzzles. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 25–32. IEEE (2011)
20. Tang, Q., Jeckmans, A.: On non-parallelizable deterministic client puzzle scheme with batch verification modes (2010)
21. Tritilanunt, S., Boyd, C., Foo, E., González Nieto, J.M.: Toward Non-parallelizable Client Puzzles. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 247–264. Springer, Heidelberg (2007)

A Proofs

A.1 Proof of Theorem 1

Suppose that Find finishes at exactly the t-th query and let $t = t_1 + t_2 + ... + t_n$ where t_i denotes the number of queries made to $\mathcal{H}$ to solve the i^{th} puzzle. The probability to solve the i^{th} puzzle at exactly the t_i query is obviously $(1 - \frac{1}{2^d})^{t_i - 1} \cdot \frac{1}{2^d}$. Since solving each puzzle is an independent event, the probability to solve the puzzles at exactly $t_1, t_2, ..., t_n$ steps for each puzzle is $\prod_{i=1,n}(1 - \frac{1}{2^d})^{t_i-1} \cdot \frac{1}{2^d} = (1 - \frac{1}{2^d})^{t-n} \cdot \frac{1}{2^{nd}}$. But there are exactly $\binom{t-1}{n-1}$ ways of writing t as a sum of exactly n integers from which the probability to solve the puzzle follows as: $\zeta^{HT}_{k,d,n}(t) = \sum_{i=n,t} \binom{i-1}{n-1} \cdot \frac{1}{2^{nd}} \cdot \left(1 - \frac{1}{2^d}\right)^{i-n}$.

We prove the adversary advantage in the random oracle model. For this, challenger $\mathcal{C}$ simulates $\mathcal{H}$ by flipping coins and playing the following game $\mathbf{G}_0$ with adversary $\mathcal{A}$:

(1) The challenger $\mathcal{C}$ runs Setup on input 1^k then it will flip coins to answer to the adversary $\mathcal{A}$,

(2) The adversary $\mathcal{A}$ is allowed to ask $\mathsf{GenSolvePuz}$, Test, $\mathsf{ComputeHash}$ which C answers as follows: (2.1) on $\mathsf{GenSolvePuz}$, challenger $\mathcal{C}$ picks $r \in \{0,1\}^k$ checks if r is present on its tape and stores it if not then randomly chooses a solution sol and returns the pair $\{r, \mathsf{sol}\}$, (2.2) on Test, challenger $\mathcal{C}$ queries itself $\mathsf{GenSolvePuz}$ but marks its answers and solutions as $\{(r_1^\diamondsuit, \mathsf{sol}_1^\diamondsuit), (r_2^\diamondsuit, \mathsf{sol}_2^\diamondsuit), ..., (r_n^\diamondsuit, \mathsf{sol}_n^\diamondsuit)\}$ and returns just $\{r_1^\diamondsuit, r_2^\diamondsuit, ..., r_n^\diamondsuit\}$, (2.3) on $\mathsf{ComputeHash}$, challenger $\mathcal{C}$ simulates $\mathcal{H}$ to the adversary A, that is, he receives r, sol from adversary $\mathcal{A}$, check if r, sol was not already queried and if not he flips coins to get y and stores stores the triple $\{r, \mathsf{sol}, y\}$ on its tape then returns y to $\mathcal{A}$,

(3) At any point the adversary $\mathcal{A}$ can stop the game by sending $\mathcal{C}$ a set of pairs $\{(r_1^\diamondsuit, \mathsf{sol}_1^\diamondsuit), (r_2^\diamondsuit, \mathsf{sol}_2^\diamondsuit), ..., (r_n^\diamondsuit, \mathsf{sol}_n^\diamondsuit)\}$,

(4) When challenger $\mathcal{C}$ receives $\{(r_1^\diamondsuit, \mathsf{sol}_1^\diamondsuit), (r_2^\diamondsuit, \mathsf{sol}_2^\diamondsuit), ..., (r_n^\diamondsuit, \mathsf{sol}_n^\diamondsuit)\}$ he checks that each $\{r_1^\diamondsuit, r_2^\diamondsuit, ..., r_n^\diamondsuit\}$ are stored on its tape and for each solution it checks that the last d bits of y in $\{r, \mathsf{sol}, y\}$ are zero. If a triple $\{r, \mathsf{sol}, y\}$ such that the last d bits of y are zero is not present on the tape, then challenger $\mathcal{C}$ flips coins one more time to get a new y and accepts the solution if y ends with d zeros (note that these values are not stored on the tape). If all these hold then challenger $\mathcal{C}$ outputs 1, otherwise it outputs 0.

Remark 14. For correct simulation of GenSolvePuz the length l of the correct answer should be chosen according to the probability distribution of the lengths for a particular difficulty level, i.e., $\Pr[l] = (1 - (1 - 2^{-d})^{2^l})(1 - 2^{-d})^{2^{l-1}}$.

Let $\mathbf{G}_1$ be the same as $\mathbf{G}_0$ with the following difference: on GenSolvePuz, challenger $\mathcal{C}$ picks $r \in \{0,1\}^k$ checks if r is present on its tape and aborts if so, otherwise it continues as in $\mathbf{G}_0$ by storing the values then sending them to $\mathcal{A}$. We have: $\Big|\Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_0\big] - \Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_1\big]\Big| \le \frac{q_{\mathsf{Gen}}^2}{2^{k+1}}$.

We now bound the adversary advantage in $\mathbf{G}_1$. At the end of the game, challenger $\mathcal{C}$ inspects his tape and sets t as the number of queries made to ComputeHash that have an $r_i^\diamondsuit, \forall i \in \{1, n\}$ as input. Let E_i denote the event that for i of the puzzles a pair $\{r^\diamondsuit, \mathsf{sol}^\diamondsuit, y\}$ where y ends with d zeros is not present on the tape. Obviously, there $n+1$ possible outcomes of $\mathbf{G}_1$: $E_0, E_1, ..., E_n$. In each E_i let $\Pr\big[\mathcal{A} \text{ wins } E_i\big]$ be the probability that the adversary has the correct answers for $n-i$ of the puzzles and he guessed the output of i of them which happens with probability 2^{-id} since the adversary never queried $\mathcal{H}$ to get a correct output. We have:

$$\Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_1\big] = \Pr\big[\mathcal{A} \text{ wins } E_0\big] + \frac{1}{2^d} \cdot \Pr\big[\mathcal{A} \text{ wins } E_1\big] + \frac{1}{2^{2d}} \cdot \Pr\big[\mathcal{A} \text{ wins } E_2\big] +$$
$$... + \frac{1}{2^{nd}} \cdot \Pr\big[\mathcal{A} \text{ wins } E_n\big] = \zeta_{k,d,n}(t) + \frac{1}{2^d} \cdot \Pr\big[\mathcal{A} \text{ wins } E_1\big] +$$
$$+ \frac{1}{2^{2d}} \cdot \Pr\big[\mathcal{A} \text{ wins } E_2\big] + ... + \frac{1}{2^{nd}} \cdot \Pr\big[\mathcal{A} \text{ wins } E_n\big] <$$
$$< \zeta_{k,d,n}^{HT}(t) + \frac{1}{2^d} + \frac{1}{2^{2d}} + ... + \frac{1}{2^{nd}} < \zeta_{k,d,n}^{HT}(t) + \frac{1}{2^d - 1}$$

By elementary calculations it follows that: $\mathsf{Win}_{\mathcal{A},d,n}^{\mathsf{HashTrail},k}(q_{\mathsf{Gen}}, t) \le \Big|\Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_0\big] - \Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_1\big]\Big| + \Pr\big[\mathcal{A} \text{ wins } \mathbf{G}_1\big] = \zeta_{k,d,n}^{HT}(t) + \frac{1}{2^d-1} + \frac{q_{\mathsf{Gen}}^2}{2^{k+1}}$.

The puzzle follows as optimal since $\epsilon_{k,d,n}^{HT}(t) \le \zeta_{k,d,n}^{HT}(t) + \frac{1}{2^d-1} + \frac{q_{\mathsf{Gen}}^2}{2^{k+1}}$ and $\frac{1}{2^d-1} + \frac{q_{\mathsf{Gen}}^2}{2^{k+1}}$ is negligible in d and k respectively.

Now we prove that the puzzle is difficulty preserving which is trivial to do. For $d = 1$ it is easy to prove that $\mathrm{t_{avr}}(k, 1, d) = 2^d$. This is straight forward since:

$$\mathrm{t_{avr}}(k,1,d) = \sum_{i=1,\infty} i \cdot \frac{1}{2^d} \cdot \left(1 - \frac{1}{2^d}\right)^{i-1} = \frac{1}{2^d} \cdot \sum_{i=1,\infty} i \cdot \left(1 - \frac{1}{2^d}\right)^{i-1} =$$
$$= \frac{1}{2^d} \cdot \lim_{i \to \infty} \frac{i \cdot \left(1 - \frac{1}{2^d}\right)^{i-1} \cdot \left(-\frac{1}{2^d}\right) - \left(1 - \frac{1}{2^d}\right)^i + 1}{\frac{1}{2^{2d}}} = 2^d$$

We now want to show that $n \cdot \mathrm{t_{avr}}(k, d, 1) = \mathrm{t_{avr}}(k, d, n)$. By definition we have $\zeta_{k,d,n}^{HT}(t) = \sum_{i=n,t} \binom{i-1}{n-1} \cdot \frac{1}{2^{nd}} \cdot \left(1 - \frac{1}{2^d}\right)^{i-n}$. Thus it follows:

$$\mathrm{t_{avr}}(k,d,n) = \sum_{i=n,\infty} i \cdot (\zeta_{k,d,n}^{HT}(t) - \zeta_{k,d,n}^{HT}(t-1)) = \sum_{i=n,\infty} i \cdot \binom{i-1}{n-1} \cdot \frac{1}{2^{nd}} \cdot \left(1 - \frac{1}{2^d}\right)^{i-n}$$

Recall that $\binom{i}{j} = \binom{i-1}{j-1} + \binom{i-1}{j}$ and write

$$\mathrm{t}_{\mathrm{avr}}(k,d,n) = \sum_{i=n,\infty} i \cdot \left[\binom{i-2}{n-2} + \binom{i-2}{n-1} \right] \cdot \frac{1}{2^{nd}} \cdot \left(1 - \frac{1}{2^d}\right)^{i-n} =$$

$$= \frac{1}{2^d} \cdot \underbrace{\sum_{i=n,\infty} i \cdot \binom{i-2}{n-2} \cdot \frac{1}{2^{(n-1)d}} \cdot \left(1-\frac{1}{2^d}\right)^{i-n}}_{\mathrm{t}_{\mathrm{avr}}(k,d,n-1) + \underbrace{\sum_{i=n,\infty} \binom{i-2}{n-2} \cdot \frac{1}{2^{(n-1)d}} \cdot \left(1-\frac{1}{2^d}\right)^{i-n}}_{\epsilon_{k,d,n-1}(\infty)=1}}$$

$$+ \left(1-\frac{1}{2^d}\right) \underbrace{\sum_{i=n,\infty} i \cdot \binom{i-2}{n-1} \cdot \frac{1}{2^{nd}} \cdot \left(1-\frac{1}{2^d}\right)^{i-n-1}}_{\mathrm{t}_{\mathrm{avr}}(k,d,n) + \underbrace{\sum_{i=n,\infty} \binom{i-2}{n-2} \cdot \frac{1}{2^{(n-1)d}} \cdot \left(1-\frac{1}{2^d}\right)^{i-n}}_{\epsilon_{k,d,n}(\infty)=1}}$$

Multiply with 2^d to get $\mathrm{t}_{\mathrm{avr}}(k,d,n) = \mathrm{t}_{\mathrm{avr}}(k,d,n-1) + 2^d$ from which by recurrence we have $\mathrm{t}_{\mathrm{avr}}(k,d,n) = n \cdot \mathrm{t}_{\mathrm{avr}}(k,d,1)$ which completes the proof.

A.2 Proof of Theorem 3

Let $\mathcal{R}$ denote the number of resources takeover by the adversary and λ the number of requests to the server. We have $\lambda \in [0, \lambda_{\mathcal{A}}]$ where $\lambda_{\mathcal{A}}$ is the maximum rate at which an adversary can request connections (limited by network parameters only). Obviously a DoS takes place if $\lambda > \theta_{service}^{-1}$ since the server can handle at most $\theta_{service}^{-1}$ connections each second. But by using client puzzle the number of requests is also bounded by the computational power of the adversary. A misleading bound on the adversary request rate is $\lambda_{\max} = \frac{\pi_{\mathcal{A}}}{d}$. By careful inspection of Definition 9 the difficulty bound includes the puzzle lifetime t_{puz} and the correct bound is $\lambda_{\max} = \frac{\pi_{\mathcal{A}} + t_{puz}\pi_{\mathcal{A}}}{d}$ (since all puzzle computed during t_{puz} can be used as well to gain resources). But puzzle lifetime t_{puz} must be bigger than the time a client needs to solve the puzzle, i.e., $t_{puz} > d\pi_{\mathcal{C}}^{-1}$, since otherwise clients are unable to solve the puzzles and cannot get resources anyway. Thus $\lambda_{\max} > \frac{\pi_{\mathcal{A}}}{d} + \frac{\pi_{\mathcal{A}}}{\pi_{\mathcal{C}}}$. It follows: $\mathcal{R}(\lambda) = \lambda$, if $\lambda \in \left[0, \frac{\pi_{\mathcal{A}}}{d} + \frac{\pi_{\mathcal{A}}}{\pi_{\mathcal{C}}}\right]$. Which means that the number of resources drops with the increase in the difficulty of the puzzle but it never drops below $\frac{\pi_{\mathcal{A}}}{\pi_{\mathcal{C}}}$ since: $\lim_{d \to +\infty} \mathcal{R}(\lambda) = \frac{\pi_{\mathcal{A}}}{\pi_{\mathcal{C}}}$. Accordingly, the adversary can always get at least $\pi_{\mathcal{A}} \cdot \pi_{\mathcal{C}}^{-1}$ resources, regardless of the puzzle difficulty level, and the DoS condition is met when $\pi_{\mathcal{A}} \cdot \pi_{\mathcal{C}}^{-1} \geq \theta_{service}^{-1}$. Obviously $\pi_{\mathcal{A}} \cdot \pi_{\mathcal{C}}^{-1}$ is the minimum amount of resources gained on the side of the adversary and this met as soon as $d > \pi_{\mathcal{A}}$.

On Optimal Bounds of Small Inverse Problems and Approximate GCD Problems with Higher Degree

Noboru Kunihiro

The University of Tokyo, Japan
kunihiro@k.u-tokyo.ac.jp

Abstract. We show a relation between optimal bounds of a small inverse problem and an approximate GCD problem. First, we present a lattice based method to solve small inverse problems with higher degree. The problem is a natural extension of small secret exponent attack on RSA cryptosystem introduced by Boneh and Durfee. They reduced this attack to solving a bivariate modular equation: $x(A+y) \equiv 1 \pmod{e}$, where A is a given integer and e is a public exponent. They proved that the problem can be solved in polynomial time when $d \leq N^{0.292}$. In this paper, we extend the Boneh–Durfee's result to more general problem. For a monic polynomial $h(y)$ of degree $\kappa(\geq 1)$, integers C and e, we want to find all small roots of a bivariate modular equation: $xh(y) + C \equiv 0 \pmod{e}$. We denote by X and Y the upper bound of roots. We present an algorithm for solving the problem and prove that the problem can be solved in polynomial time if $\gamma \leq 1 - \sqrt{\kappa\alpha}$ and $|C|$ is small enough, where $X = e^{\gamma}$ and $Y = e^{\alpha}$. We employ a similar approach as unravelled linearization technique introduced by Herrmann and May in especially evaluating the lattice volume. Interestingly, our algorithm does not rule out the case of $C = 0$, which implies that our algorithm can solve a univariate unknown modular equation $h(y) \equiv 0 \pmod{p}$, where p is unknown. Our algorithm achieves the best bound in the literature. Then, we show that our obtained bound is *natural* under the similar sense of Howgrave-Graham's discussion in CaLC2001 and we prove that our bound, including Boneh–Durfee's bound, is optimal under the reasonable assumption.

Keywords: RSA Cryptosystem, LLL algorithm, Small Inverse Problem, Approximate GCD Problem.

1 Introduction

RSA cryptosystem is the widely used cryptosystem [16]. Let N be an RSA moduli and d, e be secret and public exponents, respectively. Wiener showed that when $d < N^{1/4}$, the RSA can be broken in polynomial time [17]. In 1999, Boneh and Durfee first improved the bound to $d \leq N^{0.284}$ and then they improved the bound to $d \leq N^{0.292}$ [1]. They reduced a small secret exponent attack to solving a bivariate modular equation: $x(A+y) \equiv 1 \pmod{e}$ and proposed an efficient algorithm for solving this problem. They referred this problem as "small inverse problem". Their attack is based on Coppersmith's lattice based techniques [2–4]. In their proof for $d \leq N^{0.292}$, they introduced the concept of *Geometrically Progressive Matrix* and gave an upper bound of the volume of generated lattice. Then, its proof is very complicated.

In 2010, Herrmann and May [8] gave an elementary proof to Boneh-Durfee's stronger bound [1]. They employed unravelled linearization technique developed in [7]. In 2011, Kunihiro et al. gave a unified framework for small secret exponent attack [13]. In their analysis, they employed unravelled linearization technique too and use two-parameter space. Furthermore, they optimized the bound in their framework and proved that the Boneh–Durfee's stronger bound is still optimal in their framework. [13] also discussed more general problem $x(A+y)+C \equiv 0 \pmod{e}$ not necessary for $C \neq 1$ but small $|C|$.

On the other hand, Kunihiro introduced a class of *Generalized Small Inverse Problem* by extending small inverse problem [11]. This problem is formulated as finding all small roots of $n+1$-variate modular equation: $x_0 h(x_1, \ldots, x_n) + C \equiv 0 \pmod{N}$ for an n-variate polynomial $h(x_1, \ldots, x_n)$, integers C and N. Then, [12] demonstrated a general method which solves this problem. If we apply it to the small inverse problem, the obtained bound is $d \leq N^{0.284}$. Note that his algorithm does not achieve the stronger bound: $d \leq N^{0.292}$. In [12], the algorithm which solves $xh(y) + C \equiv 0 \pmod{e}$ for $h(y)$ of degree κ was also proposed. The condition that the problem can be solved in polynomial time was derived. Let $(x, y) = (x_0, y_0)$ be the solution satisfying $|x_0| < X := e^{\gamma}$ and $|y_0| < Y := e^{\alpha}$. The condition is given by $\gamma \leq 1 - \frac{2\sqrt{(\kappa\alpha)^2 + 3\kappa\alpha} - \kappa\alpha}{3}$.

Howgrave-Graham [10] introduced approximate common divisor problems[1]. Let a be an integer and N be a composite whose prime factor p is unknown. He showed an algorithm for solving a modular equitation: $a + y \equiv 0 \pmod{p}$ in polynomial time. May extended the problem into that with more higher degree. Letting $h(y)$be a monic polynomial with degree κ, he showed a polynomial time algorithm for solving $h(y) \equiv 0 \pmod{p}$. He showed the condition that the problem is solvable is $Y \leq N^{\beta^2/\kappa}$, where Y is an upper bound of the absolute value of the solution and $p = N^{\beta}$.

Howgrave-Graham [10] discussed the relation between two problems: $x(A + y) \equiv 0 \pmod{N}$ and $x(A + y) \equiv 1 \pmod{N}$. Then, he claimed that Boneh-Durfee's stronger bound [1] is *natural* if there exist a reduction between two problems.

1.1 Our Contributions

Improving the Bound for Small Inverse Problems with Higher Degree. We consider the following problem. Given a positive integer e, an integer C and a monic polynomial $h(y)$ with degree κ, we want to find all small roots of a

[1] He introduced a *partially* approximate common divisor problem and a *general* approximate common divisor problem. In this paper, we discuss only the *partial* one.

bivariate modular equation: $xh(y) + C \equiv 0 \pmod{e}$. We denote the solution of the equation by $(x, y) = (x_0, y_0)$. Let $X(:= e^{\gamma})$ and $Y(:= e^{\alpha})$ be the upper bound of $|x_0|$ and $|y_0|$, respectively. We first prove that the problem can be solved if

$$\gamma \leq 1 - \sqrt{\kappa\alpha},$$

$|C| < XY^{\kappa}$ and $\kappa/(\kappa+1)^2 \leq \alpha \leq 1/\kappa$. It is a better bound than that obtained in [12]. In the proof, we employ the similar technique as unravelled linearization technique [8]. This result is a natural extension of the Herrmann and May's result [8] and Kunihiro et al.'s result [13]; the degree of $h(y)$ is larger than or equal to 1.

Naturalness and Optimality of Our Bounds. Interestingly, our proposed algorithm does not rule out of the case: $C = 0$. Then, our algorithm can inherently solve the univariate unknown modular equation: $h(y) \equiv 0 \pmod{p}$, where p is unknown but its multiplier N is known. The class of these problems includes a (partially) approximate GCD problem [10]. It is remarkable that fully homomorphic encryption schemes over integers [5, 6] are based on the difficulty of this problem. Then, it is important to understand the difficulty of approximate GCD problem. Suppose that $p := N^{\beta}$ and the solution y_0 is upper bounded by N^{α}. The best known bound for the problem is $\alpha < \beta^2/\kappa$ due to May [15]. We prove that our algorithm achieves the same bound as the best known algorithm. Furthermore, we show that our obtained bound: $\gamma \leq 1 - \sqrt{\kappa\alpha}$ is *optimal* under a reasonable assumption: the optimal bound for $h(y) \equiv 0 \pmod{p}$ is $\alpha \leq \beta^2/\kappa$. Last, we show that the bound obtained in this paper is also natural under the similar sense of Howgrave-Graham's discussion.

1.2 Organization

Section 2 gives preliminaries. In this section, we introduce mathematical lemmas and review previous known results. In Section 3, we present an efficient algorithm for solving $xh(y) + C \equiv 0 \pmod{e}$ with a monic polynomial $h(y)$. We first propose how to select shift-polynomials in constructing a lattice basis. In evaluating the volume of constructed lattice, we employ a similar technique as unravelled linearization. Furthermore, we derive a condition that the problem can be solved in polynomial time. In Section 4, we show that our algorithm can also find all small roots of the univariate unknown modular equation and achieves the same bound as best known bound. Then, we show that our obtained bound $\gamma \leq 1 - \sqrt{\kappa\alpha}$ is optimal under the reasonable assumption. Then, the obtained bound is natural under the similar sense of Howgrave-Graham's discussion. Section 5 concludes a paper.

2 Preliminaries

2.1 LLL Algorithm and Howgrave-Graham's Lemma

For a vector $\boldsymbol{b} = (b_1, b_2, \ldots, b_w)$, $\|\boldsymbol{b}\|$ denotes the Euclidean norm of $\boldsymbol{b}$: $\|\boldsymbol{b}\| = \sqrt{\sum_{i=1}^{w} b_i^2}$. For a trivariate polynomial $h(x, y, z) = \sum h_{i,j,k} x^i y^j z^k$, define the norm of a polynomial as $\|h(x,y,z)\| = \sqrt{\sum h_{i,j,k}^2}$. That is, $\|h(x,y,z)\|$ denotes the Euclidean norm of the vector which consists of coefficients of $h(x,y,z)$.

Let $B = \{a_{ij}\}$ be a $w \times w$ matrix of integers. Suppose that the rows of B are linearly independent. The rows of B generate a lattice L, a collection of vectors closed under addition and subtraction; in fact the rows forms a basis of L. The lattice L is also represented as follows. Letting $\boldsymbol{a_i} = (a_{i1}, a_{i2}, \ldots, a_{iw})$, the lattice L spanned by $\langle \boldsymbol{a_1}, \ldots, \boldsymbol{a_w} \rangle$ consists of all integral linear combinations of $\boldsymbol{a_1}, \ldots, \boldsymbol{a_w}$, that is:

$$L = \left\{ \sum_{i=1}^{w} n_i \boldsymbol{a_i} | n_i \in \mathbb{Z} \right\}.$$

The volume of full-rank lattice is given by $\mathrm{vol}(L) = |\det(B)|$.

The LLL algorithm outputs short vectors in the lattice L.

Proposition 1 (LLL [14]). *Let $B = \{a_{ij}\}$ be a non-singular $w \times w$ matrix of integers. The rows of B generate a lattice L. Given B, the LLL algorithm finds vectors $\boldsymbol{b_1}$ and $\boldsymbol{b_2} \in L$ such that*

$$\|\boldsymbol{b_1}\| \le 2^{(w-1)/4}(\mathrm{vol}(L))^{1/w} \text{ and } \|\boldsymbol{b_2}\| \le 2^{w/4}(\mathrm{vol}(L))^{1/(w-1)}$$

in time polynomial in $(w, \max \log_2 |a_{ij}|)$.

We reduce the root finding problem in modular equation to the case of root finding equation over integers by using the following lemma.

Lemma 1 (Howgrave-Graham [9]). *Let $\hat{h}(x,y,z) \in \mathbb{Z}[x,y,z]$ be a polynomial, which is a sum of at most w monomials. Let m be a positive integer and X, Y, Z and ϕ be some positive integers. Suppose that*

1. *$\hat{h}(\bar{x},\bar{y},\bar{z}) \equiv 0 \pmod{\phi^m}$, where $|\bar{x}| < X, |\bar{y}| < Y, |\bar{z}| < Z$.*
2. *$\|\hat{h}(xX, yY, zZ)\| < \phi^m/\sqrt{w}$.*

Then $\hat{h}(\bar{x},\bar{y},\bar{z}) = 0$ holds over integers.

2.2 Known Algorithm for Solving $xh(y) + C \equiv 0 \pmod{e}$ with Non-linear $h(y)$

Let $h(y)$ be a monic polynomial with degree κ. We consider a bivariate modular equation: $f(x,y) = xh(y) + C \equiv 0 \pmod{e}$. [12] proposed a lattice based method for solving this problem.

The shift-polynomials are defined by

$$g_{[i,j,k]}(x,y) := x^i y^j f(x,y)^k e^{m-k}.$$

A set of shift-polynomials is denoted by $\mathcal{F}_{\mathrm{w}}$. The set $\mathcal{F}_{\mathrm{w}}$ is given by $\mathcal{F}_{\mathrm{w}} := \mathcal{G}_{\mathrm{w}} \cup \mathcal{H}_{\mathrm{w}}$, where

$$\mathcal{G}_{\mathrm{w}} := \{g_{[u-i,j,i]} : u = 0, \ldots, m; i = 0, \ldots, u; j = 0, \ldots, \kappa - 1\} \text{ and}$$
$$\mathcal{H}_{\mathrm{w}} := \{g_{[0,j,u]} : u = 0, \ldots, m; j = \kappa, \ldots, t\}.$$

Note that t is a parameter to be optimized.

Let $(x, y) = (x_0, y_0)$ be a solution of $f \equiv 0 \pmod{e}$, where it holds that $|x_0| < X(:= e^{\gamma}), |y_0| < Y(:= e^{\alpha})$. [12] proved that this problem can be solved in polynomial time if

$$\gamma \leq 1 - \frac{2\sqrt{(\kappa\alpha)^2 + 3\kappa\alpha} - \kappa\alpha}{3}. \tag{1}$$

If $\kappa = 1$ and $\alpha = 1/2$, Eq. (1) leads to the Boneh and Durfee's *weaker* bound: $d \leq N^{0.284}$. Then, we can say that Eq. (1) is a weaker bound. One of our contribution is to improve the bound to a stronger version of bound: $\gamma \leq 1-\sqrt{\kappa\alpha}$.

A common strategy for improving the bound is to remove "damaging" polynomial. The resulting lattice may be no longer full rank and computing its volume is not so easy. We resolve this problem by developing a similar technique as unravelled linearization introduced by Herrmann and May [8].

3 Improving the Bound for $xh(y) + C \equiv 0 \pmod{e}$

Let $h(y)$ be a monic polynomial with degree κ. We address how to solve

$$f(x, y) := xh(y) + C \equiv 0 \pmod{e}.$$

Our result is an extension of [8] and [13]. That is, [8] and [13] only discuss the case of $\kappa = 1$, but, we extend the result to any positive integer κ. Note that Herrmann and May claimed that *linearization* is effective in their analysis. However, our analysis is not fully based on *linearization* technique.

3.1 A New Method for Selecting Shift-Polynomials

Let $h(y) := a_\kappa + a_{\kappa-1}y + \cdots + a_1 y^{\kappa-1} + y^\kappa$. Here, we set $H(y) := a_\kappa + a_{\kappa-1}y + \cdots + a_1 y^{\kappa-1}$ and then $h(y) = y^\kappa + H(y)$. Let $f(x, y) := xh(y) + C$. We want to find all small roots of

$$f(x, y) = x(a_\kappa + a_{\kappa-1}y + \cdots + a_1 y^{\kappa-1} + y^\kappa) + C \equiv 0 \pmod{e}.$$

Let $(x, y) = (x_0, y_0)$ be a solution of $f(x, y) \equiv 0 \pmod{e}$ and X, Y be the upper bound of $|x_0|$ and $|y_0|$.

First, we transform $f(x, y)$ into

$$f(x, y) = xh(y) + C = x(y^\kappa + H(y)) + C = (xy^\kappa + C) + xH(y).$$

Letting $z := xy^\kappa + C$, we have $\bar{f}(x, y, z) := z + xH(y)$. Since $H(y)$ in [8] is a constant, this transformation is considered as "linearization". However, $H(y)$ in our paper is a polynomial of degree $\kappa - 1$; this transformation is not fully linearization if $\kappa > 1$.

We define shift-polynomials as

$$\bar{g}_{[i,j,k]}(x, y, z) := x^i y^j \bar{f}(x, y, z)^k e^{m-k} = x^i y^j (z + xH(y))^k e^{m-k}.$$

Letting $z_0 := x_0 y_0^{\kappa} + C$, it is easy to see that $\bar{g}_{[i,j,k]}(x_0, y_0, z_0) \equiv 0 \pmod{e^m}$ for any non-negative integers i, j and k. Note that the upper bound of $|z_0|$ is given by $XY^{\kappa} + |C|$. We let $Z := XY^{\kappa} + |C|$.

We denote by $\mathcal{F}$ a set of shift-polynomials. We give the set $\mathcal{F}$ as $\mathcal{F} := \mathcal{G} \cup \mathcal{H}$, where

$$\mathcal{G} := \{\bar{g}_{[u-i,j,i]} : u = 0, \ldots, m; i = 0, \ldots, u; j = 0, \ldots, \kappa - 1\} \text{ and}$$
$$\mathcal{H} := \{\bar{g}_{[0,j,u]} : u = 0, \ldots, m; j = \kappa, \ldots, \kappa + \tau u - 1\}.$$

Note that τ is a parameter to be optimized and $0 \leq \tau \leq 1$. We omit roundings as their contribution is negligible for sufficiently large m.

We define polynomial and monomial orders as follows.

polynomial order: We define $\preceq$ as $\bar{g}_{[i,j,k]} \preceq \bar{g}_{[i',j',k']}$

$$\text{if} \begin{cases} i + k < i' + k' \text{ or} \\ i + k = i' + k' \text{ and } i < i' \text{ or} \\ i = i', k = k' \text{ and } j \leq j'. \end{cases}$$

monomial order: We define $\preceq$ as $x^i y^j z^k \preceq x^{i'} y^{j'} z^{k'}$

$$\text{if} \begin{cases} i + k < i' + k' \text{ or} \\ i + k = i' + k' \text{ and } i < i' \text{ or} \\ i = i', k = k' \text{ and } j \leq j'. \end{cases}$$

We write $a \prec b$ if $a \preceq b$ and $a \neq b$.

3.2 Expansion in Shift-Polynomials

As well as [8] and [13], we will manipulate the underlying polynomial rather than the corresponding matrix. We substitute each occurrence of xy^{κ} by the term $z - C$. We call the form obtained by the substitution *compressed form*.

A lattice basis is constructed by using the coefficient vectors of shift-polynomials in $\mathcal{F}$ as basis vectors. Note that the coefficient vectors of the shift-polynomials $\bar{g}_{[i,j,k]}(xX, yY, zZ)$ are written as row vectors. Let $B(m; \tau)$ be a matrix, where all rows of $B(m; \tau)$ are coefficient vectors of shift-polynomials according to the ordering $\preceq$ of $\mathcal{F}$.

Theorem 1. *Let m be an integer. Let τ be a parameter with $0 \leq \tau \leq 1$. A lattice basis matrix $B(m; \tau)$ is triangular for any m and τ.*

To prove Theorem 1, we introduce some notations. We partially use the same symbols as [13].

Definition 1. *We denote by $\mathcal{S}(f)$ a set of monomials appearing in expansion of f on the compressed form.*

Definition 2. *We say $f(x, y) \cong g(x, y)$ if $\mathcal{S}(f) = \mathcal{S}(g)$.*

Definition 3. *We define* $\mathcal{F}(f) := \{g \in \mathcal{F} | g \prec f\}$ *and define* $\mathcal{S}(\mathcal{F}') := \bigcup_{g\in\mathcal{F}'} \mathcal{S}(g)$ *for* $\mathcal{F}' \subseteq \mathcal{F}$.

We have especially $\mathcal{S}(\mathcal{F}(f)) := \bigcup_{g\in\mathcal{F}(f)} \mathcal{S}(g)$. In general, it is enough for proving Theorem 1 to show that for any polynomial $f \in \mathcal{F}$ there exist a monomial m_f such that

- $\mathcal{S}(f - m_f) \subseteq \mathcal{S}(\mathcal{F}(f))$ and
- $m_f \notin \mathcal{S}(\mathcal{F}(f))$.

We denote $\mathcal{M} := \mathcal{S}(\mathcal{F})$. For $x^i y^j z^k \in \mathcal{M}$, we define $\mathcal{M}(x^i y^j z^k)$ according to the above monomial order as

$$\mathcal{M}(x^i y^j z^k) = \{x^{i'} y^{j'} z^{k'} \in \mathcal{M} | x^{i'} y^{j'} z^{k'} \prec x^i y^j z^k\}.$$

This means that $\mathcal{M}(x^i y^j z^k)$ is a set of elements before $x^i y^j z^k$.

Proof. It is enough for proving Theorem 1 to show that the following two lemmas.

Lemma 2. *For any* $\bar{g}_{[i,j,k]} \in \mathcal{F}$, $\mathcal{S}(\bar{g}_{[i,j,k]} - e^{m-k} x^i y^j z^k) \subseteq \mathcal{S}(\mathcal{F}(\bar{g}_{[i,j,k]}))$

Lemma 3. *For any* $\bar{g}_{[i,j,k]} \in \mathcal{F}$, $x^i y^j z^k \notin \mathcal{S}(\mathcal{F}(\bar{g}_{[i,j,k]}))$.

From Lemmas 2 and 3, we directly have Theorem 1. □

To prove Lemma 2, we prove the following two small lemmas.

Lemma 4. *For any* $\bar{g}_{[i,j,k]} \in \mathcal{F}$, $\mathcal{S}(\bar{g}_{[i,j,k]} - e^{m-k} x^i y^j z^k) \subseteq \mathcal{M}(x^i y^j z^k)$.

Lemma 5. *For any* $\bar{g}_{[i,j,k]} \in \mathcal{F}$, $\mathcal{S}(\mathcal{F}(g_{[i,j,k]})) = \mathcal{M}(x^i y^j z^k)$.

To prove Lemma 4, we prove the following three small lemmas.

Lemma 6. *If* $\bar{g}_{[u,j,0]} \in \mathcal{G}$,

$$\mathcal{S}(g_{[u-i,j,i]} - e^{m-i} x^{u-i} y^j z^i) = \emptyset.$$

Lemma 7. *If* $i \geq 1$ *and* $\bar{g}_{[u-i,j,i]} \in \mathcal{G}$,

$$\mathcal{S}(g_{[u-i,j,i]} - e^{m-i} x^{u-i} y^j z^i) \subseteq \mathcal{M}(x^{u-i} y^j z^i).$$

Lemma 8. *If* $\bar{g}_{[0,j,u]} \in \mathcal{H}$,

$$\mathcal{S}(\bar{g}_{[0,j,u]} - e^{m-u} y^j z^u) \subseteq \mathcal{M}(y^j z^u).$$

Proofs of Lemmas 6, 7 and 8 are given in Appendix A.

Proof of Lemma 3
It is enough to show that for any $[i',j',k'] \in \mathcal{F}(\bar{g}_{[i,j,k]})$ it holds that $x^iy^jz^k \notin \mathcal{S}(\bar{g}_{[i',j',k']})$. From Lemma 2, it trivially holds. Then, we have the lemma. $\square$

We give a small example. Let $h(y) = y^2 + a_1y + a_2$ and $m = 2, \tau = 1$. Since $\kappa = 2$, we will substitute each occurrence xy^2 by the term $z - C$. The polynomial $\bar{g}_{[0,2,1]} \in \mathcal{H} \subseteq \mathcal{F}$ is calculated as follows:

$$\begin{aligned}\bar{g}_{[0,2,1]} &= e^{2-1}x^0y^2(z + x(a_1y + a_2))^1 = ey^2z + exy^2(a_1y + a_2)\\ &= ey^2z + e(z - C)(a_1y + a_2) = ey^2z + e(z - C)a_1y + e(z - C)a_2\\ &= ey^2z + ea_1yz - eCa_1y + ea_2z - eCa_2\end{aligned}$$

We can easily verify that $\{yz, y, z, 1\} \subseteq \mathcal{M}(y^2z)$. More examples are given in Appendix B.

3.3 Deriving a Condition

Remember that $Z = XY^\kappa + |C|$. For Theorem 1, the volume $\mathrm{vol}(L)$ is given by

$$\mathrm{vol}(L) = \prod_{u=0}^{m}\prod_{k=0}^{u}\prod_{j=0}^{\kappa-1} X^{u-k}Y^jZ^ke^{m-k}\prod_{u=0}^{m}\prod_{j=0}^{\tau u-1} Y^{\kappa+j}Z^ue^{m-u}.$$

We describe $\mathrm{vol}(L)$ as $X^{s_X}Y^{s_Y}Z^{s_Z}e^{s_e}$. The values s_X, s_Y, s_Z, s_e and $\dim(L)$ are explicitly given by

$$\begin{aligned}s_X &= \sum_{u=0}^{m}\sum_{k=0}^{u}\sum_{j=0}^{\kappa-1}(u-k) = \frac{\kappa}{6}m^3 + o(m^3)\\ s_Y &= \sum_{u=0}^{m}\sum_{k=0}^{u}\sum_{j=0}^{\kappa-1} j + \sum_{u=0}^{m}\sum_{j=0}^{\tau u-1}(\kappa + j) = \frac{\tau^2}{6}m^3 + o(m^3)\\ s_Z &= \sum_{u=0}^{m}\sum_{k=0}^{u}\sum_{j=0}^{\kappa-1} k + \sum_{u=0}^{m}\sum_{j=0}^{\tau u-1} u = \left(\frac{\kappa}{6} + \frac{\tau}{3}\right)m^3 + o(m^3)\\ s_e &= \sum_{u=0}^{m}\sum_{k=0}^{u}\sum_{j=0}^{\kappa-1}(m-k) + \sum_{u=0}^{m}\sum_{j=0}^{\tau u-1}(m-u) = \left(\frac{\kappa}{3} + \frac{\tau}{6}\right)m^3 + o(m^3)\\ \dim(L) &= \sum_{u=0}^{m}\sum_{k=0}^{u}\kappa + \sum_{u=0}^{m}\tau u = \left(\frac{\kappa}{2} + \frac{\tau}{2}\right)m^2 + o(m^2).\end{aligned}$$

From now on, we divide the analysis into three cases based on the values of α and C.

Case 1: The first is for small $|C|$ and $\kappa/(\kappa+1)^2 \le \alpha \le 1/\kappa$;
Case 2: the second is for small $|C|$ and $0 < \alpha < \kappa/(\kappa+1)^2$;
Case 3: the third is for large $|C|$.

Note that in [8], the case: $C = 1$, $\kappa = 1$ and $\alpha = 1/2$ was only considered, which is included in Case 1.

For Case 1, which is the most important case, we have the following theorem.

Theorem 2. *Let $X = e^\gamma$ and $Y = e^\alpha$. Suppose that $XY^\kappa \ge |C|$ and $\kappa/(\kappa+1)^2 \le \alpha \le 1/\kappa$. The maximal upper bound of γ is $1 - \sqrt{\kappa\alpha}$.*

Proof. Since $Z = XY^\kappa + |C| \le 2XY^\kappa$, we have

$$\begin{aligned}\mathrm{vol}(L) &\le X^{s_X} Y^{s_Y} 2^{s_Z} (XY^\kappa)^{s_Z} e^{s_e} = 2^{s_z} X^{s_X+s_Z} Y^{s_Y+\kappa s_Z} e^{s_e} \\ &= (2^{1+2\tau} X^{2(\kappa+\tau)} Y^{(\kappa+\tau)^2} e^{2\kappa+\tau})^{(m^3/6+o(m^3))}.\end{aligned}$$

From Lemma 1, the condition that the problem can be solved in polynomial time is given by $\mathrm{vol}(L) \le e^{m\,\dim(L)}$. Letting $X := e^\gamma$ and $Y := e^\alpha$, this condition can be simplified into

$$\gamma \le 1 - \frac{1}{2}\left(\frac{\kappa}{(\tau+\kappa)} + (\kappa+\tau)\alpha\right). \tag{2}$$

By optimizing the right side of Eq. (2), we have

$$\gamma \le 1 - \sqrt{\kappa\alpha}. \tag{3}$$

In this case, the optimal setting of τ is given by $\tau = \sqrt{\kappa/\alpha} - \kappa$. □

Remark 1. It holds that $0 \le \tau \le 1$ when $\kappa/(\kappa+1)^2 \le \alpha \le 1/\kappa$.

Next, we show an analysis for Case 2. Suppose that $XY^\kappa \ge |C|$ and $0 < \alpha < \kappa/(\kappa+1)^2$. The analysis is almost the same as Theorem 2 and we have Eq. (2). In this case, the optimal value of τ is given by $\tau = 1$. The optimal bound of γ is given by

$$\gamma \le \frac{\kappa+2}{2(1+\kappa)} - \frac{\kappa+1}{2}\alpha. \tag{4}$$

Finally, we show an analysis for Case 3. Suppose that $XY^\kappa \le |C|$. Since $Z \le 2|C|$, we have

$$\mathrm{vol}(L) \le (2|C|)^{s_Z} X^{s_X} Y^{s_Y} e^{s_e}.$$

By similar calculation, we have a condition:

$$X^\kappa Y^{\tau^2} < \left(\frac{e}{|C|}\right)^{\kappa+2\tau}.$$

If $|C| \approx e$ especially, we have $X^\kappa Y^{\tau^2} < 1$. Hence, we cannot solve the problem in polynomial time in this case.

4 Discussion – The Bound Is Natural and Optimal?–

We discuss the following two problems[2].

Problem A: Let N be a positive integer. For a monic polynomial $h(y)$ and an integer C, find all small roots of a bivariate modular equation: $xh(y)+C \equiv 0 \pmod{N}$.

Problem B: Let $N(=pq)$ be a composite number of unknown factorization. For a monic polynomial $h(y)$, find all small roots of a univariate unknown modular equation: $h(y) \equiv 0 \pmod{p}$.

We especially denote Problem A with $C = 0$ by Problem A$^{(0)}$ and denote Problem A with $C = 1$ by Problem A$^{(1)}$.

Problem B has a strong relation with (partially) approximate GCD problem [10]. It is remarkable that fully homomorphic encryption schemes over integers [5, 6] are based on the difficulty of this problem.

There are reductions between Problem A$^{(0)}$ and Problem B. That is, Problem B can be solved in polynomial time if we can solve Problem A$^{(0)}$. The converse is also true. We show explicit reduction between two problems in Section 4.1. On the other hand, no reductions between Problem A$^{(1)}$ and Problem A$^{(0)}$ are known.

With regards to Problem B, May [15] proposed an algorithm which solves Problem B: $h(y) \equiv 0 \pmod{p}$. Suppose that $p \geq N^{\beta}$. Let Y be the upper bound of the solution. He showed that if

$$Y \leq N^{\beta^2/\kappa}, \tag{5}$$

this problem can be solved in polynomial time. So far, the above bound is the best known algorithm.

4.1 Our Algorithm also Solves Problem B: $h(y) \equiv 0 \pmod{p}$

As shown in Section 3, our algorithm solves Problem A for small enough $|C|$. As easily verified, our algorithm does not rule out the case of $C = 0$. Then, we can also solve Problem A$^{(0)}$ and then Problem B.

We show an explicit algorithm which solves Problem B: $h(y) \equiv 0 \pmod{p}$ by the following idea. First, find the small root of $xh(y) \equiv 0 \pmod{N}$ by our proposed algorithm in Section 3. Let $y = y_0$ be the solution of $h(y) \equiv 0 \pmod{p}$ and Y be $Y := N^{\alpha}$. We know that one of the solution of $xh(y) \equiv 0 \pmod{N}$ is given by $(x, y) = (q, y_0)$. Then, we can find the solution $y = y_0$ of $h(y) \equiv 0 \pmod{p}$ by using our proposed algorithm.

Next, we evaluate the condition at which our algorithm solves Problem B. Letting $p = N^{\beta}$, we have $q = N^{1-\beta}$. The condition at which our algorithm solves Problem A$^{(0)}$: $xh(y) \equiv 0 \pmod{N}$ is given from Theorem 2 by

$$1 - \beta < 1 - \sqrt{\kappa\alpha}.$$

[2] Some parts of this section are implicitly discussed in Section 5 of [10].

Then, we have the condition:

$$\alpha < \beta^2/\kappa.$$

This is exactly the same bound as best known one for $h(y) \equiv 0 \pmod{p}$ [15].

Note that the algorithm requires more lattice dimension while it achieves the same bound as best known algorithm.

4.2 Optimality of Our Bound under a Reasonable Assumption

We'll discuss the optimality of our obtained bound. As shown above, the best known bound for $h(y) \equiv 0 \pmod{p}$ is May's bound: $\alpha < \beta^2/\kappa$, which is shown in [15]. While it is still an open problem whether this bound is optimal or not, we suppose that this bound is optimal in the following discussion.

We denote by $\mathcal{C}$ a set of integers such that $0 \in \mathcal{C}$. Assume that we have an algorithm which solves the problem: $xh(y) + C \equiv 0 \pmod{N}$ for any $C \in \mathcal{C}$ in polynomial time whenever

$$\gamma \leq 1 - l(\kappa\alpha)$$

for some function $l()$ satisfying $t < l(t) < \sqrt{t}$ for some t in adequate interval. This implies that it holds that $1 - \sqrt{\kappa\alpha} < 1 - l(\kappa\alpha) < 1 - \kappa\alpha$. The similar discussion in previous subsection concludes that we can solve the equation: $xh(y) = 0 \pmod{N}$ if $1-\beta \leq 1-l(\kappa\alpha)$. This leads that we can solve the equation: $h(y) \equiv 0 \pmod{p}$ whenever

$$\alpha \leq l^{-1}(\beta)/\kappa.$$

From the setting of function $l()$, we have $\beta^2/\kappa < l^{-1}(\beta)/\kappa$. Then, we can improve the May's bound. This contradicts to the optimality of May's bound. Then, the bound obtained by our algorithm is also optimal under the assumption that May's bound for Problem B is optimal.

This implies that if we could find an algorithm for solving $xh(y) + C = 0$ which achieves the better bound than $\gamma \leq 1 - \sqrt{\kappa\alpha}$, we can also improve the bound for $h(y) \equiv 0 \pmod{p}$.

4.3 Natural Bound for $xh(y) + C \equiv 0 \pmod{e}$

Howgrave-Graham [10] claimed that Boneh-Durfee's stronger bound [1] is *natural* if there exist a relation between $x(A+y) \equiv 1 \pmod{e}$ and $x(A+y) \equiv 0 \pmod{e}$. By following this idea, we show that our obtained bound: $\gamma \leq 1 - \sqrt{\kappa\alpha}$ in Theorem 2 is also natural.

Assume that there exists a polynomial time algorithm for solving Problem A which calls Problem A$^{(0)}$ as a subroutine. We consider an achievable bound for Problem A which uses the best known algorithm for Problem A$^{(0)}$. We call it a *natural bound*. We'll show that our obtained bound in Theorem 2 is natural.

We derive a condition that $xh(y) + C \equiv 0 \pmod{e}$. Let $X = e^\gamma$ and $Y = e^\alpha$. Then, we have $\beta = 1 - \gamma$. By using Eq. (5), the problem can be solved if $\alpha \leq \beta^2/\kappa = (1-\gamma)^2/\kappa$. This condition can be simplified into $\gamma \leq 1 - \sqrt{\kappa\alpha}$. This bound is exactly the same bound obtained in Section 3. Then, our obtained bound is natural.

5 Conclusions

We presented the algorithm for solving the bivariate modular equation: $xh(y) + C \equiv 0 \pmod{e}$ for a monic polynomial $h(y)$ with a degree κ. This result is an extension of Herrmann-May's result. Our algorithm solves the problem if $\gamma < 1 - \sqrt{\kappa\alpha}$ and $|C|$ is small enough. Our algorithm also solves the univariate unknown modular equation: $h(y) \equiv 0 \pmod{p}$, where p is unknown but its multiplier N is known. We proved that our algorithm achieves the same bound as the best known algorithm. Furthermore, we showed that our obtained bound: $\gamma \leq 1 - \sqrt{\kappa\alpha}$ is optimal under the assumption that the optimal bound for $h(y) \equiv 0 \pmod{p}$ is $\alpha < \beta^2/\kappa$. We showed that the obtained bound is also natural under the similar sense of Howgrave-Graham's discussion in CaLC2001.

Acknowledgement. The author was supported by KAKENHI 22700006.

Reference

1. Boneh, D., Durfee, G.: Cryptanalysis of RSA with private key d less than $n^{0.292}$. IEEE Transactions on Information Theory 46(4), 1339–1349 (2000)
2. Coppersmith, D.: Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 178–189. Springer, Heidelberg (1996)
3. Coppersmith, D.: Finding a Small Root of a Univariate Modular Equation. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 155–165. Springer, Heidelberg (1996)
4. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. J. Cryptology 10(4), 233–260 (1997)
5. Coron, J.-S., Mandal, A., Naccache, D., Tibouchi, M.: Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 487–504. Springer, Heidelberg (2011)
6. van Dijk, M., Gentry, C., Halevi, S., Vaikuntanathan, V.: Fully Homomorphic Encryption over the Integers. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 24–43. Springer, Heidelberg (2010)
7. Herrmann, M., May, A.: Attacking Power Generators Using Unravelled Linearization: When Do We Output Too Much? In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 487–504. Springer, Heidelberg (2009)
8. Herrmann, M., May, A.: Maximizing Small Root Bounds by Linearization and Applications to Small Secret Exponent RSA. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 53–69. Springer, Heidelberg (2010)
9. Howgrave-Graham, N.: Finding Small Roots of Univariate Modular Equations Revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
10. Howgrave-Graham, N.: Approximate Integer Common Divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001)
11. Kunihiro, N.: Solving Generalized Small Inverse Problems. In: Steinfeld, R., Hawkes, P. (eds.) ACISP 2010. LNCS, vol. 6168, pp. 248–263. Springer, Heidelberg (2010)
12. Kunihiro, N.: Solving generalized small inverse problems. IEICE Transactions E94-A(6), 1274–1284 (2011)

13. Kunihiro, N., Shinohara, N., Izu, T.: A Unified Framework for Small Secret Exponent Attack on RSA. In: Miri, A., Vaudenay, S. (eds.) SAC 2011. LNCS, vol. 7118, pp. 260–277. Springer, Heidelberg (2012)
14. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen 261, 515–534 (1982)
15. May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods. Ph.D. thesis, University of Paderborn (2003)
16. Rivest, R.L., Shamir, A., Adleman, L.M.: A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM 21(2), 120–126 (1978)
17. Wiener, M.J.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory 36(3), 553–558 (1990)

A Proofs

Proof of Lemma 6
The polynomial $\bar{g}_{[u,j,0]}$ is given by $\bar{g}_{[u,j,0]} = e^m x^u y^j$. Then, we have the lemma. $\square$

Proof of Lemma 7
Suppose that $\bar{g}_{[u-i,j,i]} \in \mathcal{G}$. Note the from the setting of $\mathcal{G}$, it holds that $0 \le i \le u$ and $0 \le j \le \kappa - 1$. The polynomial $\bar{g}_{[u-i,j,i]}$ is transformed into

$$\begin{aligned}\bar{g}_{[u-i,j,i]} &= e^{m-i}x^{u-i}y^j(xH(y)+z)^i = e^{m-i}x^{u-i}\sum_{t=0}^{i}\binom{i}{t}y^j z^{i-t}x^t H(y)^t \\ &= e^{m-i}x^{u-i}y^j z^i + e^{m-i}\sum_{t=1}^{i}\binom{i}{t}x^{u-i+t}y^j H(y)^t z^{i-t}.\end{aligned}$$

Then, we have

$$\bar{g}_{[u-i,j,i]} - e^{m-i}x^{u-i}y^j z^i \cong \sum_{t=1}^{i}\sum_{k=0}^{t(\kappa-1)} x^{u-i+t}y^{j+k}z^{i-t} \tag{6}$$

We focus on terms $x^{u-i+t}y^{j+k}z^{i-t}$ in right hand side of Eq. (6). Let $q := \lfloor (j+k)/\kappa \rfloor$ and $r := (j+k)\%\kappa (= j+k-q\kappa)$. Note that it holds that $u-i+t \ge q$ since $u \ge i$ and $t \ge q$ by the setting of $\mathcal{G}$. The latter inequality comes from the fact $q = \lfloor (j+k)/\kappa \rfloor = \lfloor (t+1)(\kappa-1)/\kappa \rfloor < t+1$. Then, each $x^{u-i+t}y^{j+k}z^{i-t}$ can be transformed into

$$\begin{aligned}x^{u-i+t}y^{j+k}z^{i-t} &= (xy^\kappa)^q y^{j+k-\kappa q}x^{u-i+t-q}z^{i-t} \\ &= x^{u-i+t-q}y^{j+k-\kappa q}z^{i-t}(z-C)^q \\ &\cong \sum_{l=0}^{q} x^{u-i+t-q}y^r z^{i-t+l}.\end{aligned}$$

It is easily verified that $u-i+t-q \ge 0$ and $0 \le r \le \kappa - 1$. The summation of the degree of x and that of z is given by $(u-i+t-q)+(i-t+l) = u-(q-l)$.

If $q < l$, then $u - (q - l) < u$ and all terms are in $\mathcal{M}(x^{u-i}y^j z^i)$. If $q = l$ and $t < q$, then $(u-i+t-q)+(i-t+l) = u$ and $u-i+t-q < u-i$. And then, all terms are in $\mathcal{M}(x^{u-i}y^j z^i)$. Finally, if $q = l = t$, then $u - i + t - q = u - i$ and $r < j$. And then, all terms are in $\mathcal{M}(x^{u-i}y^j z^i)$. Then, for any cases, we have

$$\mathcal{S}(\bar{g}_{[u-i,j,i]} - e^{m-i}x^{u-i}y^j z^i) \subseteq \mathcal{M}(x^{u-i}y^j z^i).$$

Then, we have the lemma. □

Proof of Lemma 8
Suppose that $\bar{g}_{[0,j,u]} \in \mathcal{H}$. The polynomial $\bar{g}_{[0,j,u]}$ is transformed into

$$\begin{aligned}\bar{g}_{[0,j,u]} = e^{m-u}y^j(xH(y)+z)^u &= e^{m-u}y^j \sum_{t=0}^{u}\binom{u}{t} z^{u-t}x^t H(y)^t \\ &= e^{m-u}y^j z^u + e^{m-u}\sum_{t=1}^{u}\binom{u}{t}x^t y^j H(y)^t z^{u-t}.\end{aligned}$$

Then, we have

$$\bar{g}_{[0,j,u]} - e^{m-u}y^j z^u \cong \sum_{t=1}^{u}\sum_{k=0}^{t(\kappa-1)} x^t y^{j+k} z^{u-t}. \tag{7}$$

We focus on terms $x^t y^{j+k} z^{u-t}$ in right hand side in Eq. (7). Let $q := \lfloor (j+k)/\kappa \rfloor$ and $r := (j+k)\%\kappa (= j+k-q\kappa)$. We separate each term $x^t y^{j+k} z^{u-t}$ into two cases:

1. $t \geq q$ and
2. $t < q$.

At the first case, each $x^t y^{j+k} z^{u-t}$ can be transformed into

$$\begin{aligned}x^t y^{j+k} z^{u-t} &= (xy^\kappa)^q x^{t-q} y^{j+k-\kappa q} z^{u-t} \\ &= x^{t-q}y^r z^{u-t}(z-C)^q \cong \sum_{l=0}^{q} x^{t-q}y^r z^{u-t+l}.\end{aligned}$$

It is easily verified that r holds that $0 \leq r \leq \kappa - 1$. The summation of the degree of x and that of z is given by $(t-q)+(u-t+l) = u-(q-l)$. Hence, all terms are in $\mathcal{M}(y^j z^u)$.

At the second case, each $x^t y^{j+k} z^{u-t}$ can be transformed into

$$\begin{aligned}x^t y^{j+k} z^{u-t} &= (xy^\kappa)^t y^{j+k-\kappa t} z^{u-t} \\ &= y^{j+k-\kappa t} z^{u-t}(z-C)^t \cong \sum_{l=0}^{t} y^{j+k-\kappa t} z^{u-t+l}.\end{aligned}$$

It is enough to prove that

$$\kappa - 1 + \tau(u-t+l) \geq j + k - \kappa t$$

for all $y^{j+k-\kappa t} z^{u-t+l}$. The summation of degree of x and degree of z is bounded by

$$\begin{aligned}(\kappa-1+\tau(u-t+l))-(j+k-\kappa t) &\geq \kappa-1+\tau(u-t)-(j+k-\kappa t)\\ &\geq \kappa-1+\tau(u-t)-(j+t(\kappa-1)-\kappa t)\\ &= \kappa-1+\tau(u-t)-j+t\\ &= \kappa-1+\tau u-j+t(1-\tau).\end{aligned}$$

Here, since $\kappa-1+\tau u \geq j$ and $\tau \leq 1$, we have $\kappa-1+\tau(u-t+l)-(j+k-\kappa t) \geq 0$. Hence, all terms are in $\mathcal{M}(y^j z^u)$.

Then, we have

$$\mathcal{S}(\bar{g}_{[0,j,u]} - e^{m-u} y^j z^u) \subseteq \mathcal{M}(y^j z^u).$$

Then, we have the lemma. □

B Small Example

We give a small example. Let $h(y) = y^2 + a_1 y + a_2$ and $m = 2, \tau = 1$. Note that $\kappa = 2$. From the definition of $\mathcal{G}$ and $\mathcal{H}$ described in Section 3, we have

$$\begin{aligned}\mathcal{G} &:= \{\bar{g}_{[u-i,j,i]} : u = 0,1,2; i = 0,\ldots,u; j = 0,1\} \text{ and}\\ \mathcal{H} &:= \{\bar{g}_{[0,j,u]} : u = 0,1,2; j = 2,\ldots,1+u\}\end{aligned}$$

and $\mathcal{F} = \mathcal{G} \bigcup \mathcal{H}$. We explicitly have a set of indexes as

$$\begin{aligned}&[0,0,0],[0,1,0],[1,0,0],[1,1,0],[0,1,1],[0,1,1],[0,2,1],[2,0,0],\\ &[2,1,0],[1,0,1],[1,1,1],[0,0,2],[0,1,2],[0,2,2],[0,3,2].\end{aligned}$$

Next, we show the expansion in compressed form. We show an explicit form of lattice basis matrix derived from $\mathcal{F}$. Due to the lack of space, we omit X, Y and Z. Note that since $\kappa = 2$, we substitute each occurrence xy^2 by the term $z - C$. The following corresponds to a coefficient matrix under the compressed form:

$$\begin{array}{c|cc||ccccc||cccccccc}
 & 1 & y & x & xy & z & yz & y^2z & x^2 & x^2y & xz & xyz & z^2 & yz^2 & y^2z^2 & y^3z^2 \\
\hline
\bar{g}_{[0,0,0]} & e^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[0,1,0]} & 0 & e^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\bar{g}_{[1,0,0]} & 0 & 0 & e^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[1,1,0]} & 0 & 0 & 0 & e^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[0,0,1]} & 0 & 0 & ea_2 & ea_1 & e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[0,1,1]} & -a_1eC & 0 & 0 & a_2e & a_1e & e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[0,2,1]} & -a_2eC & -a_1Ce & 0 & 0 & a_2e & a_1e & e & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
\bar{g}_{[2,0,0]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & e^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[2,1,0]} & 0 & 0 & 0 & -0 & 0 & 0 & 0 & 0 & e^2 & 0 & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[1,0,1]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & ea_2 & ea_1 & e & 0 & 0 & 0 & 0 & 0 \\
\bar{g}_{[1,1,1]} & 0 & 0 & -ea_1C & 0 & 0 & 0 & 0 & 0 & ea_2 & ea_1 & 1 & 0 & 0 & 0 & 0 \\
\bar{g}_{[0,0,2]} & 0 & 0 & A_1 & 0 & 0 & 0 & 0 & A_2 & A_3 & A_4 & A_5 & 1 & 0 & 0 & 0 \\
\bar{g}_{[0,1,2]} & 0 & 0 & -A_3C & A_1 & -A_5C & 0 & 0 & 0 & A_2 & A_3 & A_4 & A_5 & 1 & 0 & 0 \\
\bar{g}_{[0,2,2]} & -CA_1 & 0 & -A_2C & -A_3C & C_6 & -A_5C & 0 & 0 & 0 & A_2 & A_3 & A_4 & A_5 & 1 & 0 \\
\bar{g}_{[0,3,2]} & A_3C^2 & -CA_1 & 0 & -CA_2 & -2CA_3 & A_6 & -CA_5 & 0 & 0 & 0 & A_2 & A_3 & A_4 & A_5 & 1
\end{array},$$

where $A_1 = -a_1^2 C, A_2 = a_2^2, A_3 = 2a_1a_2, A_4 = a_1^2 + 2a_2, A_5 = 2a_1, A_6 = A_1 - A_4C$. Note that we can easily verify that the corresponding matrix is lower triangular.

Strong Authentication with Mobile Phone

Sanna Suoranta, André Andrade, and Tuomas Aura

Aalto University, Department of Computer Science and Engineering, Espoo, Finland

Abstract. As critical services and personal information are moving to the online world, password as the only user authentication method is no longer acceptable. The capacity of the human memory does not scale to the ever larger number of ever stronger passwords needed for these services. Single sign-on (SSO) systems help users cope with password fatigue, but SSO systems still mostly lack support for strong two-factor authentication. At the same time, the users have adopted mobile phones as personal digital assistants that are used both for accessing online services and for managing personal information. The phones increasingly include mobile trusted computing technology that can be used for hardware-based storage of user credentials. Thus, it is rather obvious that the mobile phones should be used as authentication tokens for critical online services.

In this paper, we show that existing open-source software platforms and commonly available mobile devices can be used to implement strong authentication for an SSO system. We use the Internet-enabled mobile phone as a secure token in a federated single sign-on environment. More specifically, we extend the Shibboleth SSO identity provider and build an authentication client based on a Nokia hardware security module. Our system design is modular, and both the SSO solution and the hardware-based security module in the phone can be replaced with other similar technologies. In comparison to most commercially available strong authentication services, our system is open in the sense that it does not depend on a specific credential issuer or identity provider. Thus, it can be deployed by any organization without signing contracts with or paying fees to a third party. No modifications need to be made to the client web browser or to the online service providers. We conclude that it is possible to implement strong personal authentication for an open-source SSO system with low start-up and operating costs and gradual deployment.

1 Introduction

The traditional way to authenticate users for services is passwords. Since typical user has accounts and passwords for about 25 services and types in eight passwords daily, the password quality is often poor [21], and the users suffer from password fatigue. Federated Identity Management (FIM) systems try to help the users and the service providers by offering them Single Sign-on (SSO) that allows users to log into several services with just one credential. Moreover, as critical services such as healthcare move to the online world, stronger authentication methods than passwords are needed. Passwords alone do not provide sufficient protection against spyware and phishing attack. Typically, stronger security is achieved with two-factor authentication where the user has a physical credential or a one-time code list in addition to the traditional password.

Another recent development is that users nowadays access online services from mobile devices, such as mobile phones or tablet computers, over near-ubiquitous wireless Internet connections. The device user interfaces limit the potential authentication methods, but the devices also provide new hardware-based methods including SIM cards and mobile trusted platform modules.

In this paper, we describe an architecture for federated identity environment that provides strong authentication to online services with the mobile phone as a secure token. Our goal is to prove that standard technology and open source building blocks can be used to implement a strong mobile authentication solution. We also want to show that trusted computing technology already available in many mobile phones can be used as part of two-factor authentication, and that it can be used together with existing single sign-on architectures through open interfaces. Our architecture is modular, and both the SSO solution and the hardware-based security module can be replaced with similar technologies. Furthermore, we wanted to show that an identity provider can independently replace the password authentication method in the FIM system with a strong authentication method. Thus, in contrast with many currently deployed systems, strong authentication can be achieved without paying fees to a third-party authentication provider.

2 Strong Authentication

Authentication and user information management is a big task for organizations that start to offer their services online. For this reason, many small service providers prefer to form or join a federations for authenticating their users [10]. In federated identity management (FIM), service providers (SP) outsource the user authentication to identity providers (IdP). Two commonly used open FIM systems are OpenID [31][34] and Shibboleth [24], which is based on the Security Assertion Markup Language (SAML) [32]. For example, Google acts as an identity provider for the OpenID community [30] and other services can use it to authenticate their users. In Finland, Shibboleth is used in higher-education institutions, which form the HAKA federation [14]. Big commercial services, on the other hand, seldom give their user databases for others to manage since knowledge about the users and their behavior is valuable information and because they do not trust to other IdPs to provide sufficient security [39].

The first requirement for strong authentication is that the identity provider must verify the true identity of the person. This step, called identity proofing, is the most expensive part of establishing a strong authentication system, and it can easily result in a monopoly for authentication services. Federated authentication can reduce the cost somewhat if multiple identity providers trust each other to perform the identity proofing. However, often the service providers trust the users to give correct information about themselves even though the information is not verified at all. In the global service markets, meeting the user and really verifying her identity from a trustworthy source such as a passport is often impossible for service providers that do not have customer service personnel in every country. Typically, the identity provider, including Google, only verifies that user is reachable at an email address, not who she really is.

It is often said that authentication is based on something the user knows, has, or is. The second requirement for strong authentication is that at least two of these methods are used together. For example, Bhargav-Spantzel et al. [9] have proposed a strong authentication system for FIM that combines the two factors of biometric authentication and some information known to the user, such as a social security number or credit card number. Although some notebook computers have a fingerprint reader and mobile phones have cameras for face recognition, the mobile devices needed for biometric authentication are not widely deployed or secure enough. Thus, strong authentication is usually based on a combination of something the user knows and something she has. Commonly, the latter is a purpose-build device, such as a smart card or the SecureID token [35], but recently general purpose identifiers such as RFID tags have gained favor to be used as physical tokens in two-factor authentication [33].

Another way to increase the strength of the authentication is to use multiple communication channels. For example, in Mizuno et al. [28] and in the protocols now used by many banks, a service provider sends an identifier to the user's mobile phone through the (relatively secure) phone network and the user sends it back through an insecure channel using a computer, which thus is authenticated as the client device. The system is targeted especially for untrusted computers, e.g. mobile computers or shared hosts in an Internet cafe. However, the transfer of information between the authentication device and the client computer requires either human input or secure pairing of the devices, both of which result in relatively poor usability of the secure tokens.

In Finland, strong authentication can be achieved using the Finnish Electronic Identification (FINEID) [20] that is a PKI based system where digital certificates are stored in a chip embedded in a personal identity card. The FINEID card costs 55 euros, is valid for five years, and requires a smart card reader. Most citizens find it unnecessary or too difficult to obtain one [43]. Another reason for the low adoption of FINEID is that there exist another, more ubiquitous method for strong authentication. The Finnish banks have long offered an authentication service (TUPAS) based on the online bank login [41]. The authentication is based on a password (number) and a one-time password list (on paper) sent to the customers by the bank. Any service provider may use the TUPAS authentication but it has to agree separately with each bank about using their authentication server. Thus, TUPAS is not a complete identity federation system, and many online services only support a few of the largest banks. The service providers pay a monthly fee and a small transaction fee (typically around 0.10-0.50 euros) for each user authentication [29]. The customers pay their share in the monthly service fee of their online bank account, which makes the authentication service appear as free to the users.

The above-mentioned methods provide strong authentication but they are not easy to use on mobile devices. In the next section, we describe strong authentication methods that are targeted for mobile devices.

3 Mobile Authentication

Mobile phone networks worldwide have at least 3.5 billion subscribers [43]. Mobile phone operators authenticate the subscriber with a small smart card called SIM or USIM, given to the user when the customer relationship is established. These are the most used cryptographic authentication method in the world. The authentication can be strong if the operator verifies reliably the customer identities (i.e. no anonymous pre-paid SIMs) and if the user is additionally required to input a PIN code to access the SIM. The card stores an International Mobile Subscriber Identity (IMSI), which is the customer identifier, and the cryptographic keys and algorithms that are needed for the authentication and key exchange.

The GSM and 3G authentication can be used in services of the Internet with the help of the Extensible Authentication Protocol (EAP) [4][5], which defines several methods for authentication. The EAP protocol itself is designed for network access authentication. Li et al. [26] present a first attempt to create an EAP based uniform access authentication framework, but it is targeted for network access, not for service access. The network access authentication could also be used for authentication to services: the Generic Authentication Architecture (GAA) [2] describes on the architectural level how pre-shared secrets needed in application-level authentication protocols can be derived from the AKA authentication, and the General Bootstrapping Architecture (GBA) [1] specifies how the shared secrets are actually created.

In the recent years, mobile phone operators worldwide have launched authentication services where other online service providers can use the phone network access authentication to check the customer identity. These methods are described below. None of them provides single sign-on as such, but they can be used as strong user authentication methods in federated identity management.

3.1 One-Time Passwords with Mobile Phone

The SMS-OTP method replaces a traditional paper OTP list with text messages sent to the mobile phone stored in the customer database using the Short Message Service (SMS). In the authentication process [28], the customer first types in her username and maybe also a password to the service. Then, she receives an SMS message with a one-time password which she has to type into the web-based service. Although the method is two-factor authentication, it is not very strong since the SMS messages are delivered in plaintext in GSM networks. Nonetheless, eavesdropping SMS is not very easy in the phone network air interface or in the phone network itself, and sniffing only the one-time password is not sufficient for login. The SMS-OTP method is easy to learn because most customers are used to SMS messages. For the service provider, SMS-OTP is easy to take into use because it does not require sending separate OTP lists to the customers, and the passwords are generated when they are needed. SMS-OTP does not require any changes to the SIM card or the mobile phone software.

The OTP lists can also be stored on the SIM card, or the SIM card can generate the passwords using the SIM Application Toolkit. Each service needs its own OTP list or generator since the service and the client must be synchronized, but this is no problem because the SIM card can contain several OTP generators. The service provider must distribute the OTP list or the generator to its clients. The phone network operator can usually do this over the air but, in the worst case, it the SIM card needs to be updated to a newer version. This method is more secure than sending cleartext SMSs since the one-time password does not travel unprotected over any network until the customer uses it in the authentication process. To protect the customers against theft of the mobile phone, the OTP generation can require a PIN code. The SIM-OTP method has been tested in a pilot case in Finland for logging into an online bank, but it will not be used in the future because a more secure method is available [23].

3.2 Mobile Certificates

Mobile certificate is also based on the SIM card, which must be updated because, cryptographic keys and other information such as the user's official name, citizenship, and electrical identification number need to be stored in the card [19]. The user can get the new SIM card after the mobile phone network operator has verified her identity either face-to-face or with other strong authentication methods. In the authentication process, the user must type in her PIN code before the authentication challenge is digitally signed with the SIM card. The mobile certificate is based on standardization work of ETSI but, to achieve interoperability, industry bodies like the Finnish mobile phone operator association FiCom publish detailed instructions for applying the standards [18]. Currently, the mobile certificates work locally between operators in the same country but global roaming for using the certificates in another country has not been implemented. The costs of the authentication differs from country to country. Recently, one mobile operator in Finland launched a mobile certificate service, which costs for the customers 1,9 euros monthly and 0,07 euros for each authentication event [15]. The cost is higher for the users than in the bank-based TUPAS authentication. The fees for the service providers have not been published and the system is not yet widely deployed. Since the network operators see the mobile certificate authentication as a money-making business, they are not keen to integrate it to single sign-on or federated authentication, although such implementations have not been banned either.

All the SIM-based authentication methods described above can be strong if they require the user to input a PIN or password in addition to having the SIM card, and if the mobile phone operator performs strong identity proofing of its customers. The disadvantage of these methods is that they depend on the mobile network operator as a trusted entity that is involved in every transaction. This gives the network operator the opportunity to charge a fee for each transaction. Roaming agreements between network operators may eventually result in global federation of the authentication but, so far, interoperability between operators has been limited.

3.3 Trusted Computing Software and Hardware

The development of the SIM card based methods has taken long time and the operator-dependent business models have slowed down their adoption. Meanwhile, also other mobile-phone-based methods have been investigated. For example, SecureID tokens are available as software to many mobile phones [36]. Such software is usually not considered secure unless it is run in a secure hardware environment. The first hardware-based security modules in mobile phones were developed for digital rights management [27]. Later, they have been extended to be general-purpose trusted-computing platforms. For example, the Trusted Platform Module (TPM) [42] specifies the features required for a trusted computing platform, and M-Shield [8] is a secure hardware and software component that is used in many mobile phones.

Ekberg et al. [16][17][25] have proposed a trusted platform architecture called On-board Credentials (ObC). The architecture of ObC is depicted in Figure 1. All communication with the secure hardware goes through the Credential Manager [16]. An interpreter allows small pieces of software be run in a secure environment and a there is a cryptographic library including e.g. RSA encryption and digital-signature algorithms.There are also two keys: the symmetric ObC platform key (OPK), which is the master key of the device, and the asymmetric device secret key SK_D [17]. The device manufacturer or some other trusted third party certifies the corresponding public key PK_D. Since the memory of the trusted hardware is very small, everything that is not used at the moment is encrypted and stored in the ObC database.

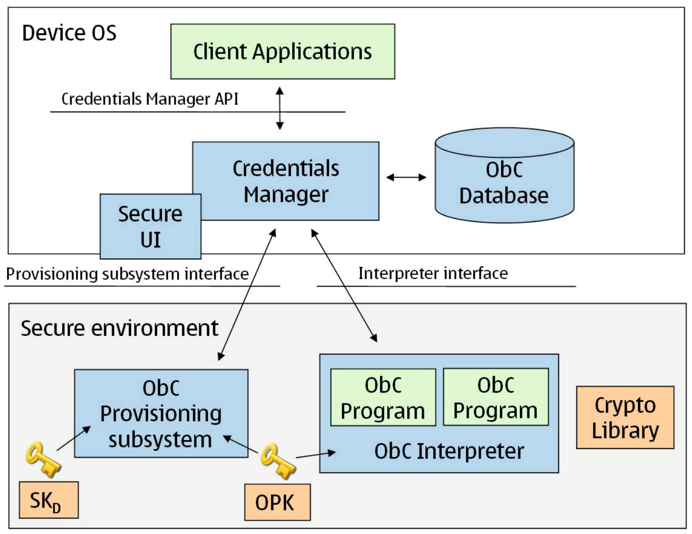


Fig. 1. Architecture of the On-board Credentials[17]

Because anyone can create pieces of software to be run in the secure environment, these programs must be isolated from each other. In the process called *provisioning*, a provisioner creates a symmetric key (RK) that is encrypted with the target-device public key PK_D. The provisioner can authorize other secure software to access its data in another process called *endorsement*. Unlike the authentication models described earlier in this paper, the ObC model is open in the sense that anyone can provision new credentials and endorse others to use them as building blocks for new credentials. Since OPK and SK_D are device dependent, separate procedures have been defined for *migration* of the credentials and keys to a new device [37]. ObC is a flexible platform for secure credential management, but it is not yet widely deployed. Although it is open, it has not been standardized and the APIs are not fully documented. Nevertheless, we find the idea of a generic, programmable hardware security module on a personal mobile device to be a compelling one, and build our architecture on the assumption that most phones will eventually have one, even if based on slightly different technologies.

3.4 Federated Identity Management with Mobile Phone

Mobile devices have developed in a direction where their usage does not differ from laptop computers. This creates an opportunity to combine the phone-based strong authentication methods with FIM systems. Abe et al. [3] have implemented an identity provider on a mobile phone. They use the USIM card to sign a SAML assertion after the user has entered her PIN code into the mobile phone. Placing the IdP on the mobile phone is a clever idea, but there are some of technical and business problems. The SP has to find the IdP to whom it sends the authentication request. Furthermore, mobile network operators often want to provide the authentication as a service for which they can charge users.

Mobile phones can also be used as clients for services. Because of the limited user interface on the phones, it would be desirable to migrate application sessions to the phone only when the user is mobile. Aoyagi et al. [7] have developed a prototype that allows single sign-on using multiple terminals with different network connections. They use two-dimensional bar code readers in the mobile phone to bind the authentication of the devices to each other. This only works with mobile phones that have a camera. Another solution is to migrate the authentication session from one device to another one. In Shibboleth, this can be done with moving the session cookies between the devices [40].

Both in OpenID and in Shibboleth, the IdP can implement different authentication methods and service providers may request a certain method to be used. Although usually only a password-based method is used, it is possible to integrate strong two-factor authentication even to OpenID. For example, mobile phone operators in Estonia provide strong authentication by acting as OpenID identity providers [22]. In this work, we have modified Shibboleth to use mobile-phone-based strong authentication. In the next section, we describe the building blocks that are needed for our architecture.

4 Building Blocks for Strong and Mobile Identity Federation

In order to build an architecture for strong and mobile authentication in a federated environment, we need a FIM system and trusted computing technology that works on a mobile phone. Our first criterion was that the solutions are already in use and that they are based on open source software. In this section, we explain our choices for the building blocks for strong and mobile identity federation.

We chose Shibboleth to be the FIM system because it is used in the HAKA federation. OpenID is not widely used in Finland since other stronger authentication methods such as TUPAS are available. The second reason for our choice is that the installation packages, source code, configuration files and other information are available on the Shibboleth wiki pages [38]. In addition, the use of the HAKA federation is not limited to Finnish academic institutions. The federation is itself part of a larger Nordic one, the Kalmar e-identity Union [12]. Shibboleth typically uses password authentication. However, the HAKA federation is prepared to add other authentication methods: the IdP can inform the service provider about the used authentication method in SAML attributes defined in the FunetEduPerson schema [13] and, in the future, the service provider can also request a specific method.

Another essential building block for strong mobile authentication is the hardware-based, tamper-proof security module that is used in the mobile phone. There are not many open source solutions of trusted computing technology that work on mobile phones and that can be used freely. We chose to use ObC as an example of a trusted computing module. We used N900 phone with ObC emulator. The reason for choosing this phone is that N900 runs Maemo, an open-source smart phone operating system. It implements the Qt framework and provides the necessary cryptographic components [6]. Similar mobile trusted computing technology is already available in some commercial smart phones and it is expected to become widely available, even though the technology has not yet received much public attention.

Based on these building blocks, we designed a protocol that allows the user to authenticate herself to a service in a federated environment. When combined with strong identity proofing, the authentication is strong because it is based on something known, a password, and something physical, a mobile phone storing user credentials in the hardware security module. Our protocol is designed in such a way that the FIM system or the hardware-security component could be replaced without major changes to the architecture. Next, we describe the system design and implementation.

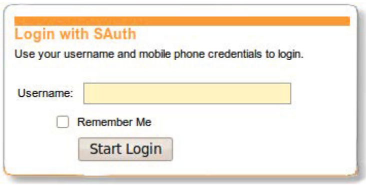
5 Design and Implementation

We consider systems where the user has two devices: a computer with a web browser, which is used for accessing the service, and a mobile phone, which is only used for authentication. Naturally, a multitasking mobile phone can be used for both purposes at the same time. We first describe and illustrate the user experience of the system through screenshots from our functional prototype implementation. Later in Section 5.3, the implementation architecture will be described.

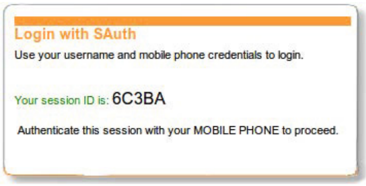
5.1 Authentication User Experience

The user accesses protected resources in a Service Provider (SP) with his web browser. The SP redirects the connection to an IdP. Optionally, the user may be allowed to choose from multiple identity providers on a Where Are You From (WAYF) page. Then, the browser shows the login form (Figure 2a) where the user types in his username and clicks the Start Login button. As a result, the browser shows a short session identifier to the user (Figure 2b) and the authentication session in the IdP is created.

The user then starts on her mobile phone the authentication client, which stores user credentials with the help of the hardware security module on the phone. The authentication client connects to a preconfigured IdP with a preconfigured username. If the user has multiple IdPs or usernames, she is prompted to select from a list (Figure 3a). Then the authentication client shows the user a list of session identifiers and the user needs to choose the one also shown in the browser (Figure 3b). The other session identifiers in the list may correspond to sessions initiated by attackers or to parallel authentication sessions initiated by the same user on different browsers, or they may be random values. The user always has to choose from multiple identifiers so that she cannot develop a habit of automatically clicking Ok.



(a) Login page in the browser


Login with SAuth

Use your username and mobile phone credentials to login.

Your session ID is: 6C3BA

Authenticate this session with your MOBILE PHONE to proceed.

(b) Result page with the session ID

Fig. 2. Web browser user interface

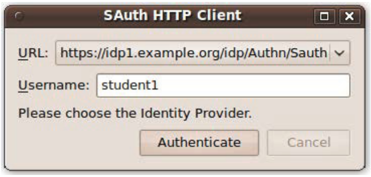
After the user has confirmed the session identifier, she still has to enter the PIN code for accessing her credential in the secure hardware module on the phone (Figure 3c). The rest of the authentication process takes place automatically between the authentication client on the phone and the IdP. If the authentication is successful (Figure 3d), the web browser is granted access to the service. The session management of the SSO system redirects the browser to the application page.

5.2 Key Design Decisions

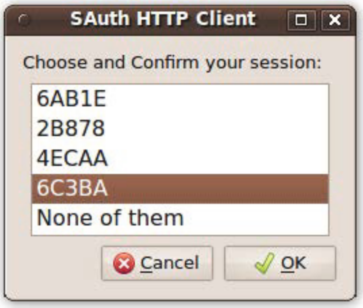
Three important design decision were made in the authentication system to make it easier to deploy and more usable in practical systems. The first design decision was to build the solution with software on platforms that are open and free to use. In particular, we did not want the solution to depend on the mobile operators or to be tied into any other commercial authentication service. Also, the authentication system should not be limited to any specific geographical region or specific mobile networks. These criteria exclude the use of the otherwise convenient SIM-based authentication methods. It is also critical that the SSO system administrators, such as the university IT services, can continue to enroll their own users as before without depending on an external certification or identity proofing services.

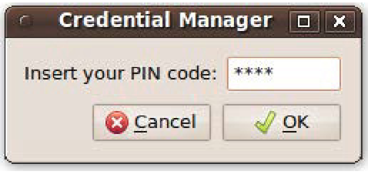
The second design choice is that the user initiates both the browser session and the authentication process. That is, the user has to manually start the authentication client on the phone rather than being prompted automatically to accept the authentication event. Event-based implementations would typically require a multitasking operating system for listening to events on the background and energy-consuming keep-alive messages. Our user-started application can be easily ported to any mobile phone platform that has a hardware security module and a data connection. The authentication client could naturally be launched automatically on phones that support some kind of event notifications, but we did not want to depend on that.

The third important design decision is that there is no direct communication between the web browser that is used for accessing the service and the mobile phone. The communication takes place via the IdP, with the help of the user who has to confirm the session identifier. This makes our solution independent of the web browser implementation and avoids the need for the user to set up a direct communication link, such as a Bluetooth association, between the two devices. Again, our design choices make the solution easy to use and to deploy on a broad range of platforms. Next, we describe the architectural components and the protocol used in the communication.

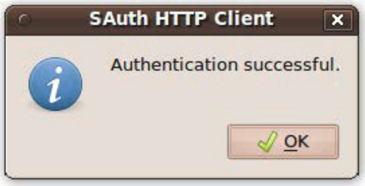

SAuth HTTP Client
URL: https://idp1.example.org/idp/Authn/Sauth
Username: student1
Please choose the Identity Provider.
Authenticate
Cancel

(a) Authentication client starting page





(b) Confirm session alias (c) The PIN query



(d) Authentication success

Fig. 3. Mobile phone user interface

5.3 Architecture and Protocol

The system architecture and *SAuth* protocol that connects its parts are depicted in Figure 4. Andrade [6] gives an even more detailed description. SAuth is a strong authentication method for single sign-on systems that uses the mobile phone as the security token. Our implementation was written for the *Shibboleth* single sign-on system [11] and it uses virtual credentials stored in the mobile phone with the *On-board Credentials* (ObC) [16] secure hardware platform. Either one of these components could be replaced with similar technology without major changes to the architecture.

The service client is assumed to be a browser on a PC, although it could also run on the same mobile phone that is used as the secure token. The client browser does not require any modifications. The service provider (typically a web server) is left out from the architecture diagram because it only forwards the connection to the IdP in the beginning and grants access to the service after successful authentication, as always in web SSO systems. The essential modifications were done to the Shibboleth IdP. An extension called Authentication Service (AuS) is added to the IdP to handle the user authentication with another device than the one accessing the service. Two software

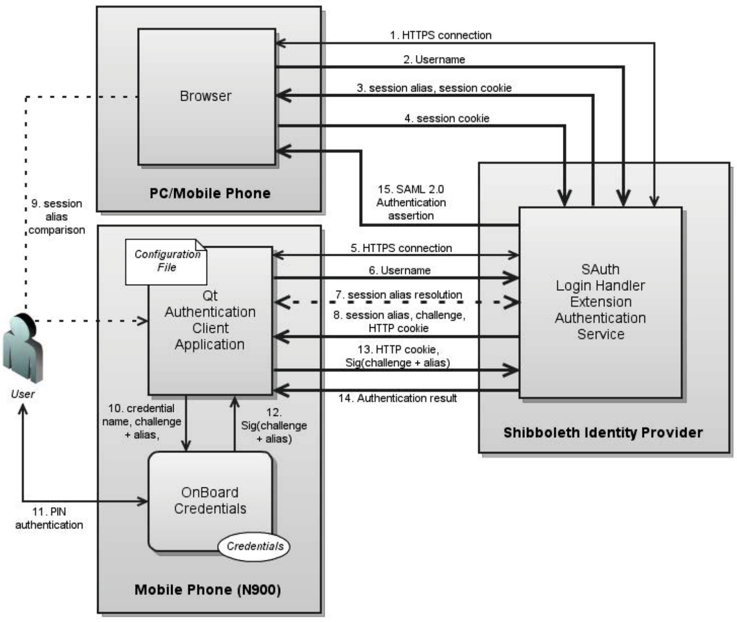

1. HTTPS connection
2. Username
3. session alias, session cookie
4. session cookie
Browser
PC/Mobile Phone
15. SAML 2.0 Authentication assertion
9. session alias comparison
Configuration File
Qt Authentication Client Application
5. HTTPS connection
6. Username
7. session alias resolution
8. session alias, challenge, HTTP cookie
13. HTTP cookie, Sig(challenge + alias)
14. Authentication result
SAuth Login Handler Extension Authentication Service
Shibboleth Identity Provider
User
10. credential name, challenge + alias,
12. Sig(challenge + alias)
OnBoard Credentials
Credentials
11. PIN authentication
Mobile Phone (N900)

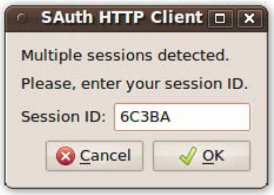
Fig. 4. SAuth implementation overview [6]

components were developed for the mobile phone: a Qt authentication client that handles the communication with the AuS and an ObC module that is executed within the secure environment and takes care of the user credentials.

When the user initiates the authentication process in the web browser, the AuS module on the IdP creates a session identifier and stores the session information in its database. The user then starts the authentication client on the mobile phone, which connects to the IdP and sends the username to it. Based on the username, the AuS finds the pending authentication sessions from the database and sends their session identifiers to the authentication client. After the user has chosen one session identifier, a simple public-key-based challenge-response protocol takes place between the AuS and the hardware security module on the phone. When the IdP has verified the user signature, it redirects the web browser to the actual service. Since the communication between the mobile phone and the IdP takes place over the Internet, the phone must have a data connection. The connections between the client browser and IdP, and between the IdP and mobile phone, are secured with TLS. While the authentication process takes place on the phone, the web browser polls for its completion with Javascript. As a standard backup, if the browser does not support Javascript, the user will have to click on a link on the IdP web page after completing the authentication on the phone.

5.4 Preventing Man-in-the-Middle Attacks

In order to guarantee that the same user controls both devices during the authentication, the user must compare the session aliases shown to her in the web browser and in the



Multiple sessions detected.
Please, enter your session ID.
Session ID: 6C3BA

Fig. 5. Session alias input

mobile phone. The AuS creates a unique session identifier (alias) for each username that starts authentication on a browser as shown above in Figure 3b. If there is more than one session associated with the same username, it could indicate an attacker trying to gain access. Since multiple identifiers are always shown to the user, even a careless user is unlikely to accept one that corresponds to the attacker session.

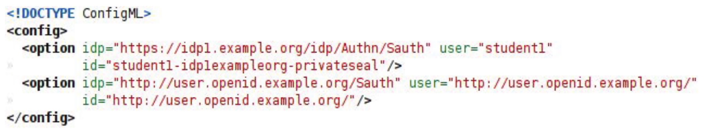
However, we implemented a design alternative for the cases when the IdP detects multiple authentication sessions for the same user: the AuS may request the user to type in on the mobile phone the session alias, as shown in Figure 5. This reduces further the likelihood that a careless user will accept the wrong session identifier. It also solves denial-of-servie (DoS) problems in situations where an attacker has opened a large number of incomplete authentication sessions. In that case, the identifiers do not need to be sent to phone or shown to the user. Instead, the user is asked to type in the correct session alias. We have left this mechanism as a configuration option in the AuS.

5.5 Mobile Phone Hardware Security Module

The actual identification of the user is achieved by a challenge-response authentication protocol with digital signatures created on the mobile phone. The private signature key is stored securely using the ObC hardware-based security module which guarantees that the key never leaves the secure environment unsealed [16]. While the hardware security module could also store passwords and symmetric keys, digital signature schemes offer optimal levels of authentication, integrity and non-repudiation while strongly resisting forgery. Public key cryptography is now feasible on most mobile devices, and hardware security (mobile trusted computing) technology with public-key support is becoming available on consumer phones. Furthermore, the credentials are protected from misuse by the PIN-based access control. Any operation on these credentials has to be authorized with the correct PIN. This helps protect the user credentials if the phone is lost or stolen. In our implementation, the user has at most three attempts to get the PIN right or, otherwise, the credentials will be blocked and the user will not be able to authenticate.

Because mobile phones have only small keyboards that are not nice for typing in a lot of information, we store the IdP address and the username in an XML configuration file on the phone. An example is shown in Figure 6. The **idp** field is the URL of the IdP with the SAuth authentication module. The **user** field contains the username. The **id** field identifies the user private key in the ObC credential manager.

The user public key naturally needs to be registered with the IdP. In our implementation, the enrollment and updating of the keys is done manually in a way similar to



```
<!DOCTYPE ConfigML>
<config>
  <option idp="https://idp1.example.org/idp/Authn/Sauth" user="student1"
          id="student1-idp1exampleorg-privateseal"/>
  <option idp="http://user.openid.example.org/Sauth" user="http://user.openid.example.org/"
          id="http://user.openid.example.org/"/>
</config>
```

Fig. 6. Example configuration file

how passwords are managed in most SSO systems. A password-based online enrollment protocol could speed up the deployment of the strong authentication method in a system where users already have passwords. However, bootstrapping with a weaker authentication method is not acceptable if the goal is strong personal authentication, and manual user registration with effective identify proofing is required. Alternatively, the authentication could be based on public-key certificates issued by a third party or on some other commercially available online authentication method.

6 Discussion

The design of our system is generic in the sense that it is not exclusive to a specific SSO or trusted-computing technology. This applies to the protocol as well as to the implementation itself. Both can be easily replaced with other similar technologies. For instance, our implementation uses Shibboleth as the base SSO system but it could be easily ported to another system such as OpenID because the AuS is built as a Java servlet extension, which is fairly independent from the rest of the SSO system. The mobile authentication client, on the other hand, was designed with a simple interface that allows other credential managers similar to ObC to be used. This modularity enhances the availability of the solution as well as eases its integration with other technologies.

If the IdP has a contract with a mobile operator, even SIM-card-based authentication could be offered as an alternative to the users who have SIMs from co-operating mobile operators. For most online services, however, it would be impossible to restrict the user base to the customers of specific mobile operators. Our architecture works equally well with foreign or anonymous prepaid SIM cards, which are popular in some countries.

The security level of our solution varies depending on the chosen trusted computing technology and credential provisioning process. ObC is considered to be tamper-proof. If an off-the-shelf Android phone with a software-based security module is used instead of the hardware-based one, the security level is lower because mobile phones can be hacked. However, the security module can be changed to whatever is available, and even the software modules are better than nothing. Moreover, the security of the credential provisioning depends on the processes of the IdP, which takes care of the initial identity proofing and credential distribution.

We have chosen not to use a direct network connection between the client computer and the mobile phone. The user initiates both the browser and authentication-client connections to the IdP and compares the session aliases, thus proving that she has both devices. This way, the user is in charge of the authentication process and no attacker can gain access to the service. Moreover, the user interface design prevents a careless user from blindly accepting incoming authentication requests. A more typical design would be to couple the devices together with a local-area radio link, e.g. Bluetooth, or some other other local communication channel, such as the camera in the mobile phone. We feel that setting up such connections is tedious to the user, prone to failure, and puts unnecessary restrictions of the types of devices used. On the other hand, our solution does not work if the mobile phone is offline. Moreover, even though usability of pairing mechanisms have been tested and our solution works in somewhat similar way, usability testing would be interesting since the beginning of the authentication process is different in our system.

Our system can be taken into use incrementally. All users of an IdP do not need to have their identity verified strongly in the start. Information about the identity proofing and authentication methods and their strength can be delivered to the online services together with the other attributes e.g. in the SAML messages. The gradual deployment of the authentication method allows the SP to move to requiring strong authentication when most of its user identities have been verified strongly. Apart from having a smart phone with a hardware security feature, there is no need for major one-time investments to hardware on the user side or for operator contracts on the SP side. Even the hardware security module on the phone is not absolutely necessary if the SPs are satisfied with the lower security of software-based credentials. Only the IdP has to update its software to provide strong authentication in addition to traditional password authentication.

7 Conclusions

We have designed and implemented strong, two-factor authentication with a mobile phone as security token for a federated single sign-on environment. The architecture is based on open source components. It is modular and allows replacing the SSO system and the hardware security module with other similar technologies. Using the system for user authentication does not require changes to client web browsers or service-provider applications. The identity provider can take the system into use incrementally, on side of weaker authentication methods such as passwords. The user takes responsibility for comparing session identifiers and thus linking the service session and authentication to each other, which avoids the need for a direct communication link between the browser and the phone. Altogether, the results of this project show that it is possible to implement strong personal authentication for an open-source SSO system with low start-up and operating costs and gradual deployment to a broad range of devices and browsers.

References

1. 3GPP. Generic bootstrapping architecture (GBA). Specification TS 33.220 v. 10.0.0, 3GPP (October 2010), http://www.3gpp.org/ftp/Specs/html-info/33220.html
2. 3GPP. Generic authentication architecture (GAA); system description. Specification TR 33.919 v. 10.0.0, 3GPP (March 2011)
3. Abe, T., Itoh, H., Takahashi, K.: Implementing identity provider on mobile phone. In: The 2007 ACM Workshop on Digital Identity Management, DIM 2007. ACM (November 2007)
4. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (Proposed Standard), Updated by RFC 5247 (June 2004)

5. Aboba, B., Simon, D., Eronen, P.: Extensible Authentication Protocol (EAP) Key Management Framework. RFC 5247 (Proposed Standard) (August 2008)
6. Andrade, A.: Strong Mobile Authentication in Single Sign-On Systems. Master's thesis, Aalto University School of Science (May 2011)
7. Aoyagi, M., Abe, T., Takahashi, K.: Symmetric identity federation for fixed-mobile convergence. In: Proceedings of the 4th ACM Workshop on Digital Identity Management, pp. 33–40 (October 2008)
8. Azema, J., Fayad, G.: M-Shield mobile security technology: making wireless secure. Texas Instruments, White paper (2008), http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf (referred July 4, 2011)
9. Bhargav-Spantzel, A., Squicciarini, A., Bertino, E.: Privacy preserving multi-factor authentication with biometrics. In: DIM 2006: Proceedings of the Second ACM Workshop on Digital Identity Management. ACM (2006)
10. Bhatti, R., Bertino, E., Ghafoor, A.: An integrated approach to federated identity and privilege management in open systems. Communications of the ACM 50(2) (February 2007)
11. Carmody, S., Erdos, M., Hazelton, K., Hoehn, W., Morgan, R.B., Scavo, T., Wasley, D.: Shibboleth Architecture: Protocols and Profiles. Technical report, Internet2 Middleware Initiative (September 2005)
12. CSC - IT Center for Science. Kalmar e-identity union linking nordic research networks (2011), http://www.csc.fi/english/csc/publications/cscnews/2010/4/kalmar/ (referred July 5, 2011)
13. CSC - IT Center for Science. Funeteduperson schema (2011), http://www.csc.fi/english/institutions/haka/definitions/funeteduperson/ (referred July 5, 2011)
14. CSC - IT Center for Science. Haka federation (2011), http://www.csc.fi/english/institutions/haka (referred July 5, 2011)
15. DNA mobile network operator. DNA mobile certificate (2011), http://www.dna.fi/yksityisille/puhe/palvelut/Sivut/DNAMobiilivarmenne.aspx (ref. July 4, 2011)
16. Ekberg, J.-E., Asokan, N., Kostiainen, K., Eronen, P., Rantala, A., Sharma, A.: Onboard credentials platform design and implementation. Technical Report NRC-TR-2008-001, Nokia Research Center (2008)
17. Ekberg, J.-E., Asokan, N., Kostiainen, K., Rantala, A.: On-board credentials with open provisioning. Technical Report NRC-TR-2008-007, Nokia Research Center (2008)
18. FiCom Ry. FiCom published application instructions for mobile certificate standard (May 25, 2005) (in Finnish), http://www.ficom.fi/ajankohtaista/ajankohtaista_1_1.html?Id=1117009845.html (referred July 4, 2011)
19. FiCom Ry. Mobile certificate makes identification simpler (2008) (in Finnish), http://www.ficom.fi/tietoa/tietoa_5_3.html (rererred July 1, 2011)
20. Finnish Population Register Centre. Fineid citizen certificate (2011), http://fineid.fi (referred July 1, 2011)
21. Florêncio, D., Herley, C.: A largescale study of web password habits. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007 (2007)
22. Ideelabor. OpenID in Estonia (2008), http://openiddirectory.com/openid-providers-c-1.html (referred February 27, 2009)
23. Information society advisory board for creating electrical authentication. Mobile authentication methods, description and comparison (November 13, 2008) (in Finnish), http://www.arjen-tietoyhteiskunta.fi/files/185/mobiilitunnistamismenetelmat.pdf
24. Internet2. Shibboleth (2006), http://shibboleth.internet2.edu/ (referred September 5, 2006)

25. Kostiainen, K., Ekberg, J.-E., Asokan, N., Rantala, A.: On-board credentials with open provisioning. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS 2009 (2009)
26. Li, X., Ma, J., Park, Y., Xu, L.: A usim-based uniform access authentication framework in mobile communication. EURASIP Journal on Wireless Communications and Networking - Special Issue on Security and Resilience for Smart Devices and Applications (January 2011)
27. Messerges, T.S., Dabbish, E.A.: Digital rights management in a 3g mobile phone and beyond. In: DRM 2003: Proceedings of the 3rd ACM Workshop on Digital Rights Management, pp. 27–38. ACM, New York (2003)
28. Mizuno, S., Yamada, K., Takahashi, K.: Authentication using multiple communication channels. In: ACM Workshop on Digital Identity Management (2005)
29. NorthID. Operations models for electrical authentication and identity management. Workshop material (September 15, 2009) (in Finnish)
30. OpenID.net. Get an OpenID (2006), http://openid.net/get-an-openid/ (referred July 1, 2011)
31. OpenID.net. Openid.net website (2008), http://openid.net/ (referred December 30, 2008)
32. Ragouzis, N., Hughes, J., Philpott, R., Maler, E., Madsen, P., Scavo, T.: Security assertion markup language (SAML) v2.0 technical overview. Technical report, OASIS (March 25, 2008)
33. Rahnama, B., Elci, A., Celik, S.: Securing rfid-based authentication systems using parsekey+. In: SIN 2010: Proceedings of the 3rd International Conference on Security of Information and Networks (September 2010)
34. Recordon, D., Reed, D.: OpenID 2.0: A platform for user-centric identity management. In: ACM Workshop on Digital Identity Management, DIM (2006)
35. RSA Security. RSA SecureID (2009), http://www.rsa.com/node.aspx?id=1156 (referred July 4, 2010)
36. RSA Security. Software authenticators (2011), http://www.rsa.com/node.aspx?id=1313 (referred July 4, 2011)
37. Sharma, A.K.: Onboard credentials: Hardware assisted secure storage of credentials. Master's thesis, Helsinki University of Technology (2007)
38. Shibboleth. Wiki (2011), https://wiki.shibboleth.net/ (referred July 5, 2011)
39. Sun, S.-T., Hawkey, K., Beznosov, K.: OpenIDemail enabled browser: Towards fixing the broken web single sign-on triangle. In: DIM 2010: Proceedigns of the 6th ACM Workshop on Digital Identity Management, October 8. ACM (2010)
40. Suoranta, S., Heikkinen, J., Silvekoski, P.: Authentication Session Migration. In: Aura, T., Järvinen, K., Nyberg, K. (eds.) NordSec 2010. LNCS, vol. 7127, pp. 17–32. Springer, Heidelberg (2012)
41. The Finnish Bankers' Association. Banks' tupas certification service for service providers (October 2005), http://www.pankkiyhdistys.fi/sisalto/upload/pdf/tupasV21eng.pdf (ref. September 8, 2006)
42. Trusted Computing Group. Trusted platform module (2011), http://www.trustedcomputinggroup.org/developers/trusted_platform_module/ (referred July 4, 2011)
43. Virtanen, M.: Mobile electronic id. Master's thesis, Aalto University School of Science and Technology (2010)

Measuring SSL Indicators on Mobile Browsers: Extended Life, or End of the Road?

Chaitrali Amrutkar[1], Patrick Traynor[1], and Paul C. van Oorschot[2]

[1] Georgia Tech Information Security Center (GTISC), Georgia Institute of Technology
{chaitrali,traynor.cc}@gatech.edu,
[2] School of Computer Science, Carleton University, Ottawa
paulv@scs.carleton.ca

Abstract. Mobile browsers are increasingly being relied upon to perform security sensitive operations. Like their desktop counterparts, these applications can enable SSL/TLS to provide strong security guarantees for communications over the web. However, the drastic reduction in screen size and the accompanying reorganization of screen real estate significantly changes the use and consistency of the security indicators and certificate information that alert users of site identity and the presence of strong cryptographic algorithms. In this paper, we perform the first measurement of the state of critical security indicators in mobile browsers. We evaluate ten mobile and two tablet browsers, representing over 90% of the market share, using the recommended guidelines for web user interface to convey security set forth by the World Wide Web Consortium (W3C). While desktop browsers follow the majority of guidelines, our analysis shows that mobile browsers fall significantly short. We also observe notable inconsistencies across mobile browsers when such mechanisms actually are implemented. Finally, we use this evidence to argue that the combination of reduced screen space and an independent selection of security indicators not only make it difficult for experts to determine the security standing of mobile browsers, but actually make mobile browsing more dangerous for average users as they provide a false sense of security.

1 Introduction

Mobile browsers provide a rich set of features that often rival their desktop counterparts. From support for Javascript and access to location information to the ability for third-party applications to render content through WebViews, browsers are beginning to serve as one of the critical enablers of modern mobile computing. Such functionality, in combination with the near universal implementation of strong cryptographic tools including SSL/TLS, allows users to become increasingly reliant upon mobile devices to enable sensitive personal, social and financial exchanges.

In spite of the availability of SSL/TLS, mobile users are regularly becoming the target of malicious behavior. A 2011 report indicates that mobile users are three times more likely to access phishing websites than desktop users [18]. Security indicators (i.e., certificate information, lock icons, cipher selection, etc.) in web browsers offer one of the few defenses against such attacks. A user can view different security indicators and related certificate information presented by the browser to offer signals or clues about the credibility of a website. Although mobile and tablet browsers appear to support similar security indicators when compared to desktop browsers, the reasons behind the increasing number of attacks on mobile browsers are not immediately clear.

In this paper, we perform the first comprehensive empirical evaluation of security indicators in mobile web browsers. The goal of this work is *not* to determine if average users take advantage of such cues, but instead to demonstrate that such indicators are lacking and thus fail to provide sufficient information for even experts. We believe that this distinction is critical because it highlights areas where not even the best trained users will be able to differentiate between malicious and benign behavior. Rather than an ad hoc analysis, we base our study on the recommendations set forward by the W3C for user interface security [9] as a proxy for best practices. In particular, we systematically measure which browsers strictly conform to the absolute requirements and prohibitions of this document. We perform our analysis across ten mobile and two tablet browsers, representing greater than 90% of the mobile market share [16], and then compare our results against the five most popular desktop browsers.

Our study makes the following contributions:

– **Widespread failure to implement recommended security indicators in mobile browsers:** We perform the first *systematic* and *comprehensive* comparison between mobile, tablet and traditional browsers based on the best practices set forth in the W3C guidelines [9]. Whereas desktop browsers largely conform to these guidelines, mobile and tablet browsers fail to do so in numerous instances. We believe that this makes even expert users subject to attacks including an undetectable man-in-the-middle.
– **Mobile security indicators are implemented inconsistently across browsers:** Our study observed tremendous inconsistency in the presentation and availability of such indicators in mobile and tablet browsers, in contrast to traditional desktop browsers. Accordingly, many of the clues experts instruct average users to look for can no longer reliably be found on these platforms.
– **Newest security indicator is largely absent:** Extended Validation (EV) SSL indicators and certificates [8,27,34] were designed to improve assurance of the identity of the certificate holder. While this mechanism is not a requirement of the W3C recommendations, its use is pervasive in desktop browsers and virtually non-existent in mobile browsers. Mobile users are therefore unable to determine if certificates have undergone so-called extended validation, and sites using these certificates may be unable to justify their significant monetary investment in them.

Our measurements and observations from examining the most widely used mobile browsers lead us to make a number of bold assertions *which we expect will be viewed as controversial*, and thus worthy of discussion within the community. (1) Browser designers have been forced by the dramatic reduction in screen space to sacrifice a number of visual security features. The determination of which features are the most useful appears to have been by independent processes, as reflected in the different subsets of security indicators implemented across the mobile platforms. (2) Previous studies have overwhelmingly demonstrated that average users simply do not understand security indicators even on desktop browsers [21–23, 25, 36, 38]. Our measurements demonstrate that the display of security indicators on mobile platforms are considerably worse, to the extent that we as experts cannot express confidence in having sufficient information to take proper decisions. Consequently, we assert that the role of security indicators in mobile browsers offers little more than a false sense of security. The security user interface must therefore either be dramatically improved, to provide indicators of demonstrable use, or should be considerably simplified, to remove unusable, unreliable, or misleading artifacts. (3) We argue that the current practice of repeatedly forcing a user-base that is largely security un-savvy to make subtle security decisions is a losing game. Minor tweaks to the wordings of certificate interface dialogues, for example, may reach a slightly higher local maxima in terms of security improvements, but are highly unlikely to attain a more global maxima offering demonstrably better security. Given the real estate constraints of the increasingly dominant mobile platforms, our evidence shows that this current practice has actually resulted in a decrease in overall security signaling. Consequently, we raise questions not only about the viability of Extended Validation (EV) SSL certificates, but about the ongoing viability of SSL indicators themselves due to the inability to convey accurate, reliable information to users as necessary for subtle security decisions.

For these reasons, we believe that a measurement study as reported in the current paper is a requisite first step for our community to address these difficult issues and deal with these problems head-on.

The remainder of our paper is organized as follows: Section 2 explains the mandatory elements of the W3C guidelines; Section 3 provides the primary results of our evaluation; Section 4 presents ways in that a user can be misled about the identity of a website or the use of encryption and attacks that are enabled by this confusion; Section 5 discusses the presence of EV-SSL, the newest security indicator; Section 6 provides a discussion of our findings; Section 7 presents an overview of related research; and Section 8 offers concluding remarks.

2 Background on W3C Recommendations

The World Wide Web Consortium (W3C) has defined user interface guidelines [9] for the presentation and communication of web security context information to end-users of both desktop and mobile browsers.

We chose a subset of the absolute requirements (MUST) and prohibitions (MUST NOT) specified in the W3C guidelines.[1] We omitted the guidelines represented by clauses including the MAY, MAY NOT, SHOULD and SHOULD NOT keywords as they represent the optional guidelines [3]. We classify the W3C guidelines into *four* categories: identity signal, certificates, robustness and TLS indicators.

1) Identity signal: availability: The security indicators showing identity of a website MUST be available to the user either through the primary or the secondary interface at all times.

[1] The guidelines deemed to be the most critical, definitively testable and enabling attacks when violated were selected based on the authors' experience and knowledge of the area of SSL indicators.

Table 1. Details of the browsers used for experimental evaluation. We selected Android browser version 2.3.3 since it has the highest market share among the available Android platforms [15]. (*: The version numbers of these browsers were not apparent. We have used the default browsers shipped with the referenced version of the OS.)

Category	Browser Name	Version	Rendering Engine	Operating System	Device
Mobile	Android	2.3.3	Webkit	Android 2.3.3	Nexus One
	Blackberry	5.0.0	Mango	Blackberry OS 5.0.0.732	Bold 9650
	Blackberry	6.0.0	Webkit	Blackberry OS 6	Torch 9800
	Chrome Beta	0.16.4130.199	Webkit	Android 4.0.3	Nexus S
	Firefox Mobile	4 Beta 3	Gecko	Android 2.3.3	Nexus One
	Internet Explorer Mobile	*	Trident	Windows Phone 7.0.7004.0 OS	LG-C900
	Nokia Browser	7.4.2.6	Webkit	Symbian Belle	Nokia 701
	Opera Mini	6.0.24556	Presto	Android 2.3.3	Nexus One
		5.0.019802	Presto	iOS 4.1 (8B117)	iPhone
	Opera Mobile	11.00	Presto	Android 2.3.3	Nexus One
	Safari	*	Webkit	iOS 4.1 (8B117)	iPhone
Tablet	Android	*	Webkit	Android 3.1	Samsung Galaxy
	Safari	*	Webkit	iOS 4.3.5 (8L1)	iPad 2
Desktop	Chrome	15.0.874.106	Webkit	OS X 10.6.8	–
	Firefox	7.0.1	Gecko	OS X 10.6.8	–
	Internet Explorer	8.0.7600.16385	Trident	Windows 7	–
	Opera	11.52	Presto	OS X 10.6.8	–
	Safari	5.1.1	Webkit	OS X 10.6.8	–

2) Certificates: required content: In addition to the identity signal, the web browsers MUST make the following security context information available through information sources (certificates): the webpage's domain name and the reason why the displayed information is trusted (or not).
3) Robustness: visibility of indicators: Web content MUST NOT obscure the security user interface.
4) TLS indicators:
a) Significance of presence: Any UI indicator (such as the padlock) MUST NOT signal the presence of a certificate unless all parts of the webpage are loaded from servers presenting at least validated certificates over strongly TLS-protected interactions.
b) Content and Indicator Proximity: Content MUST NOT be displayed in a manner that confuses hosted content and browser chrome[2] indicators, by allowing that content to mimic chrome indicators in a position close to them.
c) Availability: The TLS indicators MUST be available to the user through the primary or the secondary interface at all times.

3 Empirical Observations

We evaluate ten mobile and two tablet browsers against the W3C recommended practices for security indicators. The details of the browsers are provided in Table 1. For each of the guidelines described in Section 2, we create and run a set of experiments to verify compliance on all the candidate browsers and record our observations. All the experiments were performed on web browsers on real mobile phones, and are recreated

[2] A browser chrome refers to the graphical elements of a web browser that do not include web content.

Table 2. Results of experiments on candidate mobile browsers to test compliance with the first three W3C guidelines given in Section 2. Each guideline column consists of sub-columns stating the experiments performed on the browsers. A × implies that the browser does not comply with the respective W3C guideline. A • implies that the browser complies with the respective W3C guideline. NA: Implies that the concerned experiment is not applicable to that browser. Note that all the desktop browsers are compliant to the same guidelines.

Mobile and Tablet Browsers (See Table 1 for versions)	**1) Identity signal: availability**		**2) Certificates: required content**		**Robustness**
	Owner information available?	Cert issuer's information available?	Domain name available?	Information on why certificate trusted available?	Content does not obscure indicators on the address bar?
Android	•	•	•	×	×
Blackberry Mango	•	•	•	•	NA
Blackberry Webkit	•	•	•	•	×
Chrome Beta	•	•	•	×	•
Firefox Mobile	•	•	•	×	×
iPhone Safari	×	×	×	×	×
Nokia Browser	•	•	•	×	×
Opera Mini	×	×	×	×	×
Opera Mobile	×	×	×	×	×
Windows IE Mobile	×	×	×	×	×
Safari on iPad 2	×	×	×	×	•
Android on Galaxy	•	•	•	×	×

in the respective emulators to generate many of the figures throughout the paper. The browser versions used in our evaluation are approximately the latest as of April 20th, 2012. Tables 2 through 4 provide the synopsis of the results of our experiments.

3.1 Identity Signal: Availability

An identity signal contains information about the owner of a website and the corresponding certificate issuer. Before issuing a certificate, the certificate provider requests the contact email address for the website from a public domain name registrar, and checks that published address against the email address supplied in the certificate request. Therefore, the owner of a website is someone in contact with the person who registered the domain name. Popular browsers represent the owner information of a website using different terminology including owner, subject, holder and organization.

We visited a public webpage presenting a trusted root certificate from all the candidate browsers. We then evaluated the browsers for the presence of identity signal, either on the primary or the secondary interface.

Observations: The IE Mobile, iPhone and iPad Safari, and Opera Mini and Mobile browsers do not provide a user interface to view certificates. Accordingly, none of these five browsers comply with the W3C recommendations. We note that when a website presents a certificate that is from a Certificate Authority (CA) not from a trusted root, all the browsers provide an interface to view the certificate via an error message. The Android mobile and tablet, Blackberry Mango and Webkit, Chrome Beta and Nokia browsers always allow a user to view certificates (both trusted and untrusted) and therefore comply with this guideline. A user is required to click the lock icon to view certificate information on the Chrome Beta and Blackberry Mango browsers. However, the browsers do not provide any visual indication to the user about this process of accessing the certificate information. Browsers supporting a UI for viewing certificate information provide a clear indication in the "options" in the browser menu. Although the Firefox Mobile browser does not support a certificate UI, it displays the identity information of a website when the site identity button is clicked. All desktop browsers comply with this guideline. Table 2 provides the summary of our results.

3.2 Certificates: Required Content

In addition to the identity signal content, a certificate from a website must provide the same website's domain name and the reason why the displayed information is trusted (or not). Trust reasons include whether or not a certificate was accepted interactively, whether a self-signed certificate was used, whether the self-signed certificate was pinned to the site that the user interacts with, and whether trust relevant settings of the user agent were otherwise overridden through user action. We believe that information such as "certificate is implicitly trusted" and "the certificate chain is trusted/valid" also conveys the reason behind a browser trusting or not trusting a particular website.

We analyzed the candidate browsers for the presence of the required certificate content by visiting a website that uses strongly TLS-protected connection with its clients.

Observations: The IE Mobile, iPhone and iPad Safari, and Opera Mini and Mobile browsers do not provide a user interface to view certificates from trusted CAs. Therefore, these browsers fail to meet the W3C guideline. Additionally, even though the remaining mobile and tablet browsers provide a user interface to view certificate information, they do not provide an explanation on why a particular certificate is trusted. Only the Blackberry Mango and Webkit browsers comply with the guideline by making all the required parts of a certificate available. When a website presents a certificate from a trusted CA, the Blackberry Mango and Webkit browsers show the reason "certificate is implicitly trusted". Therefore, all but two mobile and tablet browsers fail to meet this W3C guideline. All desktop browsers follow this guideline correctly. Table 2 provides the summary of our results.

3.3 Robustness: Visibility of Indicators

The TLS indicators generally found on the primary interface are lock icon, `https` URL prefix, URL coloring and site identity button. Typically, the address bar in a web browser holds these indicators. Therefore, we examined whether web content overwrites or pushes the address bar out of a user's view during browsing.

Observations: Presumably, in order to free up screen real estate for other purposes, the address bar on all but two mobile and tablet browsers is overwritten by web content once a webpage is rendered and/or when a user starts interacting with the page. The IE Mobile browser always displays the address bar in the portrait view. However, the address bar is never displayed in IE Mobile when a user interacts with a webpage in the landscape mode. The Chrome beta browser makes the address bar persistently available in both the portrait and landscape modes. Out of the two tablet browsers, only the

Table 3. Results of experiments on candidate mobile browsers to test compliance with the W3C guidelines 3a, 3b, and 3c given in Section 2. The symbol notation is as defined in Table 2. 's': Implies that the `https` URL prefix is present on the 's'econdary interface.

Mobile and Tablet Browsers (See Table 1 for versions)	**TLS indicators**					
	3a) significance of presence		**3b) position**	**3c) availability**		
	Mixed content: no lock shown?	Mixed content: no `https` shown?	Favicon not next to lock icon?	`https` prefix available?	Lock shown?	Cipher details available?
Android	Open lock with a question mark	×	×	• (s)	•	×
Blackberry Mango	×	×	•	• (s)	•	•
Blackberry Webkit	×	×	•	• (s)	•	•
Chrome Beta	Closed lock with a cross on top	https striked through	•	• (s)	•	•
Firefox Mobile	No security indicators shown	×	•	• (s)	• On clicking the site identity button	×
iPhone Safari	•	×	•	• (s)	•	×
Nokia Browser	•	×	•	• (s)	•	×
Opera Mini	•	×	•	• (s)	•	×
Opera Mobile	•	×	•	• (s)	•	×
Windows IE Mobile	×	×	•	• (s)	•	×
Safari on iPad 2	•	×	•	• (s)	•	×
Android on Galaxy	Open lock with a question mark	×	•	• (s)	•	×

Table 4. Results of experiments on desktop browsers to test compliance with the guidelines in Table 3. The symbol notation is as defined in Table 2. 'p': Implies that the `https` URL prefix is present on the 'p'rimary interface.

Desktop Browsers (See Table 1 for versions)	**TLS indicators**					
	3a) significance of presence		**3b) position**	**3c) availability**		
	Mixed content: no lock shown?	Mixed content: no `https` shown?	Favicon not next to lock icon?	`https` prefix available?	Lock shown?	Cipher details available?
Chrome	Lock with a yellow triangle	×	•	• (p)	•	•
Firefox	•	×	•	• (p)	• On clicking the site identity button	•
IE	•	×	•	• (p)	•	×
Opera	•	×	•	• (p)	•	•
Safari	•	×	•	• (p)	•	×

tablet Safari browser avoids the security indicators on the address bar being overwritten by a webpage's content, therefore allowing a persistent view of the security indicators on the primary interface. The Android tablet browser hides the address bar once a webpage is rendered. Tables 2 shows that the mobile and tablet browsers do not follow this guideline unlike the desktop browsers.

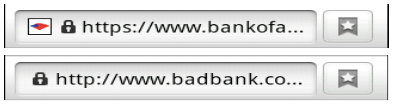


Fig. 1. The address bar of the Android browser when a webpage over SSL is loaded. The browser places the favicon adjacent to the lock icon, thereby violating the W3C guideline 3b described in Section 2. The star icon to the right of the address bar is to bookmark the webpage.

3.4 TLS Indicators

TLS indicators include the `https` prefix, the padlock icon, information about the ciphers used in the connection and URL coloring (or site identity button) to depict the difference between EV-SSL and SSL certified webpages.

a) Significance of presence: If a web browser displays a TLS indicator for the presence of a certificate for a webpage consisting of content obtained over both `http` and `https` connections (mixed content), this guideline is not followed.

We created a simple webpage that uses a strong TLS connection to retrieve the top-level resource and embedded a map obtained from a third-party over an unsecured `http` connection. We rendered this webpage on the candidate browsers and analyzed the browsers for the presence of two basic TLS security indicators: the `https` URL prefix and the padlock icon. If a browser shows any of these two indicators on a mixed content webpage, it does not follow the W3C guideline.

Observations: The Blackberry Mango, Blackberry Webkit and IE Mobile browsers display a lock icon on a webpage holding mixed content, thus failing to meet the W3C guideline. The Blackberry Webkit and IE Mobile browsers display a mixed-content warning and, if the user proceeds to the webpage, a lock icon is displayed. The Android browsers on the mobile and tablet devices present an open lock with a question mark inside the lock. The Chrome Beta browser displays a closed lock with a cross on top and a striked through `https` URL prefix for a mixed content webpage. This behavior of Android and Chrome is inconsistent with the other browsers. Therefore, it is necessary for the users of these browsers to understand the meaning of the new symbols in order to interpret its reference to mixed content on a webpage.

All browsers display the `https` URL prefix on the primary or the secondary interface. We note that this issue is present even in popular desktop browsers. The behavior of displaying the `https` URL prefix on a mixed content webpage fails to meet the W3C recommendation in the desktop and mobile browsers as shown in Tables 3 and 4.

b) Content and Indicator Proximity: The padlock icon used as a security indicator and the favicon used as an identity element of a website are two popular elements that use a browser's chrome. If a browser allows a favicon to be placed next to the padlock, an attacker can feign a secure website by mimicking the favicon as a security indicator. We evaluated this scenario by visiting a webpage over a strong TLS connection from all candidate browsers and observing the relative locations of the favicon and padlock.

Observations: The Android mobile browser does not follow the W3C guideline. The browser places the favicon of a webpage beside the padlock icon as shown in Figure 1. All other browsers adhere to this guideline, as shown in Tables 3 and 4.

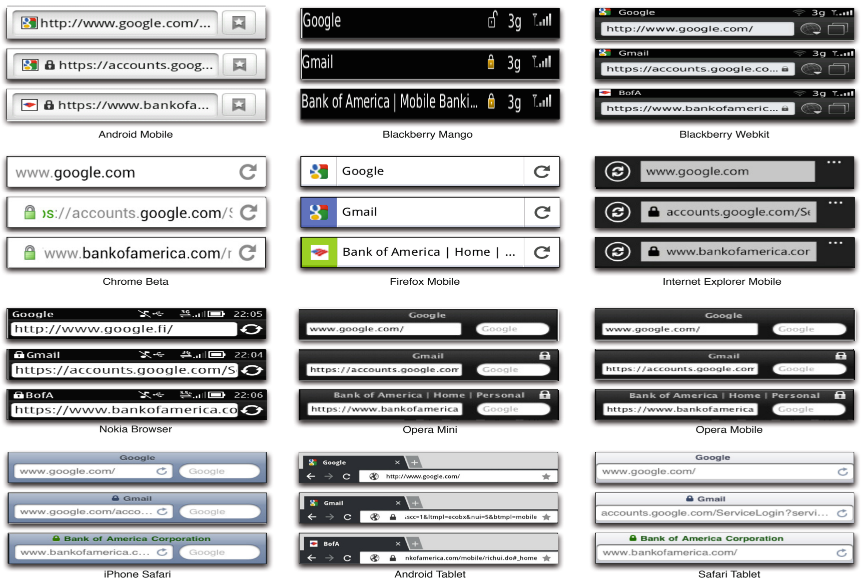


Fig. 2. Security indicators on the primary interface (address bar) of all the mobile and tablet browsers. Every browser has three screenshots of the address bar: from top to bottom, the websites are Google over an `http` connection, Gmail over a secure connection with an SSL certificate and Bank of America over a secure connection with an EV-SSL certificate.

We observed several inconsistencies in the use and position of the padlock icon and the favicon in the mobile and tablet browsers. As shown in Figure 2, the favicon is displayed only on the Android (mobile and tablet), Blackberry Webkit and Firefox Mobile browsers. The remaining mobile and tablet browsers never display a favicon. This behavior is inconsistent with desktop browsers. We believe lack of screen space to be one of the drivers behind the removal of the favicon from the mobile environment. In addition to the almost total lack of use of favicons, we also noticed that the position of the padlock icon in mobile browsers is inconsistent across different mobile browsers. In the past, researchers have shown that the padlock icon is the security indicator most often noticed by users [23,38]. Traditional desktop browsers generally display the padlock icon in the address bar. However, all mobile and tablet browsers except Android (mobile and tablet), Blackberry Webkit, Chrome Beta and IE Mobile browsers display the lock icon on the title bar instead of the address bar. We believe that the reason behind this shift of location of the padlock icon in the mobile and tablet browsers is the non-persistent availability of the address bar to the user. Whenever a user starts interacting with a webpage, most mobile browsers hide the address bar to accommodate more content on the small screen.

c) Availability: We studied the presence of lock icon, `https` URL prefix and details of the cipher used in a TLS connection by visiting a TLS protected webpage from all candidate browsers. The padlock icon and `https` URL prefix are primary interface indicators and cipher information is a secondary interface indicator on desktop browsers.

Observations: Websites handling sensitive digital transactions (such as banks) ask users to search for the `https` URL prefix to ensure security of their transactions. Therefore, easy access to the `https` URL prefix is important. This indicator is present in the address bar (primary interface) of desktop browsers and is clearly visible to the user at all times. Among the mobile and tablet browsers, all but the Blackberry Mango browser display the `https` URL prefix in the address bar. The Blackberry Mango browser does not have an address bar and provides a choice to view the webpage's URL from the browser's options. This setting requires a user to be knowledgeable of the change to be able to find the URL of the current webpage and also makes the `https` URL prefix a secondary interface indicator. Although the other mobile browsers display the `https` URL prefix in the address bar, they hide the address bar (except Chrome Beta) for better usability. In the Chrome Beta browser, if the URL of a webpage is longer than the screen size, the `https` URL prefix is hidden. Since a user is required to interact with the address bar to view the URL prefix of a webpage, the `https` URL prefix becomes a secondary interface indicator in all mobile and tablet browsers. This increases the likelihood of a successful downgrade attack (e.g., SSLstrip [6] attack) on the mobile and tablet browsers, since a user requires effort to view the `https` URL prefix.

The information about the ciphers used in setting up the TLS connection between a website server and the user's browser is not available in any of the browsers except Blackberry Mango and Webkit and Chrome Beta. Accordingly, all the mobile and tablet browsers except three do not comply this W3C guideline for our experiments. Tables 3 and 4 provide the summary of our results.

4 User Deception and Potential Attacks

The W3C user interface guidelines, which we use as a proxy for best practice, are an effort to communicate security information to users such that they can make informed decisions about websites that they visit. If these guidelines are not implemented by a browser, users are more easily misled about the identity of a website or the security of a connection. We discuss four attacks that are enabled on browsers violating one or more of the W3C guidelines. Table 5 provides a summary of potential attacks described in this section on the candidate browsers.

i) Phishing without SSL: An attacker masquerades as a trustworthy entity in a phishing attack. By closely imitating a legitimate website's identity information in combination with lock icon spoofing, a malicious website can launch a phishing attack without SSL on a browser violating the W3C guidelines 1, 2 and 3b as follows.

An attacker buys a domain name that closely resembles the domain name of the legitimate website. For example, to spoof `www.bankofamerica.com`, the attacker buys the domain name `www.bankofamericaa.com`. The attacker then imitates the content of the targeted legitimate website. Instead of spending money on purchasing an SSL certificate to increase the "false" credibility of the malicious website, an attacker instead makes the favicon of the malicious website a lock image. Therefore, the closely imitated domain name provides an impression of correct identity of the intended website and the spoofed lock provides an illusion of strong encryption.

Table 5. Summary of potential attacks on candidate mobile browsers. A × implies that the attack is possible. A • implies that the corresponding attack is not possible on the browser.

Attacks	Android	Blackberry Mango	Blackberry Webkit	Chrome Beta	Firefox Mobile	iPhone Safari	Nokia	Opera Mini	Opera Mobile	IE Mobile	Safari on iPad2	Android on Galaxy
Phishing without SSL	×	•	•	•	•	•	•	•	•	•	•	•
Phishing with SSL	•	×	•	•	•	×	•	×	×	×	×	•
Phishing using a compromised CA	•	×	•	•	•	×	•	×	×	×	×	•
Industrial espionage/ Eavesdropping	×	×	×	×	×	×	×	×	×	×	×	×

When this malicious website is rendered in a browser that makes viewing the URL of the website difficult, situates the favicon next to the padlock icon and does not offer a UI to view identity information such as website owner's name, even an advanced user might be subjected to phishing.

ii) Phishing with SSL: Spoofing only the lock icon may not be adequate to launch a successful phishing attack. To increase the credibility of a phishing website, the attacker can buy an inexpensive SSL certificate for the website. The presence of a valid certificate causes a browser to display SSL indicators such as the `https` URL prefix and URL coloring (or colored site identity button) in addition to the lock icon in the browser's chrome. If a user blindly trusts just these SSL indicators and can not verify additional identity information of the website (violation of guideline 1 and 2), he can be subjected to a phishing attack.

iii) Phishing using a compromised CA: Compromising a CA allows an attacker to obtain rogue certificates for legitimate websites. There have been several such attacks recently [12,13]. If a user's browser trusts a CA, the browser will accept all certificates signed by the CA without showing any warning to the user. This behavior persists even when the same CA is compromised and the necessary update to remove the trusted CA from the browser has not been installed. An expert user who is knowledgeable of a CA compromise can verify every certificate issuer's organization in the certificate chain, therefore declining interacting with a malicious website with a rogue certificate. If a browser fails to meet guidelines 1 and 2, thereby not presenting user interface to enable certificate viewing, even an expert user could be exposed to a phishing attack.

iv) Industrial espionage / eavesdropping: A man-in-the-middle (network) attacker can use any one of the cipher downgrade, substituting `http` for `https` or inserting mixed content techniques for user deception to launch an eavesdropping attack on a user's session as follows:

SSLstrip attack: The SSLstrip [6] man-in-the-middle attacker sits on a local network and intercepts traffic. When the attacker detects a request to an encrypted `https` site, he substitutes a duplicate of the intended destination as an unencrypted `http` site. This switching strips away the security that prevents a third party from stealing or modifying data, while deceiving the server that an encrypted page has been sent to the client. The network attacker can also fake a lock icon in the stripped `http` page, by replacing the favicon by a lock icon [29]. If the `https` prefix is not available to a user persistently, he may not be able to recognize that he is using an unsecured connection by noticing the change from `https` to `http` in the address bar. A browser not displaying the `https` prefix persistently does not follow requirement 3c in Section 2.

Cipher downgrade attack: A man-in-the-middle (network attacker) can tamper with the initial messages sent by a client browser to establish an SSL connection with a website server. Before a TLS connection is set up, a client and server exchange a list of ciphers that they support. A network attacker can modify the list of supported ciphers sent by the client to a list containing only weak ciphers, and then forward the client's request/response to the server. On receiving a list of only weak ciphers (e.g., DES-CBC-SHA), the server can either drop the connection because no ciphers are mutually supported, or provide support for that cipher and begin an encrypted session with the weak cipher. When a connection using the weak cipher is initiated, all the data in transit is protected using the weak cipher's encryption scheme. This allows a network attacker to capture the stream of data and break the weak encryption offline. The attack is also useful to mislead even an expert user that their transactions are over a connection with strong encryption algorithms, since the SSL indicators such as `https` URL prefix and lock icon are present even for a connection using a weak cipher. If a browser does not display cipher information, it fails to meet the W3C requirement 3c in Section 2.

Mixed content attack: A man-in-the-middle attacker can tamper (e.g., code injection) with the unencrypted content present on a webpage consisting of mixed content and replace the original content with any malicious content of his choice. If a web browser displays SSL indicators for a webpage containing mixed content (violation of guideline 3a), even an expert user may be unable to detect a network attack exploiting the mixed content on a webpage.

Our experimental results combined with this threat model make the candidate mobile and tablet browsers susceptible to phishing and eavesdropping attacks as shown in Table 5.

5 Inadequate EV-SSL Differentiation

The W3C guidelines do not establish recommendations for the browser user interface to signify the difference between EV-SSL [8,34] and SSL certificates. The sole distinction between an SSL and an EV-SSL certificate from a user's perspective is the set of indicators on his browser. For example, the Firefox desktop browser uses a green site identity button to convey the presence of an EV-SSL certificate on a website. However, the site identity button is blue in the same browser when a website with an SSL certificate is rendered.

SSL certificates can be 'domain-validation-only' with minimal verification performed on the details of the certificate. Since any successful SSL connection causes the padlock icon to appear, users are not likely to be aware of whether the website owner has been validated or not. Therefore, fraudulent websites have started using inexpensive domain-validated SSL certificates with minimal verification to gain user trust. EV-SSL

Table 6. Results of whether browsers differentiate between EV-SSL and SSL certified webpages. A × implies that the browser does not provide differentiating indicators and a • implies the presence of such indicators in the browser.

	Android	Blackberry Mango	Blackberry Webkit	Chrome Beta	Firefox Mobile	iPhone Safari	Nokia	Opera Mini	Opera Mobile	IE Mobile	Safari on iPad2	Android on Galaxy
EV-SSL v/s SSL differentiation	×	×	×	×	•	•	×	×	×	×	•	×

certificates were created to restore confidence among users that a particular website owner has been subjected to more rigorous vetting than simply determining control of a domain name [37]. If browsers do not differentiate between SSL and EV-SSL certificates, then the fundamental motivation [8] behind EV-SSL certificates becomes void. So too does the incentive for site owners to pay extra for such certificates – an SSL certificate from Go Daddy costs $12.99/year [1] and an EV-SSL certificate from VeriSign costs $1499/year [2]. In a browser with no differentiation between SSL and EV-SSL, both these certificates appear the same from a user's perspective. An adversary holding a domain name and willing to spend money for the SSL certificate would then trigger exactly the same user interface elements to users, and thus appear to provide identical guarantees as a website certified by the more expensive certificate.

Experimental Observations: We browsed both EV-SSL and SSL certified webpages using all the candidate browsers. With the exceptions of the Firefox Mobile and the iPhone and iPad Safari browsers, none of the mobile or tablet browsers display any indicators that differentiate between EV-SSL and SSL certified webpages. The Firefox Mobile browser changes the site identity button green or blue to depict the presence of EV-SSL and SSL certified webpages respectively. The Safari mobile and tablet browsers use green and blue coloring of the 'title' to represent the difference between EV-SSL and SSL. This behavior of the Firefox Mobile and the Safari browsers is consistent with their desktop counterparts. All tested desktop browsers indicate the difference between these two certificate types. Table 6 provides a summary of the results.
We note the following advice within official guidelines from the CA/Browser Forum [5]:

In cases where the relying application accepts both EV and non-EV certificates, it is recommended that the application's behavior differ in a distinct way for each type of certificate. Application developers should consider the EV treatment offered by other application developers that also recognize EV certificates and, where practical, provide consistent treatment.

We assert that much more specific advice is essential, for example, in a revision or extension of the W3C user interface guidelines [9] and potentially in the CA/Browser Forum's own recommendations, as well as in appropriate supporting standards.

6 Discussion and Implications

Having performed comprehensive measurements of security indicators in the most widely used mobile browsers (over 90% of the market share), we now discuss what we see to be the implications. We have separated this from our measurements to allow independent interpretations by others.

It appears quite clear that the lack of available screen real estate has dramatically influenced the use of security indicators on mobile devices. Whereas traditional browsers have a considerably richer space budget to accommodate current (and experimental) indicators, mobile browsers struggle to clearly show content, let alone signals of the origin and security of the content's connection. Our study suggests that browser vendors have individually made different decisions to best balance these competing demands, independently selecting to implement different subsets of the indicators in use on desktop browsers. Unfortunately, our study also reveals that the choices made by each browser removes signals available to users to detect, and possibly avoid, specific attacks—even if only by expert users. It is *not* our conclusion that implementing exactly the same security indicators on mobile platforms is impossible from an engineering perspective; but rather, the real estate limit of mobile phones makes unclear the means of doing so in a manner that does not simply overwhelm the content on small mobile screens.

Moreover, even if possible, it is entirely unclear that duplicating all desktop security indicators on mobile browsers is the best course of action to pursue. Studies over the last decade have repeatedly shown that *average* users either ignore or do not understand browser security indicators [21–23, 25, 36, 38]. While usability and HCI experts have contributed to some improvements, the community appears to be at a stalemate, lacking new ideas—even across 10 mobile browsers—to improve (or retain) security signaling while preserving usability. For example, wording and sequencing of warning dialogs has arguably been improved, but such changes are minor enhancements at best—and appear to move us at best to local maxima, constrained by the current set of indicators and the SSL security framework, while higher global maxima seem beyond reach.

Given these observations, our work reaches the following logical conundrum: Mobile browsers are currently vulnerable to attacks that could potentially be ameliorated by implementing known security indicators at least as well as on desktops. These indicators have not been implemented due to space constraints. Moreover, were these indicators to be implemented on mobile browsers (likely at the expense of page content), it is entirely unclear that users would profitably notice them. Accordingly, our community faces a difficult question:

> *Should mobile browsers vendors continue to include only a subset of indicators and provide a false sense of security to even expert users, or implement the full corpus of indicators (which past studies indicate users will largely ignore) at the risk of drastically reducing the usability of these applications?*

Answering this question invokes a discussion of the limits an SSL security framework, based on third-party certificates, with trust deriving from a combination of browser configuration data and user input. A key assumption is that users can be provided with sufficient interface cues (e.g., regarding the origin and algorithms used to exchange sensitive content) and then take correct security actions based thereon. However, as the cited studies (and others) indicate, the largely security un-savvy average user cannot perform this task. Presented with subtle streams of information, users cannot realistically be expected to sort out differences between cryptographic algorithms, nor proper responses to increasing incidents of CA compromise [12, 13]. In short, continuing to "punt" security decisions to users who, in spite of browsing daily for the past decade have not become security experts, is a strategy that appears no longer to anyone's advantage. Compounded by the screen constraints of mobile devices, and these devices becoming the dominant means of end-user computing on the planet, continuing the current approach becomes even less plausible. We argue that something must change.

As a straw-man suggestion, to progress conversation, we encourage discussion of the following resolution. As mobile users increasingly offload computationally expensive operations to "the cloud" (e.g., complex image rendering), so too might they outsource more security decisions. Large external organizations or specialist entities are more likely to have an understanding of the currently appropriate security footing for a browser (e.g., sufficiently strong algorithms, websites using certificates from bad CAs, etc.). Accordingly, the *average* user could be better served by simply receiving a single indicator representing the opinion of the identity and security of the connection through one of these entities. Desktop browsers such as Chrome are already beginning to rely on such services to protect users from malicious downloads and phishing websites, and other researchers have proposed using CDNs to provide more expensive security services (e.g., defense against heap spraying attacks) [28]. Based on browser configuration, expert users could continue to make their own decisions, perhaps through secondary interfaces. In particular, given that our study shows that the majority of security information available in mobile browsers is in such secondary interfaces, expert users could look at a more complete set of information than currently available, and make more informed decisions themselves. This approach would both minimize the space required for security indicators and would potentially provide a more useful metric for users.

We do not believe that this straw-man proposal is without any issues, but rather that the community is overdue to directly confront the (lack of) security provided by current indicators. Our measurement study shows that currently, the value is clearly insufficient.

7 Related Work

TRADITIONAL BROWSER INDICATORS: Traditional desktop browsers contain a range of security indicators in the chrome of the browser including the lock icon, the `https` URL prefix, and certificates. Several studies have indicated that these security cues used in desktop browsers go unnoticed [21, 23, 33, 34, 38] or are absent in websites [35]. Although domain name mismatches between certificates and websites are observed often [37], Sunshine et al [36] showed that users ignore TLS warnings for domain name mismatches, and showed that users ignore TLS warnings for expired certificates and unknown CAs. Moreover, a majority do not understand these warnings. The lock icon is the security indicator most often noticed [23, 38]. However, even when used as a security cue by users, many do not fully understand its meaning [21–23] and its absence also often goes unnoticed [21]. Additionally, the majority of users who rely on the lock icon remain unaware of its identity feature [21, 23, 25, 38] and do not reliably understand the concept of certificates [21, 22]. Indicators for newer technologies such as EV-SSL have also been shown to be ineffective to convey better security to the user as compared to a simple SSL certificate [17, 27].

TECHNIQUES FOR BETTER INDICATORS: Several techniques have been proposed to design better security indicators to prevent potential attacks such as phishing. Researchers have proposed better warnings [36], more effective interface dialogues [17], browser plugins [19], trusted path from the browser to the human user [39] and mandatory security indicators [26] to help users make correct security decisions. Other proposed security mechanisms include disabling JavaScript in the user browser and forcing persistent visibility of the browser's location line [24]. Dynamic Security Skins [22] allow a remote web server to prove its identity in a way that is easy for a human user to verify and hard for an attacker to spoof. Finally, efforts have been taken [5,8,9,11,14] to standardize security indicators and thus minimize confusion across browsers.

MOBILE BROWSER INDICATORS: Almost all the efforts in the area of security indicators in browsers have been focused on desktop browsers. The increasing user base of mobile web browsers and mobile e-commerce has made mobile browsers attractive targets for attacks [4,7,10,20,30–32]. In light of these developments and considering how the mobile browser user interface differs from desktops, it is important to analyze and understand the security indicators used in mobile browsers. Although the W3C [9] guidelines consider mobile browsers in their definitions, a large-scale evaluation of the state-of-the-art security indicators in mobile browsers has not been carried out.

8 Concluding Remarks

Modern mobile browsers enable a range of sensitive operations over SSL/TLS connections. Although these browsers aim for equivalent functionality to traditional desktops, their smaller screen size has resulted in significant changes to the presentation and availability of SSL indicators. This paper presents the first large scale, cross-sectional measurement of this class of applications and compares the security indicators used in the overwhelming majority of mobile browsers to their traditional desktop counterparts. Our results are threefold: that mobile browsers implement only a subset of the recommended indicators from the desktop world thus eliminating the opportunity for even expert users to avoid attacks such indicators might signal, that the subset chosen across each browser are inconsistent and that the newest indicator (EV-SSL) is virtually unseen. Our measurements lead us to the conclusion that current security indicators force our community to either accept a false sense of security or to argue for the complete implementation of the W3C's best practices at the cost of screen space otherwise available for content. Having presented our empirical evidence of these problems, we argue for high profile exposure of our results within the security community. Moreover, we offer a straw-man proposal to motivate a call-to-arms to address mobile browser security interface challenges, and better allow informed decisions based on knowledge of the state-of-the-art.

The importance of mobile browsers is only increasing. In particular, a growing number of the most popular mobile apps (e.g., Facebook, ESPN) are simply wrappers for the mobile browser via APIs such as WebViews. This approach provides the advantage of a consistent user experience across virtually all platforms while drastically reducing the engineering effort to achieve this feat. In spite of being reliant on the browser, such apps provide users with *none* of the traditional indicators of security offered by the browsers. Specifically, these APIs present all of the content of the browser without any of the chrome, meaning that all primary and secondary security interfaces are absent. If such indicators are critical to providing security to browser users, the community must then confront the question directly and argue decisively for their use. If they are not, the community should instead argue to remove them entirely to avoid unnecessary use of precious real estate, user confusion and a false sense of security. Regardless of which argument is made, the community must face these questions head-on.

References

1. GoDaddy SSL certificate, http://www.godaddy.com/Compare/gdcompare_ssl.aspx?isc=sslqgo016b
2. VeriSign certificate, https://www.verisign.com/ssl/buy-ssl-certificates/index.html?sl=t72010166130000002&gclid=CIKMyY2GuKgCFYg32godV2_8Bw
3. Key words for use in RFCs to Indicate Requirement Levels (March 1997), http://www.ietf.org/rfc/rfc2119.txt
4. Overflow clickjacking (November 2008), http://research.zscaler.com/2008/11/clickjacking-iphone-style.html
5. Guidelines for the Processing of EV Certificates, version 1.0 (January 2009), http://www.cabforum.org/Guidelines_for_the_processing_of_EV_certificatesv1_0.pdf
6. SSLstrip, presented at Black Hat DC (2009), http://www.thoughtcrime.org/software/sslstrip/
7. Android Browser Exploit (2010), http://threatpost.com/en_us/blogs/researcher-publishes-android-browser-exploit-110810
8. Guidelines for the Issuance and Management of Extended Validation Certificates, version 1.3 (November 20, 2010), http://www.cabforum.org/Guidelines_v1_3.pdf
9. W3C: Web Security Context: User Interface Guidelines (August 2010), http://www.w3.org/TR/wsc-ui/
10. Web-based Android attack (November 2010), http://www.infoworld.com/d/security-central/security-researcher-releases-web-based-android-attack-317?source=rss_security_central/
11. Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates, version 1.0 (April 11, 2011), http://www.cabforum.org/Announcement-Baseline_Requirements.pdf
12. Comodo compromise (April 1, 2011), http://www.csoonline.com/article/678777/comodo-compromise-expands-hacker-talks
13. DigiNotar CA compromise (August 30, 2011), http://community.websense.com/blogs/securitylabs/archive/2011/08/30/diginotar-ca-compromise.aspx
14. The CA/Browser forum (April 11, 2011), http://www.cabforum.org/
15. Android OS market share by version (May 2012), http://developer.android.com/resources/dashboard/platform-versions.html
16. Mobile Browser Market Share (May 2012), http://gs.statcounter.com/#mobile_browser-ww-monthly-201204-201205
17. Biddle, R., van Oorschot, P., Patrick, A., Sobey, J., Whalen, T.: Browser interfaces and extended validation SSL certificates: an empirical study. In: Proceedings of the ACM Workshop on Cloud Computing Security (2009)

18. Boodaei, M.: Mobile users three times more vulnerable to phishing attacks (2011), http://www.trusteer.com/blog/mobile-users-three-times-more-vulnerable-phishing-attacks
19. Chou, N., Ledesma, R., Teraguchi, Y., Boneh, D., Mitchell, J.: Client-side defense against web-based identity theft. In: Proc. NDSS (2004)
20. Davies, C.: iPhone Os Safari Vulnerable To DoS Attacks (April 16, 2008), http://www.iphonebuzz.com/iphone-safari-dos-bug-discovered-162212.php
21. Dhamija, R., Tygar, J.D., Hearst, M.: Why phishing works. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (2006)
22. Dhamija, R., Tygar, J.: The battle against phishing: Dynamic security skins. In: Proceedings of the Symposium on Usable Privacy and Security (2005)
23. Downs, J., Holbrook, M., Cranor, L.: Decision strategies and susceptibility to phishing. In: Proceedings of the Second Symposium on Usable Privacy and Security (2006)
24. Felten, E.W., Balfanz, D., Dean, D., Wallach, D.S.: Intrusion Detection Prevention Web Spoofing: An Internet Con Game. In: 20th National Information Systems Security Conference (1997)
25. Friedman, B., Hurley, D., Howe, D., Felten, E., Nissenbaum, H.: Users' conceptions of web security: a comparative study. In: CHI Extended Abstracts on Human Factors in Computing Systems (2002)
26. Herzberg, A., Jbara, A.: Security and identification indicators for browsers against spoofing and phishing attacks. ACM Transactions on Internet Technology (2008)
27. Jackson, C., Simon, D.R., Tan, D.S., Barth, A.: An Evaluation of Extended Validation and Picture-in-Picture Phishing Attacks. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 281–293. Springer, Heidelberg (2007)
28. Livshits, B., Molnar, D.: Empowering Browser Security for Mobile Devices Using Smart CDNs. In: Proceedings of the Workshop on Web 2.0 Security and Privacy, W2SP (2010)
29. Marlinspike, M.: More Tricks For Defeating SSL in Practice (2009), http://www.blackhat.com/presentations/bh-usa-09/MARLINSPIKE/BHUSA09-Marlinspike-DefeatSSL-SLIDES.pdf
30. Niu, Y., Hsu, F., Chen, H.: iPhish: Phishing Vulnerabilities on Consumer Electronics. In: Usability, Psychology, and Security (2008)
31. Porter Felt, A., Wagner, D.: Phishing on mobile devices. In: Web 2.0 Security and Privay (2011)
32. Resig, J.: iPhone overflow clickjacking (November 2008), http://ejohn.org/blog/clickjacking-iphone-attack/
33. Schechter, S., Dhamija, R., Ozment, A., Fischer, I.: The Emperor's New Security Indicators. In: IEEE Symposium on Security and Privacy (2007)
34. Sobey, J., Biddle, R., van Oorschot, P.C., Patrick, A.S.: Exploring User Reactions to New Browser Cues for Extended Validation Certificates. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 411–427. Springer, Heidelberg (2008)
35. Stebila, D.: Reinforcing bad behaviour: the misuse of security indicators on popular websites. In: Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction (2010)
36. Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., Cranor, L.F.: 18th USENIX Security Symposium Crying Wolf: An Empirical Study of SSL Warning Effectiveness. Work (2009)
37. Vratonjic, N., Freudiger, J., Bindschaedler, V., Hubaux, J.P.: The inconvenient truth about web certificates. In: The Workshop on Economics of Information Security, WEIS (2011)
38. Whalen, T., Inkpen, K.: Gathering evidence: use of visual security cues in web browsers. In: Proceedings of Graphics Interface (2005)
39. Ye, Z.E., Smith, S., Anthony, D.: Trusted paths for browsers. ACM Transactions on Information and System Security (TISSEC) (May 2005)

Domain-Specific Pseudonymous Signatures for the German Identity Card

Jens Bender[1], Özgür Dagdelen[2], Marc Fischlin[2], and Dennis Kügler[1]

[1] Bundesamt für Sicherheit in der Informationstechnik (BSI), Germany
[2] Technische Universität Darmstadt, Germany

Abstract. The restricted identification protocol for the new German identity card basically provides a method to use pseudonyms such that they can be linked by individual service providers, but not across different service providers (even not malicious ones). The protocol can be augmented to allow also for signatures under the pseudonyms. In this paper, we thus view —and define— this idea more abstractly as a new cryptographic signature primitive with some form of anonymity, and use the term domain-specific pseudonymous signatures. We then analyze the restricted identification solutions in terms of the formal security requirements.

1 Introduction

The protocols designed for the new German identity card [7] secure the communication between the card and the reader resp. the terminal. See Figure 1 for an overview. Initially, the parties run the password-authenticated connection establishment protocol (PACE) to derive a secure channel between the card and the local reader, with the user's consent who needs to enter the password. Then the card and the remote end point, called the terminal or service provider, run the extended access protocol (EAC) to authorize mutually and to establish another secure channel between these points. In the card authentication step anonymity is guaranteed by the fact that a large number of cards share the same secret. The security of these protocols has been analyzed in [4,6,12].

The overall design of the identity card includes another protocol, called restricted identification. In this optional protocol, card holders can use domain-specific pseudonyms to interact with service providers such that (a) a service provider can recognize pseudonyms of individual cards and use this information for the service (domain-specific linkability), and (b) different service providers cannot link interactions of one user in their respective domains (cross-domain anonymity). Although the concept of restricted identification in [7] —and the Diffie-Hellman based solution— currently only support recognition of pseudonyms, it can be easily

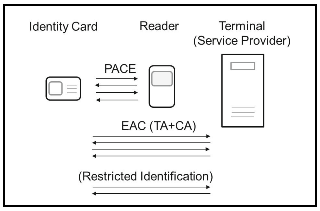


Fig. 1. Protocols for the German ID Card

extended for more functionality, by allowing users to create signatures under their pseudonyms.

1.1 Domain-Specific Pseudonymous Signatures

To analyze the protocols of the new German identity card we formally introduce the concept of *domain-specific pseudonymous signatures* which captures the scenario of the restricted identification protocols, augmented through signatures, but also allows to reason about the security of the basic restricted identification protocol.[1] In a sense, (domain-specific) pseudonymous signatures can be seen as a relaxed version of group signatures with a limited form of linkability: While group signature schemes, as formalized in [3], provide a very strong form of anonymity, preventing an adversary to identify signers even when knowing the secret keys, pseudonymous signatures are designed specifically to allow a well-defined verifier to link signatures.

The group-signature ancestry also lays the ground to model the security of pseudonymous signatures. As mentioned, according to [3], secure group signature schemes should satisfy full anonymity, the resistance to identify the origin of signatures, and full traceability, the ability of the group manager to trace the origin of a signature with the help of some trapdoor information. The latter property implies, for example, unforgeability (because a forgery could not be traced) and non-frameability, saying that a set of malicious parties, potentially including the group manager, could falsely blame an honest user (which would again contradict traceability).

Concerning anonymity, we note that the goal of restricted identification is exactly to allow a service provider to link previously seen pseudonyms. We thus relax the security requirement for pseudonymous signatures and only demand cross-domain anonymity, i.e., the inability to link signatures given for different service providers, even if both of them are malicious. Furthermore, since the restricted-identification scenario does not involve an authority for traceability, we revert to unforgeability directly for our setting, meaning that one cannot forge signatures on behalf of honest users.

We also introduce another property saying that it is infeasible to make the verifier accept a signature for an invalid domain-specific pseudonym. This assumes a white- or black-listing approach, in the former case making sure that valid pseudonyms are those in the white list, and in the other case demanding that the verifier would not accept a black-listed pseudonym. We observe that this property contrasts with unforgeability which merely protects honest signers from signature forgeries under their name. While this additional property, which we call *seclusiveness*, follows from full-traceability for group signatures, we need to state it explicitly in our setting without a tracing authority. Roughly, unforgeability together with seclusiveness provide a weaker, yet "best-we-can-hope-for" form of full-traceability for domain-specific pseudonymous signatures.

[1] Interestingly, the term "pseudonymous signatures" has occasionally been already mentioned in the literature, typically referring to regular signatures under pseudonyms [14], but not considered formally, from a cryptographic point of view.

1.2 Analysis of the Restricted-Identification Solutions

We then continue to analyze the current solution for restricted identification (and its signature-augmented version). The basic solution is roughly to give users a random value x_1 and the service provider a certified group element R, such that the user will derive the domain-specific pseudonym as the Diffie-Hellman key $I_R = R^{x_1}$ (or, to be precise, the hash value thereof). This value is then sent over a previously established secure channel to the service provider (but we ignore the channel part in our analysis).

To achieve the additional unforgeability and seclusiveness properties in the signature-augmented case the user is instead given a random representation (x_1, x_2) of the authority's public key y, i.e., $y = g_1^x = g_1^{x_1} g_2^{x_2}$ for generators $g_1, g_2 = g_1^z$. Here, the authority can create such representations easily with the knowledge of $z = \log_{g_1} g_2$. Each domain will again hold a certified group element R, and the user will derive once more the domain-specific pseudonym as the Diffie-Hellman key $I_R = R^{x_1}$ and transmits this value to the provider.

Obviously, if two malicious users could pool their distinct secrets (x_1, x_2) and (x_1', x_2') then they could recover the authority's secret key z. However, we note that these secrets are protected through the hardware of the identity card and are not available to users. In a sense, our security analysis relies on this fact, but at the same time even allows a single (malicious) user access to its secret key. In case of suspicion of leakage of the authority's secret key z a new key can be generated, the old key can be revoked, but white-listing can be used to mark the user keys under the old key as still valid.

In the signature-augmented version, the user additional signs a message m by giving two intertwined non-interactive proofs of knowledge (where the message enters the hash evaluation to derive the challenge non-interactively in the random oracle model). This first proof shows that the user knows the discrete logarithm x_1 of I_R to base R, and the other proof shows that it additionally knows x_2 such that (x_1, x_2) forms a representation of $y = g_1^{x_1} g_2^{x_2}$. The first proof is basically a Schnorr proof of knowledge (PoK) [21], and the second one is an Okamato PoK [17] but re-using the data from the Schnorr proof.

We show that the basic restricted-identification protocol ensures cross-domain anonymity according to our notion. Since pseudonyms $I_R = R^{x_1}$ are essentially Diffie-Hellman keys, this holds under the Decisional Diffie-Hellman assumption. For the signature-augmented version we show that unforgeability holds as well (in the random oracle model, and under the discrete log assumption). This follows from the Schnorr PoK for x_1. Seclusiveness follows from the Okamoto proof showing that the pair (x_1, x_2) is a representation of y and has thus been issued by the authority. Here, however, due to the construction we require that no two malicious users collaborate. As explained above, this is prevented by hiding and protecting the secret of users within the secure hardware of the chip. Thereby, users can use their secrets, however, they cannot extract the keys to collude. Note that the basic version does not implement a signature functionality and thus cannot satisfy these two notions.

1.3 Related Work

As noted above, domain-specific pseudonymous signatures can be seen as descendants of group signatures, only with weaker anonymity requirements but with domain-specific verifiers. The related concept of ring signatures [20] can be viewed as "ad-hoc" group signatures without a central manager. For such schemes, rings of users can be formed at will, and one user can sign on behalf of the ring, but there is usually no mean to identify the actual signer later. As such, domain-specific pseudonymous signatures provide stronger notions of traceability than ring signatures, but a weaker form of anonymity —the domain holder can link signatures.

We note that credential systems [5,8] are very similar to our pseudonymous signatures, but diverge in some important aspects. Most importantly, credential systems typically provide multi-show unlinkability, as opposed to our pseudonymous signatures. As such, solutions for multi-show credential systems are usually slightly more complex [9–11]. For one-show solutions, where the user can use a credential (under a pseudonym) only once, domain-specific linkability —and therefore cross-domain anonymity— has not been considered before. We also note that the question of turning cross-domain pseudonymous signatures into full-fledged multi-show credential systems is beyond the scope of the paper here: the requirement of recognizing pseudonyms for domain holders is inherent in the application requirement.

Finally, let us point out that our notion of domain-specific pseudonymous signatures is close to a recent proposal of Bernhard et al. [2] for defining direct anonymous attestation (DAA). Their security definition for DAA also resembles the one for group signature schemes but comes with a limited form of linkability: signatures of the same user in the same domain (called base there) should be publicly linkable. At the same time, one should be able to identify signatures given the user's secret key, a requirement which we do not impose in our setting. Moreover, their scenario assumes that linkability must be enforced via cryptographic means, domain-specific pseudonymous signatures allow this by default through the pseudonyms. Besides minor technical differences concerning security in presence of compromised keys or incorporating black-listing, the main difference to their setting is that our model takes into account the extra layer of domain-specific pseudonyms and its unlinkability to the pseudonym layer.

Similarly, Wei [22] also uses DAA as a motivation for his work on tracing-by-linking group signatures, but he considers a weaker form of anonymity where a signer's identity is only hidden if he signs at most a fixed number of times for $k \geq 1$. Any additional signature enables to trace back the identity of the signer by a public algorithm. The approach also includes a notion of k-linkability which corresponds to (public) traceability and essentially means that more than k signatures are linkable and allow to identify the origin. It also contains a notion of non-slanderness which means that no group of malicious users (including the issuing authority) can sign more than k times such that it points to an honest user outside of the group. This primitive does not help in our scenario, because we demand unlinkability only among different domain sectors. Furthermore, our notion of seclusiveness, in contrast with non-slanderness which addresses the linkability of more than k signatures, refers to the fact that one cannot produce any signature on behalf of honest users.

The idea of extending restricted identification to allow unlinkable sector signatures has appeared concurrently in [15]. The scheme does not cover the issue of seclusiveness, though, and is less explicit about the underlying security model, e.g., it remains unclear if unlinkability holds for multiple signatures. Exploring such questions and providing sound models, including issues related to black-listing or white-listing, and to prove security according to these models for the (augmented) restricted identification protocol of the new German identity cards is our contribution here.

2 Domain-Specific Pseudonymous Signatures

To simplify the notation we denote by $\{a\}_n$ a set of n elements $a_1, \dots, a_n$ where the indices $i = 1, 2, \dots, n$ are implicit.

2.1 Preliminaries

Below we define our domain-specific pseudonymous signature scheme first for static groups. Since both the secret keys of users and the domain keys are chosen by the authority, we can imagine that a sufficiently large set is chosen at the outset, and individual entries only become active if required. We hence assume an algorithm $\mathsf{NymKGen}$ generating the group manager's key pair, as well as sufficiently many pseudonyms nym (and secret keys $\mathsf{gsk}[\mathsf{nym}]$) and public domain keys dpk. Each pseudonym and its secret key can now be combined via an algorithm $\mathsf{NymDSGen}$ to build a domain-specific pseudonym $\mathsf{dsnym} = \mathsf{nym}[\mathsf{dpk}]$ which, together with $\mathsf{gsk}[\mathsf{nym}]$ can be used to sign messages.

In addition, we can also introduce the concept of black-lists to the model. For this, we assume that the list is represented by a (dynamically updated) set B of domain-specific pseudonyms which have been revoked earlier. We note that, alternatively, one could employ white-listing where valid pseudonyms are those included in the list; to revoke such a pseudonym one simply removes the entry from the white-list.

Definition 1 (Domain-Specific Pseudonymous Signature). *A domain-specific pseudonymous signature scheme is a collection of the following efficient algorithms* $\mathcal{NYMS} = (\mathsf{NymKGen}, \mathsf{NymDSGen}, \mathsf{NymSig}, \mathsf{NymVf})$ *defined as follows.*

$\mathsf{NymKGen}(1^\kappa, 1^n, 1^d)$ *is a probabilistic algorithm which, on input a security parameter* 1^κ *and parameters* $1^n, 1^d$ *(both polynomial in* κ*) outputs a pair* $(\mathsf{gpk}, \mathsf{gmsk})$ *where* gpk *is the group public key and* gmsk *the secret key of the group manager, and outputs* n *(unique) pseudonyms* nym *with their corresponding secret keys* $\mathsf{gsk}[\mathsf{nym}]$*, and* d *domain descriptions* dpk.

$\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk})$ *is a deterministic algorithm which maps a pseudonym* nym *(and its secret key* $\mathsf{gsk}[\mathsf{nym}]$*) and the domain* dpk *to a domain-specific pseudonym* $\mathsf{dsnym} = \mathsf{nym}[\mathsf{dpk}]$*.*

$\mathsf{NymSig}(\mathsf{dsnym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}, m)$ *is a probabilistic algorithm which, on input a domain-specific pseudonym* dsnym*, a secret key* $\mathsf{gsk}[\mathsf{nym}]$*, a domain* dpk*, and message* m*, outputs the signature* σ *of* m *under* $\mathsf{gsk}[\mathsf{nym}]$ *for domain* dpk*.*

$\mathsf{NymVf}(\mathsf{gpk}, \mathsf{dsnym}, \mathsf{dpk}, m, \sigma, \mathsf{B})$ *is a deterministic algorithm which, on input a message* m *and a signature* σ *together with the group public key* gpk*, a domain-specific pseudonym* dsnym*, the domain's key* dpk*, and a list* B*, outputs either* 1 *(=valid) or* 0 *(=invalid).*

We assume the usual completeness property, that for any honestly generated parameters, domain-specific pseudonyms, and signatures the verification algorithm accepts and outputs 1*.*

Note that we can assume that the group manager uses some standard way of certification for the public keys dpk given out to registered verifiers, and that the signing and verification algorithms check the validity of the keys. We thus often omit this step from the description of protocols.

2.2 Cross-Domain Anonymity

Cross-domain anonymity protects against linking pseudonyms across different domains —within one domain pseudonyms are meant to be linkable. We define cross-domain anonymity via a game between the adversary and a left-or-right oracle which, given two pseudonyms, a message, and a domain, generates a signature for either the left or the right pseudonym in the domain, according to a secret random bit b. We assume that the adversary can make adaptive calls to this left-or-right oracle; this is necessary since we cannot apply a hybrid argument to reduce such multiple queries to a single one (as opposed to the case of full-anonymity for group signatures as in [3], where this is possible since anonymity even holds if the adversary knows the users' secret keys). The adversary's goal is to predict b significantly beyond the pure guessing probability (condition (a) below).

In addition to challenge queries to the left-or-right oracle, the adversary may decide to blacklist domain-specific pseudonyms, to create additional signatures via NymSig, and to corrupt users. For simplicity, we define a version for *static corruptions* where all corruptions are made at the outset, before any other oracle calls are made, and discuss the adaptive version briefly below. To exclude trivial attacks, we must take into account that domain-specific pseudonyms are in principle linkable by the domain holder. Hence, in "transitivity" attacks where the adversary asks the left-or-right oracle first about a signature for domain-specific pseudonyms $\mathsf{dsnym}_0, \mathsf{dsnym}_1$ and then for $\mathsf{dsnym}_0, \mathsf{dsnym}'_1$ for the same domain, but $\mathsf{dsnym}_1 \neq \mathsf{dsnym}'_1$, the signatures would point to the same domain-specific pseudonym if and only if the oracle signs under the left pseudonym. We thus exclude such queries in condition (b) below.

We require another case to be excluded. Namely, the additional signatures generated through NymSig cannot hide the pseudonyms behind the signatures; this would require further means like anonymous signatures and would only work if signatures are not publicly verifiable [13,23]. Since such extra signatures for domain-specific pseudonyms would thus also allow to link the origin in the left-or-right queries to the pseudonyms, we must disallow the adversary from querying NymSig about domain-specific pseudonyms, which are also used in left-or-right queries (condition (c) below).

The model below assumes, to the advantage of the adversary, that all pseudonyms, all domain keys, and domain-specific pseudonyms are known at the outset; only the assignment of pseudonyms to domain-specific pseudonyms remains hidden. Nonetheless, we assume that the relation of domains and domain-specific pseudonyms is known (see below for the motivation). All this is captured by giving the adversary the corresponding data as sets, with $\mathsf{W} \odot \mathsf{D}$ being the set of all domain-specific pseudonyms dsnym and the corresponding domains dpk. The adversary will thus attack domain-specific pseudonyms from W. As common, we measure the adversary's running time including also all steps of honest parties, and covering both phases of the adversary.

In the definition, we presume that domain-specific pseudonyms are unique within a domain; global uniqueness can then be trivially achieved by attaching the domain key to the pseudonym. Indeed, domain-specific uniqueness will be later ensured by the unforgeability property anyway.

Definition 2 (Cross-Domain Anonymity). *A domain-specific pseudonymous signature scheme* $\mathcal{NYMS} = (\mathsf{NymKGen}, \mathsf{NymDSGen}, \mathsf{NymSig}, \mathsf{NymVf})$ *is* (n, d, t, Q, ϵ) *cross-domain anonymous with* $Q = (q_c, q_s, q_t)$ *if for any algorithm* $\mathcal{A}$ *running in time* t*, and making at most* q_c *queries to the corruption oracle,* q_s *queries to the signing oracle, and* q_t *queries to the left-or-right oracle, the probability that the following experiment returns* 1 *is at most* ϵ*:*

```
Experiment $\mathsf{CD-Anon}_{\mathcal{A}}^{\mathcal{NYMS}}(\kappa, n, d)$
  $b \xleftarrow{\$} \{0,1\}$
  $\mathsf{T}, \mathsf{S}, \mathsf{W}, \mathsf{B}, \mathsf{C}, \mathsf{N}, \mathsf{D} \leftarrow \emptyset$
  $(\mathsf{gpk}, \mathsf{gmsk}, \{\mathsf{gsk}[\mathsf{nym}]\}_n, \{\mathsf{nym}\}_n, \{\mathsf{dpk}\}_d) \leftarrow \mathsf{NymKGen}(1^\kappa, 1^n, 1^d)$
  $\mathsf{N} = \{\mathsf{nym}\}_n,$ and $\mathsf{D} = \{\mathsf{dpk}\}_d$
  $\mathsf{W} = \{\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}) \mid \mathsf{nym} \in \mathsf{N}, \mathsf{dpk} \in \mathsf{D}\}$
  $\mathsf{W} \odot \mathsf{D} := \{(\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}), \mathsf{dpk}) \mid \mathsf{nym} \in \mathsf{N}, \mathsf{dpk} \in \mathsf{D}\}$
  $st \leftarrow \mathcal{A}^{\mathsf{Corrupt}}(\mathsf{gpk}, \mathsf{W}, \mathsf{W} \odot \mathsf{D}, \mathsf{N})$
      If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{nym})$ on input $\mathsf{nym} \in \mathsf{N}$
          – set $\mathsf{C} \leftarrow \mathsf{C} \cup \{\mathsf{nym}\}$
          – return $\mathsf{gsk}[\mathsf{nym}]$
  $d \leftarrow \mathcal{A}^{\mathsf{B}', \mathsf{NymSig}', \mathsf{LoR}}(st)$
      If $\mathcal{A}$ queries $\mathsf{B}'(\mathsf{dsnym})$ on input $\mathsf{dsnym} \in \mathsf{W}$
          – set $\mathsf{B} \leftarrow \mathsf{B} \cup \{\mathsf{dsnym}\}$
      If $\mathcal{A}$ queries $\mathsf{NymSig}'(\mathsf{dsnym}, \mathsf{dpk}, m)$ on input $\mathsf{dsnym} \in \mathsf{W} \setminus \mathsf{B}$, $\mathsf{dpk} \in \mathsf{D}$,
      and message $m$
          – set $\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{dsnym}, \mathsf{dpk}, m)\}$
```

```
        – find $\mathsf{nym} \in \mathsf{N}$ such that $\mathsf{dsnym} = \mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk})$
        – return $\mathsf{NymSig}(\mathsf{dsnym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}, m)$
    If $\mathcal{A}$ queries $\mathsf{LoR}(\mathsf{dsnym}_0, \mathsf{dsnym}_1, \mathsf{dpk}, m)$ on input $\mathsf{dsnym}_0, \mathsf{dsnym}_1 \in \mathsf{W} \backslash \mathsf{B}$
    $\mathsf{dpk} \in \mathsf{D}$, and message $m$,
        – set $\mathsf{T} \leftarrow \mathsf{T} \cup \{(\{\mathsf{dsnym}_0, \mathsf{dsnym}_1\}, \mathsf{dpk}, m)\}$
        – find $\mathsf{nym}_0, \mathsf{nym}_1 \in \mathsf{N} \setminus \mathsf{C}$ such that
            $\mathsf{dsnym}_i = \mathsf{NymDSGen}(\mathsf{nym}_i, \mathsf{gsk}[\mathsf{nym}_i], \mathsf{dpk})$ for $i = 0, 1$
        – return $\perp$ if no such $\mathsf{nym}_0, \mathsf{nym}_1$ exist,
            else return $\mathsf{NymSig}(\mathsf{dsnym}_b, \mathsf{gsk}[\mathsf{nym}_b], \mathsf{dpk}, m)$
Return 1 iff
    (a) $d = b$ and
    (b) for any $(\{\mathsf{dsnym}_0, \mathsf{dsnym}_1\}, \mathsf{dpk}, m)$, $(\{\mathsf{dsnym}'_0, \mathsf{dsnym}'_1\}, \mathsf{dpk}, m') \in \mathsf{T}$
        we have either $\{\mathsf{dsnym}_0, \mathsf{dsnym}_1\} = \{\mathsf{dsnym}'_0, \mathsf{dsnym}'_1\}$
        or $\{\mathsf{dsnym}_0, \mathsf{dsnym}_1\} \cap \{\mathsf{dsnym}'_0, \mathsf{dsnym}'_1\} = \emptyset$, and
    (c) for any $(\mathsf{dsnym}, \mathsf{dpk}, m) \in \mathsf{S}$ there is no $\mathsf{dsnym}', m'$
        such that $(\{\mathsf{dsnym}, \mathsf{dsnym}'\}, \mathsf{dpk}, m') \in \mathsf{T}$.
```

The probability is taken over all coin tosses of NymKGen, NymSig, *and* $\mathcal{A}$, *and the choice of* b.

We note that the adversary in our game always accesses oracles through their domain-specific pseudonyms. This is possible since $\mathcal{A}$ knows the set W of such pseudonyms (but not the relation to the pseudonyms nym). Having this list at the outset is motivated by the fact that the adversary can potentially collect such domain-specific pseudonyms when acting as a domain holder, where it gets to learn the domain-specific pseudonyms and signatures under these pseudonyms. Note that this also allows to link domain-specific pseudonyms dsnym to domain keys dpk, hence we give $\mathsf{W} \odot \mathsf{D}$ as additional input. This also implies that we cannot grant the adversary access to another signature oracle which it can provide $\mathsf{nym}, \mathsf{dpk}, m$ to get a signature for m under the corresponding dsnym; it would be easy to check for the validity of the signature under the domain-specific pseudonym with the help of W and to link dsnym to nym. In other words, one can link pseudonyms to their domain-specific pseudonyms given a signature under the pseudonym for the respective domain.

For an adaptive version, the adversary may interleave Corrupt queries with the other oracle queries arbitrarily. Then we must ensure that no nym in a Corrupt query has appeared in an LoR query before, declaring the adversary to lose if this is the case.

2.3 Unforgeability

Unforgeability of domain-specific pseudonymous signatures follows the basic paradigm for regular signatures: It should be infeasible to create a valid signature on behalf of an honest pseudonym for a previously unsigned message. This should even hold if the adversary knows the group manager's secret key (but not the user's secret key, else trivial attacks would be possible). Since for unforgeability we do not need to hide the link between pseudonyms and domain-specific pseudonyms, we assume that the adversary simply knows the tuples $(\mathsf{dsnym}, \mathsf{nym}, \mathsf{dpk})$ of domain-specific pseudonyms, and corresponding pseudonyms and domain keys. Below we say that the adversary wins if it manages to forge a signature under a domain-specific pseudonym dsnym^* which is potentially derived from *some* pseudonym nym in *some* domain dpk; but the adversary does not to specify these values.

Our notion of unforgeability is weaker than the non-frameability property of group signatures in the sense that, even though the adversary may know the group manager's secret key, it must not collaborate with the group manager during generation. We again consider only the version of static corruptions, although here it is straightforward to capture adaptive corruptions by giving the adversary simply the corruption oracle in the second phase, too.

Definition 3 (Unforgeability). *The domain-specific pseudonymous signature scheme* $\mathcal{NYMS} = (\mathsf{NymKGen}, \mathsf{NymDSGen}, \mathsf{NymSig}, \mathsf{NymVf})$ *is* (n, d, t, q, ϵ)*-unforgeable if any algorithm* $\mathcal{A}$*, running in time* t *and making at most* q *signing queries, makes the following experiment output* 1 *with probability at most* ϵ*:*

```
***Experiment*** $\mathsf{Unforge}_{\mathcal{A}}^{\mathcal{NYMS}}(\kappa, n, d)$
    $\mathsf{S}, \mathsf{B}, \mathsf{C}, \mathsf{N}, \mathsf{D} \leftarrow \emptyset$
    $(\mathsf{gpk}, \mathsf{gmsk}, \{\mathsf{gsk}[\mathsf{nym}]\}_n, \{\mathsf{nym}\}_n, \{\mathsf{dpk}\}_d) \leftarrow \mathsf{NymKGen}(1^\kappa, 1^n, 1^d)$
    $\mathsf{N} = \{\mathsf{nym}\}_n$, and $\mathsf{D} = \{\mathsf{dpk}\}_d$
    $\mathsf{W} \odot \mathsf{N} \odot \mathsf{D} := \{(\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}), \mathsf{nym}, \mathsf{dpk}) \mid \mathsf{nym} \in \mathsf{N}, \mathsf{dpk} \in \mathsf{D}\}$
    $st \leftarrow \mathcal{A}^{\mathsf{Corrupt}}(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{W} \odot \mathsf{N} \odot \mathsf{D})$
        If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{nym})$ on input $\mathsf{nym} \in \mathsf{N}$
            – set $\mathsf{C} \leftarrow \mathsf{C} \cup \{\mathsf{nym}\}$
            – return $\mathsf{gsk}[\mathsf{nym}]$
    $(m^*, \sigma^*, \mathsf{dsnym}^*) \leftarrow \mathcal{A}^{\mathsf{B}', \mathsf{NymSig}', \mathsf{Corrupt}}(st)$
        If $\mathcal{A}$ queries $\mathsf{B}'(\mathsf{dsnym})$ on input $\mathsf{dsnym} \in \mathsf{W}$
            – set $\mathsf{B} \leftarrow \mathsf{B} \cup \{\mathsf{dsnym}\}$
        If $\mathcal{A}$ queries $\mathsf{NymSig}'(\mathsf{dsnym}, \mathsf{dpk}, m)$ on input $\mathsf{dsnym} \in \mathsf{W} \setminus \mathsf{B}$, $\mathsf{dpk} \in \mathsf{D}$,
        and message $m$
            – set $\mathsf{S} \leftarrow \mathsf{S} \cup \{(\mathsf{dsnym}, \mathsf{dpk}, m)\}$
            – find $\mathsf{nym} \in \mathsf{N}$ such that $\mathsf{dsnym} = \mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk})$
            – return $\mathsf{NymSig}(\mathsf{dsnym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}, m)$
    Output 1 iff there are $\mathsf{nym}^* \in \mathsf{N} \setminus \mathsf{C}$ and $\mathsf{dpk}^* \in \mathsf{D}$ such that
        (a) $\mathsf{NymDSGen}(\mathsf{nym}^*, \mathsf{gsk}[\mathsf{nym}^*], \mathsf{dpk}^*) = \mathsf{dsnym}^*$, and
        (b) $\mathsf{NymVf}(\mathsf{gpk}, \mathsf{dsnym}^*, \mathsf{dpk}^*, m^*, \sigma^*, \mathsf{B}) = 1$, and
        (c) $(\mathsf{dsnym}^*, \mathsf{dpk}^*, m^*) \notin \mathsf{S}$.
```

The probability is taken over all coin tosses of $\mathsf{NymKGen}$*,* NymSig*, and* $\mathcal{A}$*.*

Note that conditions (a) and (c) also imply that domain-specific pseudonyms of different users (within a domain) cannot collide, except with negligible probability. Otherwise, the adversary may corrupt one of the two parties, and any signature created under the domain-specific pseudonym of the one party would immediately constitute a forgery under the other party's pseudonym. (In the above model it would then be more appropriate to let the adversary in calls to NymSig' also specify nym, instead of searching for it; since the adversary knows the list $\mathsf{W} \odot \mathsf{N} \odot \mathsf{D}$ it can look this value up.)

2.4 Seclusiveness

Seclusiveness finally considers the case that the verifier would accept a signature under a domain-specific pseudonym which has not been created by the authority. Note that this assumes that only the black-listed domain-specific pseudonyms are available, but not the universe of all created pseudonyms. This is indeed a valid assumption, following the suggestion in [7] about revocation for pseudonyms through black-lists, with no intention for white-lists. It is also clear that, unlike in case of unforgeability, we thus cannot allow the adversary to know the manager's secret key; else generating keys for additional users would be easy.

As in unforgeability, we again consider only the version of static corruptions. One captures adaptive corruptions by giving the adversary simply the corruption oracle in the second phase, too.

Definition 4 (Seclusiveness). *The domain-specific pseudonymous signature scheme* $\mathcal{NYMS} = (\mathsf{NymKGen}, \mathsf{NymDSGen}, \mathsf{NymSig}, \mathsf{NymVf})$ *is* (n, d, t, Q, ϵ)*-secluding with* $Q = (q_c, q_s)$ *if any algorithm* $\mathcal{A}$*, running in time* t *and making at most* q_s *signing queries and* q_c *corruption queries, makes the following experiment output* 1 *with probability at most* ϵ*:*

```
Experiment $\mathsf{Sec}_{\mathcal{A}}^{\mathcal{NYMS}}(\kappa, n, d)$
    $\mathsf{W}, \mathsf{B}, \mathsf{C}, \mathsf{N}, \mathsf{D} \leftarrow \emptyset$
    $(\mathsf{gpk}, \mathsf{gmsk}, \{\mathsf{gsk}[\mathsf{nym}]\}_n, \{\mathsf{nym}\}_n, \{\mathsf{dpk}\}_d) \leftarrow \mathsf{NymKGen}(1^\kappa, 1^n, 1^d)$
    $\mathsf{N} = \{\mathsf{nym}\}_n$ and $\mathsf{D} = \{\mathsf{dpk}\}_d$
    $\mathsf{W} = \{\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}) \mid \mathsf{nym} \in \mathsf{N}, \mathsf{dpk} \in \mathsf{D}\}$
    $\mathsf{W} \odot \mathsf{N} \odot \mathsf{D}$
      $:= \{(\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}), \mathsf{nym}, \mathsf{dpk}) \mid \mathsf{nym} \in \mathsf{N}, \mathsf{dpk} \in \mathsf{D}\}$
    $st \leftarrow \mathcal{A}^{\mathsf{Corrupt}}(\mathsf{gpk}, \mathsf{W} \odot \mathsf{N} \odot \mathsf{D})$
        If $\mathcal{A}$ queries $\mathsf{Corrupt}(\mathsf{nym})$ on input $\mathsf{nym} \in \mathsf{N}$
            – set $\mathsf{C} \leftarrow \mathsf{C} \cup \{\mathsf{nym}\}$
            – return $\mathsf{gsk}[\mathsf{nym}]$
    $(m^*, \sigma^*, \mathsf{dsnym}^*) \leftarrow \mathcal{A}^{\mathsf{B}', \mathsf{NymSig}'}(st)$
        If $\mathcal{A}$ queries $\mathsf{B}'(\mathsf{dsnym})$ on input $\mathsf{dsnym} \in \mathsf{W}$
            – set $\mathsf{B} \leftarrow \mathsf{B} \cup \{\mathsf{dsnym}\}$
        If $\mathcal{A}$ queries $\mathsf{NymSig}'(\mathsf{dsnym}, \mathsf{dpk}, m)$ on $\mathsf{dsnym} \in \mathsf{W} \setminus \mathsf{B}$, $\mathsf{dpk} \in \mathsf{D}$,
        and message $m$
            – find $\mathsf{nym} \in \mathsf{N}$ such that $\mathsf{dsnym} = \mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk})$
            – return $\mathsf{NymSig}(\mathsf{dsnym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}, m)$
    Output 1 iff there exists $\mathsf{dpk}^* \in \mathsf{D}$ such that
        (a) $\mathsf{NymVf}(\mathsf{gpk}, \mathsf{dsnym}^*, \mathsf{dpk}^*, m^*, \sigma^*, \mathsf{B}) = 1$
        (b) $\mathsf{dsnym}^* \notin \mathsf{W}$
```

The probability is taken over all coin tosses of $\mathsf{NymKGen}$*,* NymSig*, and* $\mathcal{A}$*.*

3 Construction

The idea of the discrete-log based construction is as follows: The group manager will hold two generators $g_1, g_2 = g_1^z$ with $z \in \mathsf{gmsk}$ for which it knows the discrete log with respect to each other. In addition, it will hold a public key $y = g_1^x$, such that it can easily compute many pairs (x_1, x_2) such that $y = g_1^{x_1} g_2^{x_2}$ with the help of z; this is the trapdoor property of such values [1, 18]. Each user pseudonym nym will receive one of these pairs as its secret key $\mathsf{gsk}[\mathsf{nym}]$. The domain parameters are given by values $\mathsf{dpk} = g^r$. A user can then compute the domain-specific pseudonym as $\mathsf{dsnym} = \mathsf{dpk}^{x_1}$.

To sign a message the user can then use common discrete-log based protocols (in the random oracle model) to show that (a) it knows the discrete-log x_1 of dsnym with respect to dpk, and (b) it knows a matching value x_2 to this discrete logarithm x_1 such that the pair forms a representation of y. Essentially, this is accomplished by running the non-interactive version of the Okamoto proof of knowledge [17] for x_1, x_2 and y to base g_1, g_2, where the x_1-part can simultaneously be used to show knowledge of x_1 of dsnym with respect to base dpk. As usual, the message to be signed enters the hash computations.

Construction 1. *The construction of the domain-specific pseudonymous signature scheme* $\mathcal{NYMS} = (\mathsf{NymKGen}, \mathsf{NymDSGen}, \mathsf{NymSig}, \mathsf{NymVf})$ *is as follows:*

$\mathsf{NymKGen}(1^\kappa, 1^n, 1^d)$**:** *Let* $\mathcal{G} = \langle g \rangle$ *be a (public) cyclic group of prime order* q*. We also assume a public hash function* H*, modeled as a random oracle in the security proofs. Choose* $z \in_R \mathbb{Z}_q$ *randomly and calculate* $g_1 := g$ *and* $g_2 := g^z$*. Define* $\mathsf{gpk} := g_1^x$ *for random* $x \in_R \mathbb{Z}_q$*. To generate the secrets for the pseudonyms choose* n *random elements* $x_{2,1}, \ldots, x_{2,n} \in_R \mathbb{Z}_q^*$ *and calculate* $x_{1,i} = x - z \cdot x_{2,i}$ *for* $i = 1, 2, \ldots, n$*. Define* $\mathsf{gsk}[i] := (x_{1,i}, x_{2,i})$*. By* x_j *we denote the* $x_{j,i}$ *when pseudonym* i *is clear from context. For the domain-parameters pick random* $r_1, \ldots, r_d \in_R \mathbb{Z}_q^*$ *and define* $\mathsf{dpk}_i := g^{r_i}$ *for* $i = 1, \ldots, d$*. Store* z *in* gmsk*. (Note that once the values* $\mathsf{gsk}[\cdot]$ *have been output resp. given to the users, the group manager deletes them.)*

$\mathsf{NymDSGen}(\mathsf{nym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk})$**:** *Compute and output the domain-specific pseudonym* $\mathsf{nym}[\mathsf{dpk}] := \mathsf{dpk}^{x_1}$*, which is also sometimes denoted as* dsnym *when* nym *and* dpk *are known from context.*

$\mathsf{NymSig}(\mathsf{dsnym}, \mathsf{gsk}[\mathsf{nym}], \mathsf{dpk}, m)$**:** *Let* $a_1 = g_1^{t_1} \cdot g_2^{t_2}$ *and* $a_2 = \mathsf{dpk}^{t_1}$*, for random* $t_1, t_2 \in_R \mathbb{Z}_q$*. Compute* $c = H(\mathsf{dpk}, \mathsf{dsnym}, a_1, a_2, m)$*. Let* $s_1 = t_1 - c \cdot x_1$ *and* $s_2 = t_2 - c \cdot x_2$*. Then, output* $\sigma = (c, s_1, s_2)$*. (Note that in the Restricted-Identification protocol the user also sends* dsnym *which we can include here in the signature, in order to match the protocol description.)*

$\mathsf{NymVf}(\mathsf{gpk}, \mathsf{dsnym}, \mathsf{dpk}, m, \sigma, \mathsf{B})$**:** *To verify a signature perform the following steps:*

1. *Parse* $(c, s_1, s_2) \leftarrow \sigma$*.*
2. *Let* $a_1 = y^c \cdot g_1^{s_1} \cdot g_2^{s_2}$ *and* $a_2 = \mathsf{dsnym}^c \cdot \mathsf{dpk}^{s_1}$*.*
3. *Output* 1 *iff* $c = H(\mathsf{dpk}, \mathsf{dsnym}, a_1, a_2, m)$ *and* $\mathsf{dsnym} \notin \mathsf{B}$*.*

Revocation Mechanisms. We presented our construction above in terms of black-listing, revoking fraudulent domain-specific pseudonyms by listing them explicitly. Alternatively, and our definitions and constructions are robust in this regard, one can use a white-listing approach to list valid entries only. To represent the whitelisting approach in our framework any delisted domain-specific pseudonym from W will be put in B, such that $\mathsf{W} \setminus \mathsf{B}$ corresponds to the set of currently whitelisted entries. Checking for whitelisting thus corresponds to verifying that the entry is in $\mathsf{W} \setminus \mathsf{B}$ in our framework.

Blacklisting and whitelisting is performed by calculating the domain-specific pseudonym dsnym for domain dpk, but without knowledge of the private keys x_1 and x_2. One important difference between black- and whitelisting is that whitelisting allows to retain security even if authority's secret key z is compromised. If whitelisting is used it is not strictly required to keep z secret. While an attacker would be able to construct valid private keys (x_1, x_2) corresponding to the group public key y, the corresponding pseudonyms would not be listed on the whitelist and thus, the signatures would be rejected. Therefore, the attacker would have to find corresponding private keys for given pseudonyms on the whitelist, which in turn would require to calculate discrete logarithms.

4 Security Analysis

Our proofs work in the random oracle model, and thus, an adversary may also query a random hash function oracle. By q_h, we denote the maximum number of queries to hash function oracle made by the adversary.

4.1 Number-Theoretic Assumptions

Our proof for the anonymity follows by reduction to the Decisional Diffie-Hellman (DDH) assumption whereas the hardness of unforgeability and seclusiveness is reduced to the Discrete Logarithm (DL) problem. Both are considered as standard assumptions, and are believed to be hard for decades. Roughly, the DDH assumption says that distinguishing the tuples (g^a, g^b, g^{ab}) and (g^a, g^b, g') where g' is picked randomly from all group elements, is hard. The DL assumption says it is hard to find a given group $\mathcal{G}$ generated by g and element $A = g^a$. In our security analysis we use concrete bounds for the underlying problems, i.e., we assume that the Discrete-Logarithm problem is (t, ϵ)-hard if any algorithm $\mathcal{A}$ given $A \in \mathcal{G}$ (sampled uniformly in $\mathcal{G}$), running in time t, finds a with $g^{a'} = A$ with probability at most ϵ. Second, the Decisional Diffie-Hellman problem is (t, ϵ)-hard if any algorithm $\mathcal{A}$ can differentiate tuples (g^a, g^b, g^c) and (g^a, g^b, g^{ab}) with probability at most $\frac{1}{2} + \epsilon$ in runtime t where $a, b, c \in \mathbb{Z}_{\varphi(\mathcal{G})}$ are sampled uniformly.

4.2 Anonymity

Informally, the protocol is cross-domain anonymous (with respect to static corruptions) because the domain-specific pseudonyms appear to be random to the adversary under the decisional Diffie-Hellman assumption. Below we write $t' \approx t$ to denote the fact that t' is essentially the same running time as t, except for minor administrative overhead.

Theorem 1. *Assume the DDH problem is (t, ϵ)-hard. Then the domain-specific pseudonymous signature scheme $\mathcal{NYMS}$ of Section [3] is (n, d, t', Q, ϵ') cross-domain anonymous with $Q = (q_c, q_s, q_t, q_h)$, where $\epsilon' \leq nd\epsilon + \frac{(q_s+q_t+q_h)(q_s+q_t)}{q^2}$ and $t' \approx t$. This holds in the random oracle model.*

Proof. We combine the ideas of pseudorandom synthesizers of Naor and Reingold [16] with the simulation soundness of the non-interactive zero-knowledge proofs (aka. signatures) in the random oracle model. That is, assume the original attack of the adversary on our protocol. In a first game hop we replace the actual signature computations in LoR and NymSig' queries by simulated signatures (via programming the random oracle). See [19] for details. This strategy is valid if programming the hash values in such computations have not appeared in previous hash queries of the adversary, nor in previous signature queries. However, since the random group elements a_1, a_2 enter the hash evaluations, the probability that an input collision occurs in any of the at most $q_s + q_t$ signature generations, is at most $(q_s + q_t + q_h)(q_s + q_t)/q^2$. Given that no such collision occurs the simulation is perfectly indistinguishable such that the adversary's success probability cannot increase by more than this term.

Note that after the game hop, we can create valid signatures on behalf of users without knowing the secret keys. In the next game hop, we replace the domain-specific pseudonyms dsnym in LoR queries by random group elements (but in a consistent way). That is, whenever we are supposed to use dsnym we instead use a new random element $\mathsf{dsnym}' \leftarrow \mathcal{G}$, unless nym in combination with dpk has been used before, either in a signature request or an LoR query, in which case we use again the previously generated random value dsnym'.

We claim that this hop cannot increase the adversary's success probability noticeable by the DDH assumption. To this end, we briefly recall the notion of a pseudorandom synthesizer in [16]. The pseudorandom synthesizers for the DH pairs is an $a \times b$ matrix, with the rows labeled by values g^{x_i} and the columns labeled by g^{r_j}, and entries at position i, j set to $g^{x_i r_j}$, such that this matrix is indistinguishable from an $a \times b$ matrix of independent and random group elements, even if the row and column labels g^{x_i} and g^{r_j} are given. In a sense, the matrix entries are correlated but still look random. As discussed in [16] this holds in our case under the DDH assumption. In fact, it allows for a reduction to the DDH problem with a loss of a factor ab where, in our case, after the initial corruption, there are at most $ab \leq nd$ entries of honest users.

In the next game hop, we always use the left domain-specific pseudonym dsnym_0 in LoR queries, independently of the value of b. We stress that, in case of $b = 1$, this does not change the adversary's success probability at all. Assume from now on $b = 1$. Note that each LoR query about $(\mathsf{dsnym}_0, \mathsf{dsnym}_1, \mathsf{dpk}, m)$ is answered by a random element, just as it would be for $b = 0$. All other LoR queries involving dpk can only be about the same pair $(\mathsf{dsnym}_0, \mathsf{dsnym}_1)$ in this order, in reverse order $(\mathsf{dsnym}_1, \mathsf{dsnym}_0)$, or for distinct entries. In the first case, we would answer again consistently with the (wrong) random element, in the second case, we would switch to the other random element, and in the third case use an independent random value. This behavior, however, is identical to the case $b = 0$ from the adversary's point of view. Similarly, the adversary cannot make any signature request for $(\mathsf{dsnym}_0, \mathsf{dpk})$ nor $(\mathsf{dsnym}_1, \mathsf{dpk})$ without losing. It follows that such signature requests do not depend on the bit b. Hence, the probability of the experiment returning 1 does not change.

In the final game, the adversary's success probability is independent of b, and the adversary cannot win with probability more than $\frac{1}{2}$. Collecting all probabilities from the game hops yields the claimed bound. □

Anonymity of Restricted Identification. Recall that in the basic version of the restricted identification protocol, the user merely shows the domain-specific pseudonym dsnym to the service provider (who checks that this value has not been revoked yet). Anonymity of this solution follows from the proof above under the DDH assumption alone, noting that we do not need to simulate the additional signatures in the random oracle model.[2]

4.3 Unforgeability

Theorem 2. *Assume the DL problem is (t, ϵ)-hard on $\mathcal{G}$, then the domain-specific pseudonymous signature scheme $\mathcal{NYMS}$ of Section 3 is (n, d, t', Q, ϵ')-unforgeable with $Q = (q_s, q_h)$, where*

$$\epsilon' \approx (2q)^{-1}(q_h - \sqrt{q_h}\sqrt{\delta\epsilon + (2\delta/q)(q_s + q_h)^2 + q_h})$$

and $t' \approx t$ with $\delta = 4ndq^2$. This holds in the random oracle model.

Due to space limitations, the proof of Theorem 2 appears in the full version of this work; however, we provide a sketch here.

Proof (Sketch). We reduce the hardness of the unforgeability game (cf. Definition 3) to the discrete-logarithm problem. We embed a DL challenge $A = g^a$ into a domain-specific pseudonym dsnym^* of an randomly chosen pseudonym $\mathsf{nym}^* \in \mathsf{N}$ and domain $\mathsf{dpk}^* \in \mathsf{D}$. Similarly to the proof of Schnorr signatures [19], we leverage the Forking Lemma, in order to obtain two related forgeries $(c, s_1, s_2), (c', s_1', s_2')$ on a message m under dsnym^* from which we can extract the witness (resp. discrete logarithm) from the challenge as follows. Given both signatures σ, σ' we extract the discrete-log a of $\mathsf{dsnym}^* = A = g^a$ as follows. Given $s_1 = t_1 - c \cdot a$ and $s_1' = t_1 - c' \cdot a$, we have $a = (s_1 - s_1')/(c' - c)$. Hence, we found the solution a for the DL instance A.

We require that $\mathcal{A}$ succeeds to forge on behalf of pseudonym nym^* under domain public key dpk^* in the first signature. In addition, we loose a tightness

[2] The specification actually lets the user send a hash value of dsnym. This does not affect the discussion, though.

factor due to Forking Lemma, which yields the probability to find the discrete logarithm of A at most $\epsilon = \epsilon'/nd \cdot (\epsilon'/q_h + 1/q) - (2q_s + 2q_h)^2/q$ where ϵ' is the success probability of $\mathcal{A}$.

4.4 Seclusiveness

As remarked before, seclusiveness only holds as long as the adversary does not get a hold of the group manager's secret key. By construction, this means that the adversary can thus only corrupt one user, else z becomes known. When considering blacklisting for our construction, we stipulate this below by requiring that the secrets are stored securely in hardware, or, respectively, that the number of corrupt requests q_c is at most 1. If whitelisting is used instead, then we do not require any bound on the number of corruptions the adversary can made, since learning z does not help the adversary to compute a domain-specific dsnym which is listed in the (still trustworthy) whitelist.

Theorem 3. *Assume the DL problem is (t, ϵ)-hard on $\mathcal{G}$, then the domain-specific pseudonymous signature scheme of Section 3 is (n, d, t', Q, ϵ')-secluding with $Q = (q_c, q_s, q_h)$, where $q_c = 1$, $\epsilon' \approx (2q)^{-1}(q_h - \sqrt{q_h}\sqrt{q_h + 4q^2\epsilon + 4q^2\delta})$ and $t' \approx t$ with $\delta = 2(q_s + q_h)/q$. This holds in the random oracle model.*

Due to space limitations, the proof of Theorem 2 appears in the full version of this work; however, we provide a sketch here.

Proof (Sketch). We are given an adversary $\mathcal{A}$ which wins in the seclusiveness game of $\mathcal{NYMS}$. Here, $\mathcal{A}$ outputs a signature under a domain-specific pseudonym dsnym to no corresponding identity $\mathsf{nym} \in \{\mathsf{nym}\}_d$. Intuitively, the proof works as follows. We are asked for the discrete-log a of an element $A := g_{dl}^a$ from a presumably DL-hard group $\mathcal{G}_{dl}$. We embed A in generator g_2 such that the group's master key $z \in \mathsf{gmsk}$ equals a. We are able to generate one secret key pair (x_1, x_2) satisfying $x = x_1 + zx_2$ for unknown x, gmsk. Using the signature given by $\mathcal{A}$ we can extract a second pair (x_1^*, x_2^*) . Those two key pairs suffice to disclose z (resp. a) and thus, we solve the DL problem. □

References

1. Brassard, G., Chaum, D., Crépeau, C.: Minimum disclosure proofs of knowledge. J. Comput. Syst. Sci. 37(2), 156–189 (1988)
2. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N.P., Warinschi, B.: Anonymous attestation with user-controlled linkability. Cryptology ePrint Archive, Report 2011/658 (2011), http://eprint.iacr.org/
3. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and A Construction Based on General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
4. Bender, J., Fischlin, M., Kügler, D.: Security Analysis of the PACE Key-Agreement Protocol. In: Samarati, P., Yung, M., Martinelli, F., Ardagna, C.A. (eds.) ISC 2009. LNCS, vol. 5735, pp. 33–48. Springer, Heidelberg (2009)

5. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy. The MIT Press (2000)
6. Brzuska, C., Dagdelen, Ö., Fischlin, M.: TLS, PACE, and EAC: A Cryptographic View at Modern Key Exchange Protocols. In: GI-Sicherheit 2012. GI-LNI, pp. 71–82 (2012)
7. Advanced security mechanism for machine readable travel documents extended access control (eac). Technical Report (BSI-TR-03110) Version 2.05 Release Candidate, BSI (2010)
8. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM 28 (October 1985)
9. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
10. Camenisch, J., Lysyanskaya, A.: Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Heidelberg (2002)
11. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
12. Dagdelen, Ö., Fischlin, M.: Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents. In: Burmester, M., Tsudik, G., Magliveras, S., Ilić, I. (eds.) ISC 2010. LNCS, vol. 6531, pp. 54–68. Springer, Heidelberg (2011)
13. Fischlin, M.: Anonymous Signatures Made Easy. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 31–42. Springer, Heidelberg (2007)
14. Koops, C.f.B.J., Buitelaar, H., Lips, M. (eds.): D5.4: Anonymity in electronic government: a case-study analysis of governments? identity knowledge. FIDIS report (February 2012)
15. Kutylowski, M., Shao, J.: Signing with multiple ID's and a single key. In: 38th CCNC, pp. 519–520. IEEE (2011)
16. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS, pp. 458–467. IEEE (1997)
17. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 31–53. Springer, Heidelberg (1993)
18. 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)
19. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13(3), 361–396 (2000)
20. Rivest, R.L., Shamir, A., Tauman, Y.: How to Leak a Secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001)
21. Schnorr, C.-P.: Efficient signature generation by smart cards. Journal of Cryptology 4(3), 161–174 (1991)
22. Wei, V.K.: Tracing-by-Linking Group Signautres. Cryptology ePrint Archive, Report 2004/370 (2004), http://eprint.iacr.org/
23. Yang, G., Wong, D.S., Deng, X., Wang, H.: Anonymous Signature Schemes. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 347–363. Springer, Heidelberg (2006)

Solutions for the Storage Problem of McEliece Public and Private Keys on Memory-Constrained Platforms

Falko Strenzke*

FlexSecure GmbH, Germany
strenzke@flexsecure.de

Abstract. While it is generally believed that due to their large public and private key sizes code based public key schemes like the McEliece PKC cannot be conveniently implemented on memory-constrained devices, we demonstrate otherwise. We show that for the public key we face rather a transmission problem than a storage problem: we propose an approach for Public Key Infrastructure (PKI) scenarios which totally eliminates the need to store public keys of communication partners. Instead, all the necessary computation steps are performed during the transmission of the key. We show the feasibility of the approach through an example implementation and give arguments that it will be possible for a smart card controller to carry out the associated computations fast enough to sustain the transmission rates of possible future high speed contactless interfaces. Concerning the McEliece private key, we demonstrate, contrasting to previously published implementations, that the parity check matrix, which is by far the largest part of this key, is not necessary to achieve fast decryption on embedded systems.

Keywords: post-quantum cryptography, code-based cryptography, public key encryption scheme, efficient implementation, embedded devices.

1 Introduction

Code-based cryptography, i.e. the class of cryptographic schemes built on error correcting codes, encompasses public key encryption schemes [1,2] as well as signature schemes [3,4] and an identification scheme [5]. The main advantage of code-based cryptographic schemes over currently used schemes that are based on the factoring or discrete logarithm problem is their believed security in the presence of quantum computers [6], but at least the encryption schemes' operations can also be implemented comparatively fast [7].

However, the large public key size in these schemes are considered a tremendous disadvantage. For this reason, a number of attempts have been made to

* A part of the work of F. Strenzke was done at Cryptography and Computeralgebra, Department of Computer Science, Technische Universität Darmstadt, Germany.

reduce the public key size by using special codes [8,9,10]. But some of these attempts have already been shown to result in insecure cryptosystems [11,12,13]. All recent proposals that reduce the key size using other codes than in the original McEliece scheme will have to prevail for some time until they can be granted the same trust as the original scheme, employing classical binary Goppa codes [14], the security of which is still unquestioned after more than 30 years – however, with the exception of certain choices of code parameters [15], which are only of relevance for code-based signature schemes [3].

Accordingly, in the first part of the work, we address the problem of performing the public operations, i.e. encryption or signature verification, of conventional code-based public key cryptosystems with large public keys on devices with limited memory resources, like for instance smart cards. Typically, smart cards have less than 20 KB of RAM, while the available amount of non-volatile memory (NVM), e.g. flash-memory, can be as large as 512 KB [16,17]. If a public key of a communication partner shall be temporarily stored on the device for the purpose of performing e.g. an encryption, it would have to be stored in the NVM since it exceeds the size of the RAM many times over. Specifically, the public keys will be at least 100 KB large for reasonable security parameters, as we will see in Section 2.2. For instance, the works [18,19,20] all describe implementations of code-based encryption schemes on embedded devices, where the public key is stored in the devices NVM. The drawbacks of storing such an amount of data in the device's NVM are first of all the cost of keeping such a large amount of memory available for this purpose and also the much slower writing speed compared to RAM access. In order to circumvent these problems, we show in this work that the public operations can be executed by only storing very small parts of the public keys at any given time during the operation. Our approach also considers that these operations are always carried out in a PKI context, which implies the verification of user public key certificates against issuer certificates. We point out that this approach is equally usable for the Niederreiter PKC as well as certain code-based signature schemes.

The second part of this work shows a solution for the McEliece private key, which in previously published implementations [21,18,19,22] of the McEliece PKC features the parity check matrix, the size of which is similar to that of the McEliece public key. The benefit of the parity check matrix, which is not an essential part of the private key, but a precomputation that allows faster computation of the so called syndrome vector. We show that an optimized implementation of the syndrome computation allows practical decryption times without this matrix, reducing the private key size dramatically.

2 Preliminaries

2.1 Public Key Cryptography

In a public key infrastructure, the trustworthiness of a public key is always verified against a trust anchor. From the trust anchor, which is usually a certification authority (CA) certificate, to the user certificate, there is a certificate chain involved. The trustworthiness of a certificate lower in the chain is guaranteed by its authentic digital signature created by the respective issuer, verifiable via the corresponding public key contained in the issuer certificate.

For the case of public key encryption, it means that a user A's public key intended for encryption is contained in the user certificate. A user B willing to encrypt a message for A thus goes through the following steps:

1. retrieve A's public encryption certificate Enc-Cert_A (for example by accessing a database or asking A directly)
2. verify the authenticity of Enc-Cert_A by checking the signature on the certificate against the trust anchor (CA certificate)
3. encrypt the secret message using Enc-Cert_A and send it to A

Since in this work we will address problems and solutions for embedded devices such as smart cards, we wish to point out why it is necessary to be able to carry out not only the private operations of a public key scheme (i.e. decryption or signature generation) but also the public operations on such devices. One application are key exchange schemes. Key exchange schemes based on public key cryptography are used for instance in the context of the German ePassport [23]. There, an elliptic curve based key agreement scheme is realized [24]. In order to replace this scheme with a quantum computer secure solution, one would have to combine a public key encryption scheme with a public key signature scheme that both have this property. Then, one party sends the signed and encrypted symmetric key to the other party. In the mentioned context this means that eventually the ePassport's chip has to carry out the encryption operation.

2.2 Code-Based Encryption Schemes

In the following, we explain two code-based encryption schemes, where we focus on the encryption operation and those parts of the decryption operation that are relevant for the understanding of this work.

Both code-based encryption schemes employ irreducible binary Goppa Codes [14] as error correcting codes.

Definition 1. *Let the polynomial $g(Y) = \sum_{i=0}^{t} g_i Y^i \in \mathbb{F}_{2^m}[Y]$ be monic and irreducible over $\mathbb{F}_{2^m}[Y]$, and let m, t be positive integers. Then $g(Y)$ is called a* Goppa polynomial *(for an irreducible binary Goppa code).*

Then an irreducible binary Goppa code is defined as $\mathcal{C}(\Gamma, g(Y)) = \{\boldsymbol{c} \in \mathbb{F}_2^n | S_{\boldsymbol{c}}(Y) := \sum_{i=0}^{n-1} \frac{c_i}{Y-\alpha_i} = 0 \mod g(Y)\}$, where $n \leqslant 2^m$, $S_{\boldsymbol{c}}(Y)$ is the syndrome of $\boldsymbol{c}$, $\Gamma = \{\alpha_i | i = 0, \ldots, n-1\}$, the support *of the code, where the α_i are pairwise distinct elements of $\mathbb{F}_{2^m}$, and c_i are the entries of the vector $\boldsymbol{c}$.*

The code defined in such way has length n (i.e. the length of the code words), dimension $k \geqslant n - mt$ (i.e. the length of the message words) and can correct up to t bit flip errors.

As for any linear error correcting code, for a Goppa code there exists a generator matrix $G \in \mathbb{F}_2^{n \times k}$ and a parity check matrix $H \in \mathbb{F}_2^{mt \times n}$ [25]. Given these matrices, a message $\boldsymbol{v} \in \mathbb{F}_2^k$ can be encoded into a codeword $\boldsymbol{c}$ of the code by computing $\boldsymbol{c} = \boldsymbol{v}G$, and the syndrome $\boldsymbol{s} \in \mathbb{F}_2^{mt}$ of a (potentially distorted) codeword can be computed as $\boldsymbol{s} = \boldsymbol{c}H^T$. Here, we do not give the formulas for the computation of these matrices as they are of no importance for the understanding of the topics of this work. The interested reader, however, is referred to [25].

The first encryption scheme we present is the McEliece [1] scheme.

Overview of the McEliece PKC. In this section we give a brief overview of the McEliece PKC.

Algorithm 1. The McEliece encryption Operation

```
Require: the McEliece public key $G_p \in \mathbb{F}_2^{k \times n}$ and the message $m \in \mathbb{F}_2^k$,
Ensure: the ciphertext $z \in \mathbb{F}_2^n$
1: create a random binary vector $e \in \mathbb{F}_2^n$ with Hamming weight $\mathrm{wt}(e) = t$
2: $z \leftarrow mG_p \oplus e$
```

The McEliece *secret key* consists of the Goppa polynomial $g(Y)$ of degree t and the support Γ, together they define the secret code $\mathcal{C}$.

The *public key* is given by the public $n \times k$ generator matrix $G_p = TG$ over $\mathbb{F}_2$, where G is a generator matrix of the secret code $\mathcal{C}$ and T is a non-singular $k \times k$ matrix over $\mathbb{F}_2$, the purpose of which is to bring G_p into reduced row echelon form, i.e. $G_p = [\mathbb{I}|G_2]$, which results in a more compact public key [7]. The *encryption* operation (Algorithm 1) allows messages $\boldsymbol{v} \in \mathbb{F}_2^k$. A random vector $\boldsymbol{e} \in \mathbb{F}_2^n$ with hamming weight $\mathrm{wt}(\boldsymbol{e}) = t$ has to be created. Then the ciphertext is computed as $\boldsymbol{z} = \boldsymbol{v}G_p + \boldsymbol{e}$. Note that due to the reduced row echelon form of G_p the first k bits of $\boldsymbol{v}$ are reproduced in $\boldsymbol{v}G_p$. However, since the McEliece PKC, as well as the Niederreiter PKC, which is introduced shortly, needs to be wrapped in a so-called CCA2 conversion [26], this is not a problem [7].

McEliece *decryption* is performed by applying error correction to the ciphertext $\boldsymbol{z}$, which can only be done with the knowledge of the secret key, i.e. the code $\mathcal{C}$. With the exception of the first step, which is the computation of the syndrome polynomial $S(Y)$, which will be the topic of Section 4 and will be explained there, the further details of the decryption procedure are irrelevant for the understanding of the topics of this work.

The McEliece parameters are given by the code parameters n, k and t. An example parameter set giving about 100 bits of security with respect to the attacks given in [27] would be be $n = 2048$, $k = 1498$ and $t = 50$, yielding a public key size of about 100 KB.

The other encryption scheme is the Niederreiter [2] scheme. Here, the public key consists of the public parity check matrix $H_p = TH_s$, where H_s is the parity check matrix of the private code and $H_p \in \mathbb{F}_2^{(n-k) \times n}$, and T is chosen equivalently to its counterpart in the McEliece scheme. Furthermore, as in the McEliece scheme, H_p can be put in systematic form. Then the public key will be of the same size as for the McEliece cryptosystem. The Niederreiter encryption is depicted in Algorithm 2. The message is encoded into an error vector of weight t and the ciphertext is the corresponding syndrome, which can only be decoded by the holder of the private key.

Algorithm 2. The Niederreiter encryption Operation

```
Require: the Niederreiter public key $H \in \mathbb{F}_2^{(n-k)\times n}$ and the message $m$
Ensure: the ciphertext $z \in \mathbb{F}_2^{n-k}$
1: encode the message $m$ into $e \in \mathbb{F}_2^n$, where $\mathrm{wt}(e) = t$, using an appropriate algorithm
   ("constant-weight-word encoding")
2: $z \leftarrow eH$
```

3 Online Public Operation

In this section, we explain the main idea of the paper, namely how to implement the public operations of code-based schemes without storing full public keys on the device. In a naive approach, the public operation, which we here assume to be an encryption operation, would be realized by first retrieving the public key (embedded into a public key certificate containing also a signature) of the communication partner, storing it on the device, computing the hash value of the certificates to-be-signed (TBS) data (which includes the code-based public key), verifying the signature, and finally encrypting the designated message using the certificate's public key. With the proposed approach however, no storage of the whole public key is required. Instead, only a comparatively small amount of RAM memory will be used. The basic idea is to use the computation time that is available to the devices CPU in the time interval between the receival of two bytes via the serial interface. During this interval both the encryption algorithm and the hash algorithm are advanced by one small step. Hence we call this approach "on-line public operation".

This approach works because both the computation of the hash value of the public key and the matrix-vector product only depend on a small part of the whole public key at any given point in time: while the hash function acts on blocks of multiple bytes (for instance 64 bytes for SHA-256), the matrix multiplication could in principle be carried out bit-wise.

3.1 Description of the Online Public Operation

In Figure 1, the complete process of the on-line public operation approach is depicted. On the left hand side, the processing of the certificate containing the code-based public key to be used in the public operation is shown. Here, we assume that the public key is contained in an X.509 public key certificate [28]. Such a certificate is constituted by the sequence of the TBS data, followed by a field containing information about the signature algorithm (not shown in the figure) and finally the signature. The signature ensures the authenticity of the TBS data, and is calculated based on their hash value, using a hash algorithm as specified in the preceding information field. Please note that the signature algorithm used to sign the user certificate needs not to be code-based (in which case the trust anchor certificate would contain a large code-based key itself). Instead, a hash based signature scheme [29] could be used. These schemes are also quantum computer resistant and feature extremely small public keys.

In Step 1a the part of the TBS data that precedes the public key is received by the device and processed in the normal manner, which includes the computation of the hash value of the received data. Once the transmission of the public key, i.e. the public matrix M, begins (Step 2a), the computation of the product vM begins, where v is a binary vector whose meaning depends on the type of the code-based scheme. In an encryption scheme like McEliece or Niederreiter, v represents a message. The hash computation is also continued. After the whole public matrix has been received, the remaining TBS data is again processed in the normal manner (Step 3a). Finally, when the TBS data have been completely received the hash value of the TBS data is ready. It is then used to verify the certificate's signature with the help of the certificate of the issuer I which is stored on the device as the trust anchor (Step 4).

The public operation of the code-based scheme is potentially composed of computations before the matrix-vector product is needed (Step 1b). These computations can be done before the public matrix transmission begins, e.g. they could be carried out before and/or during the receival of the TBS data preceding the public key. Once the public key matrix has been fully received and processed (i.e. after Step 2a), the remaining computations of the public operation are carried out (Step 3b), e.g. the addition of the error vector e in the McEliece scheme. The result is either a ciphertext (in case of an encryption scheme) or a Boolean value (in case of a signature verification). But whether this result is output respectively further processed by the device (Step 5a) depends on the result of the signature verification (Step 4). If the verification fails, the device will output an error answer (Step 5b).

3.2 Transmission Rates

In this section, we give an overview of transmission rates available for embedded systems, especially smart card microcontrollers.

For instance a SLE66CLX360PE [17] smart card platform from Infineon Technologies AG features an ISO/IEC 14443 compliant contactless interface which can transmit up to 106 KB/s. This allows the transmission of a McEliece public key of size 100 KB for the parameters given in Section 2.2 in about 1s, which can be considered at least acceptable for certain applications.

In the future, contactless transmission rates may be about 837,500 bytes/s [30], i.e. about 8 times higher than the rate considered above[1]. In the following

[1] In the referenced work, this transmission rate is actually only achieved in the direction from the card to the reader. However, we want to use it merely as an orientation for the transmission rates achievable in the near future.

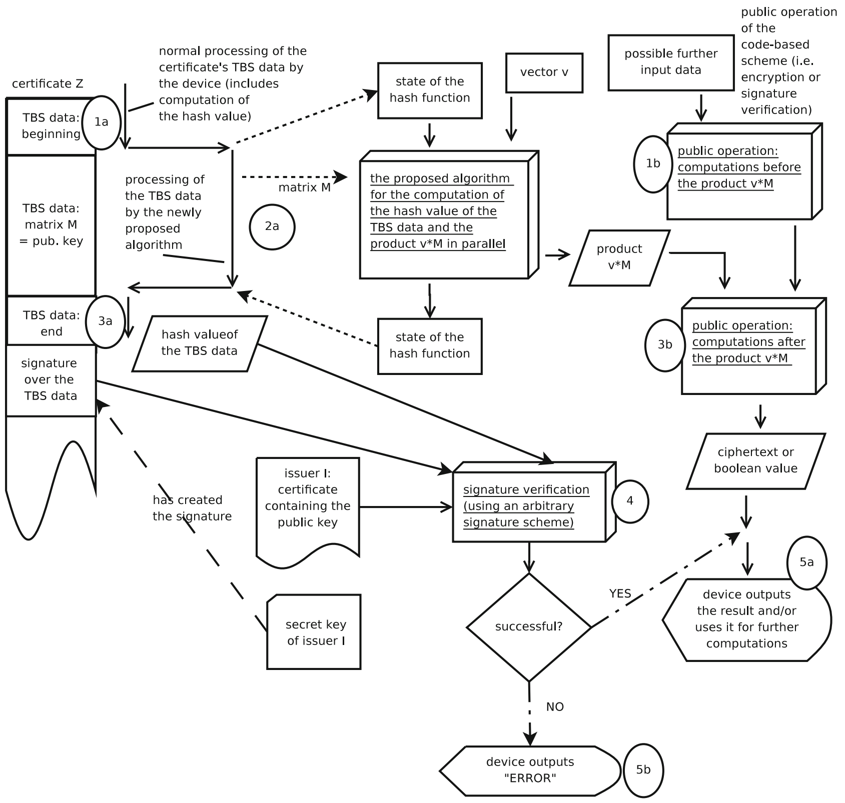

certificate Z
TBS data: beginning
1a
normal processing of the certificate's TBS data by the device (includes computation of the hash value)
TBS data: matrix M = pub. key
processing of the TBS data by the newly proposed algorithm
2a
matrix M
state of the hash function
vector v
the proposed algorithm for the computation of the hash value of the TBS data and the product v*M in parallel
product v*M
possible further input data
public operation of the code-based scheme (i.e. encryption or signature verification)
1b
public operation: computations before the product v*M
TBS data: end
3a
hash valueof the TBS data
state of the hash function
3b
public operation: computations after the product v*M
signature over the TBS data
ciphertext or boolean value
has created the signature
issuer I: certificate containing the public key
signature verification (using an arbitrary signature scheme)
4
5a
YES
device outputs the result and/or uses it for further computations
secret key of issuer I
successful?
NO
device outputs "ERROR"
5b

Fig. 1. Overview of the complete process of the on-line public operation

section, we will show that it is still feasible to sustain such a high transmission rate at typical smart card CPU speeds of about 30 MHz if adequate hardware support is available on the device. Note that in this case there are still about 35 CPU cycles available between the receival of two bytes.

3.3 Example Implementation

We implemented the proposed approach in the C programming language on an ATUC3A1512 32-bit microcontroller from Atmel's AVR32 family. We chose an embedded 32-bit platform basically because SHA-256 is designed for 32-bit platforms. There also exist 32-bit smart card controllers [31], thus our evaluations are significant for this type of platform.

The personal computer (PC) communicates with the AVR32 over a serial line. For the implementation of the serial communication, on the AVR32, we used the API for the device's Universal Asynchronous Receiver Transmitter (UART) provided by Atmel. On the side of the personal computer, we used the API to the serial port of the Linux operating system. The PC can send commands to the AVR32, which are formed by a six byte header and optional payload data, the length of which is encoded the last four header bytes. The first header byte is zero for all commands, and the second byte determines one of the following commands:

– set the vector to multiply
– carry out the on-line multiplication (starts an interactive protocol for the matrix transmission described below)
– get the multiplication result from the AVR32
– get the hash result from the AVR32

The AVR32 responds to these commands by sending a two byte status code and optional data payload preceding the status code, or in the case of the on-line multiplication command, by starting an interactive protocol.

This protocol is depicted in Figure 2. As a precondition, the vector to multiply has to be set in the device through the corresponding command. After the receival of the on-line multiplication command (which does not carry payload data), the AVR32 sets up two buffers $B1$ and $B2$ which are of an equal predefined size. It sends a two byte acknowledgement (ACK) code to the PC as the answer to the command. Then the PC sends the first matrix part which is of equal size as the buffers $B1$ and $B2$. The receival of a single byte over the UART interface of the AVR32 triggers an interrupt which is serviced by an Interrupt Service Routine (ISR) which writes the byte to the next free position in $B1$. After the first block has been received completely, the AVR32 sends another ACK code to the PC, who in turn reacts by sending the next part. At this point the AVR32 exchanges the role of the buffers $B1$ and $B2$: the data is now received to $B2$ (which did not play any role while receiving the first part), and $B1$, containing the first matrix part, is fed into the SHA-256 computation and the matrix multiplication. Both, the hashing and matrix multiplication are implemented as objects which can be updated by calling routines that take arbitrary amounts of data as an argument.

For hash functions, this is the standard implementation technique. Because demanded by our approach, we adopted this technique for the matrix multiplication. In our implementation, the matrix-vector multiplication is carried out column-wise. The advantages and disadvantages of this approach in contrast to row-wise multiplication is discussed in Section 3.4. The multiplication object knows the number of rows and columns of the matrix and has the source vector set. As the matrix data is fed column-wise it keeps track of the current row and column position. It processes the current column by carrying out the logical AND (multiplication in $\mathbb{F}_2$) between the matrix column and the vector 32-bit word-wise, and computes the XOR (addition in $\mathbb{F}_2$) with a 32-bit accumulator. When a column is finished, the parity (i.e. sum of all the word's bits in $\mathbb{F}_2$) of the accumulator is written to the corresponding result bit.

Non-interactive Version of the Protocol. It turned out that the interactive protocol incurs significant delay in the communication which most probably results from the fact that our PC program is running in user space and thus sending and receiving data via the serial interface is delayed. If the protocol were implemented in a card terminal, which could be the case in a real world implementation of the on-line multiplication such issues would not arise. To show the efficiency of the approach, we modified the protocol depicted in Figure 2: the AVR32 does not send any ACK answers beyond the very first one. Consequently, the matrix data is sent as a continuous stream after the AVR32 has sent the initial ACK. In this way, the protocol looses the feature that it works independently of the ratio of transmission speed and computation speed: in this non-interactive setting, it must be guaranteed that the hash and multiplication computation of the processed buffer has finished before the receive buffer has been completely filled. With this approach the performance could be improved by a factor of roughly 1.3 compared to the interactive variant of the protocol. The concrete results are discussed shortly.

Simulation of Higher Transmission Rates. On the chosen AVR32 platform, the maximal transmission speed is given by a baud rate of 460,800. In the RS232 transmission format each data byte is encoded in 10 bits, yielding a net transmission rate of 46,080 byte/s. In order to demonstrate the computation speed that would be possible beyond this limitation, we implemented a means of simulating higher transmission speeds. This is achieved by creating a matrix whose rows have repetitive entries, i.e. the values of 8-bit chunks repeats r times. An example of the beginning of a row for $r = 4$ would be

```
0x1D, 0x1D, 0x1D, 0x1D, 0xA3, 0xA3, 0xA3, 0xA3, 0x22, ...
```

In this setting, on the PC side such a repetitive matrix is generated. When the matrix is transmitted, however, each repeated element is sent only once. On the receiving side, the repetition value r is also known and each received byte is appended to the buffer r times. In this way, we simulate a transmission rate $B_{\text{sim}} = rB_{\text{real}}$, where B_{real} is the actual UART transmission rate.

Table 1 shows the measurement results for the non-interactive version described in the previous paragraph. Here, we used a matrix with 1000 rows and 800 columns, i.e. yielding a size of 100,000 bytes. This is approximately the size of McEliece public keys with 100 bit security [19]. In all our measurements the CPU speed of the AVR32 was set to 33 MHz, since also todays contactless smart card platforms run at approximately this speed, for instance the Infineon Technologies SLE76 [16] smart card controller. The SLE76 CPU only runs at 30MHz, using this in our implementation showed that at this CPU speed the (simulated) transmission rate given in the rightmost column of Table 1 could not be supported in the experiment.

Furthermore, we measured the random error vector creation as the second part of the encryption operation for parameters $n = 2048$ and $t = 50$ to be less than 4 ms at a CPU speed of 33MHz, the addition (XOR) of the error vector to the intermediate vector is certainly even much less complex and thus completely negligible for the timings considered here.

The transmission speed of 386,640 bytes/s, that can be sustained in our test setup, is approximately half of that of the research implementation presented in [30] already mentioned in Section 3.2. Thus our results show that even without dedicated hardware, todays embedded platforms already enable computation speeds for the hash computation and matrix multiplication not too far from the associated transmission rates that can be expected to be supported by contactless devices in the near future. This makes it feasible that with adequate hardware support the full 837,500 Byte/s rate given in [30] can be supported by the throughput of the computational operations.

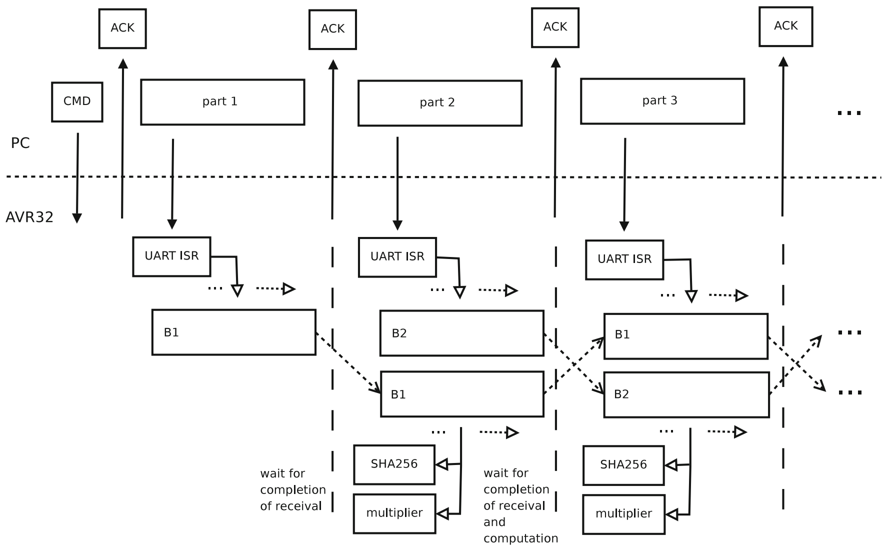


Fig. 2. Schematic overview of the interrupt based implementation of the on-line multiplication

The hash implementation is based on the open source implementation [32]. The C source code allows for complete unrolling of the SHA-256 compression function through a macro definition. Activating loop unrolling resulted in a performance gain of 1.6 for the hash function computation. All further performance data is based on this implementation choice.

3.4 Column-wise vs. Row-wise Matrix-Vector Multiplication

The row-wise computation of the matrix-vector multiplication is an alternative to the column-wise approach. In this case the computation of the result is according to $b = \sum_i M_i a_i$, where M_i is the vector represented by the i-th row of M and a_i it the i-th entry of the vector a. This means that a row M_i is added to the result if the corresponding bit a_i is one, otherwise nothing has to be done. In the normal case, where the whole matrix is available instantly, this approach has a significant

Table 1. Performance of the SHA-256 and binary matrix multiplication on the AT32UC3A1 platform. The results in the first column are based on the throughput benchmarking results for the two computational tasks. The following two entries in this column, that give the resulting time for the on-line multiplication and the transmission rate necessary to support the throughput of both computational tasks, are theoretically derived from the former. In the second column, the time of the whole on-line matrix multiplication with the given transmission rate $B_{\text{sim}} = 8 \cdot 46,080$ byte/s was measured on the ATUC3A1512 platform and the computational throughput given in the first row is the effective throughput corresponding to the measured running time. Here a receive buffer size of 1536 bytes was used.

	based on computation throughput	**experimental result - w/o ACK**
cycles/byte	measured: 55.6 for SHA-256, 4.2 for mult. yields: **59.8**	92
time at 33MHz CPU for 100,000 Bytes	181ms	**279ms**
transmission rate in bytes/s	551,839	$B_{\text{sim}} = 368,640$ ($r = 8$)

advantage over the column-wise approach since on average half of vector a's bits have value zero. But in the case of the on-line public operation, this advantage disappears since the matrix-vector multiplication's running time is determined by the transmission time alone (under the assumption of sufficient computational power of the device as analyzed in Section 3.2). The row-wise approach would only have an advantage if the saved computational effort could be used to perform other tasks, which can be assumed to be rather unlikely or at least of minor relevance in the context of embedded devices such as smart cards.

On the other hand, the disadvantage of the row-wise multiplication lies in its potential side-channel vulnerability. Specifically, if an attacker is able to find out whether the currently transmitted row is added or ignored, for instance by analyzing the power trace [33], he can deduce the value of the secret bit a_i. Of course, countermeasures can be implemented. A certain randomization could for instance be introduced by keeping a number of received rows in a buffer and processing them in a randomized order. However, whether the questionable computational advantage of this method is worth such efforts must be decided in a concrete implementation scenario.

In any case, once the X.509 key format for a code-based scheme is defined, the choice for one of the two methods is taken. While it then would still be possible to transmit the matrix in the other orientation in order to carry out the multiplication, the on-line hash computation only works if the correct orientation is used.

3.5 Code-Based Signature Schemes

A number of code-based signature schemes have been proposed. In the following, we will address two of these schemes very briefly with the goal of showing that the proposed approach for the on-line public operation is applicable to both of them.

In [3], the McEliece scheme is inverted in the sense that the signer proves his ability to decode a binary vector related to the message using a certain code. Thus, the signature verification basically consists of a matrix-vector multiplication just like for the encryption schemes described in Section 2.2. For security considerations concerning this scheme please refer to [34,35].

A signature scheme involving two binary matrices as the public key is presented in [4]. In the verification operation, both matrices have to be multiplied by a vector. Thus the on-line public operation can be carried out by transmitting them one after another. Note, however, that the originally proposed parameters for this scheme are insecure [36].

4 McEliece Decryption without the Parity Check Matrix

In the McEliece scheme, the first step of the decryption operation, which for the sake of brevity we will not not fully explain here, is, as already mentioned in Section 2.2, the computation of the syndrome polynomial $S(Y)$ as $S(Y) \equiv \sum_{i=1}^{n} \frac{c_i}{Y \oplus \alpha_i} \mod g(Y)$, where $g(Y)$ is the Goppa Polynomial, c_i is the i-th ciphertext bit and the α_i is the i-th support element.

In the implementations [21,18,19,22], the syndrome computation is done with the help of the parity check matrix H of the code. This matrix is in fact nothing else than a list of all the n different polynomials $\frac{1}{Y \oplus \alpha_i} \mod g(Y)$ which yields the syndrome vector when multiplied with the ciphertext as a bit vector.

The syndrome computation without the parity check matrix is in principle achieved by invoking an Extended Euclidean Algorithm (EEA) with $g(Y)$ and $Y \oplus \alpha_i$ as the initial remainders. This EEA executes in a single iteration. Accordingly, in an implementation of the syndrome computation a number of optimizations are possible. The resulting Algorithm is given in Algorithm 3. There, $z[i]$ denotes the i-th ciphertext bit and B_j the coefficient to Y^j of $B(Y)$, etc. Its average complexity (i.e. for a ciphertext with Hamming weight $n/2$), expressed in the terms of additions, multiplication and inversions in $\mathbb{F}_{2^m}$, is $C_{\text{syndr}} = nt(C_{\text{mult}} + C_{\text{add}}) + \frac{n}{2} C_{\text{inv}}$.

We implemented this algorithm in a McEliece PKC implementation based on the open source implementation [21] presented in [7]. Table 2 shows the timing results measured on an Atmel AT32 AP7000 CPU, a CPU similar to the AT32UC3A1. The CPU runs at 150 MHz, however, we give the according running time for the typical smart card CPU speed of 33 MHz, which was already employed in Section 3.3. The smaller parameter set has already been introduced in Section 2.2, the larger one is is proposed in [27] for 128-bit security though there the authors assume addition of $t+1$ errors, which is possible through the employment of list-decoding [37], which is not supported by our implementation. Accordingly the security level here is only approximate, but the reduction of security of our implementation only using t errors, however, can easily be bounded by understanding that an attacker can get from a ciphertext with $t+1$ errors to t errors by guessing one error position correctly, the success probability of which is $(t+1)/n = 0.02$. Accordingly, the security of the scheme with t errors cannot be smaller than $128 - \log_2(1/0.02) > 122$ bits.

Algorithm 3. The Syndrome computation without parity check matrix

```
Require: the ciphertext $\boldsymbol{z} \in \mathbb{F}_2^n$, and the Goppa Polynomial $g(Y) \in \mathbb{F}2^m[Y]$ of degree $t$
Ensure: the syndrome polynomial $S(Y) \in \mathbb{F}2^m[Y]$ of degree $\leqslant t-1$
  $S(Y) \leftarrow 0$
  for $i \leftarrow 0$ up to $n-1$ do
    if $\boldsymbol{z}[i] = 1$ then
      $B(Y) \leftarrow 0$
      $b \leftarrow g_t$
      for $j \leftarrow t-1$ down to 0 do
        $B_j \leftarrow b$
        $b \leftarrow b \cdot \alpha_i \oplus g_j$
      end for
      $f \leftarrow b^{-1}$
      for $j \leftarrow 0$ up to $\deg(B(Y))$ do
        $S_j \leftarrow S_j \oplus f \cdot B_j$
      end for
    end if
  end for
```

Table 2. Private key sizes, cycle counts and corresponding timings taken for the McEliece decryption operation and its suboperation, the syndrome computation, on an Atmel AT32 AP7000 CPU. Each cycle count was obtained by carrying out the operation ten times and taking the mean of the results.

code parameters		$n = 2048$, $t = 50$		$n = 2960$, $t = 56$	
security level		100 bit		> 122 bit	
		cycles	t @ 33 MHz	cycles	t @ 33 MHz
with par. ch. mat.	cyc. whole decr.	$2.00 \cdot 10^6$	61 ms	$3.12 \cdot 10^6$	95 ms
	cyc. only syndr. comp.	$0.26 \cdot 10^6$	8 ms	$0.39 \cdot 10^6$	12 ms
	private key bytes	158,140		277,328	
w/o par. ch. mat.	cyc. whole decr.	$4.42 \cdot 10^6$	134 ms	$7.39 \cdot 10^6$	224 ms
	cyc. only synd. comp.	$2.65 \cdot 10^6$	80 ms	$4,71 \cdot 10^6$	143 ms
	private key bytes	17,340		28,688	

The respective private key sizes without the parity check matrix given in Table 2 are formed by the Goppa Polynomial $g(Y)$, the support Γ, a matrix for computing the square root modulo $g(Y)$, which is needed to speed up the decryption, and the logarithm and anti-logarithm tables for $\mathbb{F}_{2^m}$, each of size $\lceil \log_2 n \rceil$ elements, i.e. a total of 8192 resp. 16384 bytes for either parameter set (each element occupies 2 bytes). These tables need not necessarily be stored in the key, instead they can be created in RAM before the decryption operation, if allowed by the memory constraints of the given platform.

From this example implementation, that does not use any hardware support for the $\mathbb{F}_{2^m}$ operations or DSP instructions, we see that the decryption time approximately doubles when the parity check matrix is not stored as part of the key, but due to the general speed advantage of the McEliece scheme over RSA or Elliptic curve based schemes [7,18] these timings are still highly competitive.

5 Conclusion

In this work, on the one hand, we have shown an approach for implementing the operations involving code-based public keys on memory-constrained devices like smart cards, that covers the matrix-vector multiplication as well as the hash computation for the verification of the user certificate. The solution is applicable to basically all code-based encryption and signature schemes that have been proposed so far. Thus we are confident that this work improves on the applicability of this class of cryptographic schemes by reducing the impact of the large public key sizes for memory-constrained devices.

Furthermore, we also showed that the McEliece private key size can be dramatically reduced by excluding the parity check matrix while still allowing for practical decryption timings on memory constrained devices. As a result, the McEliece scheme in our opinion gains superiority over the Niederreiter scheme: while the situation for the public key as discussed in this work is equal for both schemes, only the McEliece scheme allows for the reduction of the private key size as proposed in this work. The reason is simply that the Niederreiter private key size is mainly determined by the size of the scrambler matrix T, which cannot be excluded in this scheme. In [20], the matrix T is implemented as a pseudorandom sequence of bits, which effectively reduces the Niederreiter private size, but this comes at the expense of public key size: because of the pseudorandom nature of T, the public key matrix cannot be in reduced row echelon form, resulting in an otherwise unnecessary increase of the public key size. In view of the results of the first part of this work, this directly affects the time taken by the encryption operation on an embedded device (at least in a PKI context).

References

1. McEliece, R.J.: A public key cryptosystem based on algebraic coding theory. DSN Progress Report 42-44, 114–116 (1978)
2. Niederreiter, H.: Knapsack-type cryptosystems and algebraic coding theory. Problems Control Inform. Theory 15(2), 159–166 (1986)
3. Courtois, N.T., Finiasz, M., Sendrier, N.: How to Achieve a McEliece-Based Digital Signature Scheme. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 157–174. Springer, Heidelberg (2001)
4. Kabatianskii, G., Krouk, E., Smeets, B.: A Digital Signature Scheme Based on Random Error-Correcting Codes. In: Darnell, M. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 161–167. Springer, Heidelberg (1997)
5. Stern, J.: A New Identification Scheme Based on Syndrome Decoding. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 13–21. Springer, Heidelberg (1994)
6. Bernstein, D.J., Buchmann, J., Dahmen, E.: Post Quantum Cryptography. Springer Publishing Company, Incorporated (2008)

7. Biswas, B., Sendrier, N.: McEliece Cryptosystem Implementation: Theory and Practice. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 47–62. Springer, Heidelberg (2008)
8. Berger, T.P., Cayrel, P.-L., Gaborit, P., Otmani, A.: Reducing Key Length of the McEliece Cryptosystem. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 77–97. Springer, Heidelberg (2009)
9. Berger, T.P., Loidreau, P.: How to Mask the Structure of Codes for a Cryptographic Use. Designs, Codes and Cryptography 35, 63–79 (2005), doi:10.1007/s10623-003-6151-2
10. Misoczki, R., Barreto, P.S.L.M.: Compact McEliece Keys from Goppa Codes. In: Jacobson Jr., M.J., Rijmen, V., Safavi-Naini, R. (eds.) SAC 2009. LNCS, vol. 5867, pp. 376–392. Springer, Heidelberg (2009)
11. Otmani, A., Tillich, J.P., Dallot, L.: Cryptanalysis of Two McEliece Cryptosystems Based on Quasi-Cyclic Codes. Mathematics in Computer Science 3, 129–140 (2010)
12. Faugère, J.-C., Otmani, A., Perret, L., Tillich, J.-P.: Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 279–298. Springer, Heidelberg (2010)
13. Umana, V.G., Leander, G.: Practical key recovery attacks on two McEliece variants. In: Cid, C., Faugere, J.-C. (eds.) SCC 2010, pp. 27–44 (2010)
14. Goppa, V.D.: A new class of linear correcting codes. Problems of Information Transmission 6, 207–212 (1970)
15. Faugère, J.C., Otmani, A., Perret, L., Tillich, J.P.: A distinguisher for high rate McEliece cryptosystems. In: 2011 IEEE Information Theory Workshop (ITW), pp. 282–286. IEEE (2011)
16. Infineon Technologies AG: SLE76 Product Data Sheet, `http://www.infineon.com/cms/de/product/channel.html?channel=db3a3043156fd57301161520ab8b1c4c`
17. Infineon Technologies AG: SLE 66CLX360PE(M) Family Data Sheet, `http://www.infineon.com/dgdl/SPI_SLE66CLX360PE_1106.pdf?folderId=db3a304412b407950112b408e8c90004&fileId=db3a304412b407950112b4099d6c030a&location=Search.SPI_SLE66CLX360PE_1106.pdf`
18. Eisenbarth, T., Güneysu, T., Heyse, S., Paar, C.: MicroEliece: McEliece for Embedded Devices. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 49–64. Springer, Heidelberg (2009)
19. Strenzke, F.: A Smart Card Implementation of the McEliece PKC. In: Samarati, P., Tunstall, M., Posegga, J., Markantonakis, K., Sauveron, D. (eds.) WISTP 2010. LNCS, vol. 6033, pp. 47–59. Springer, Heidelberg (2010)
20. Heyse, S.: Low-Reiter: Niederreiter Encryption Scheme for Embedded Microcontrollers. In: Sendrier, N. (ed.) PQCrypto 2010. LNCS, vol. 6061, pp. 165–181. Springer, Heidelberg (2010)
21. Biswas, B., Sendrier, N.: HyMES - an open source implementation of the McEliece cryptosystem (2008), `http://www-rocq.inria.fr/secret/CBCrypto/index.php?pg=hymes`
22. Shoufan, A., Wink, T., Molter, G., Huss, S., Strenzke, F.: A Novel Processor Architecture for McEliece Cryptosystem and FPGA Platforms. In: ASAP 2009, pp. 98–105. IEEE Computer Society, Washington, DC (2009)
23. German Federal Bureau of Information Security (BSI): Technical Guideline TR-03110: Advanced Security Mechanisms for Machine Readable Travel Documents, Version 2.02 (2009)
24. German Federal Bureau of Information Security (BSI): Technical Guideline TR-03111: Elliptic Curve Cryptography, Version 1.11 (2009)

25. MacWilliams, F.J., Sloane, N.J.A.: The theory of error correcting codes. North Holland (1997)
26. Kobara, K., Imai, H.: Semantically Secure McEliece Public-Key Cryptosystems - Conversions for McEliece PKC. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 19–35. Springer, Heidelberg (2001)
27. Bernstein, D.J., Lange, T., Peters, C.: Attacking and Defending the McEliece Cryptosystem. In: Buchmann, J., Ding, J. (eds.) PQCrypto 2008. LNCS, vol. 5299, pp. 31–46. Springer, Heidelberg (2008)
28. Cooper, et al.: RFC 5280, http://tools.ietf.org/html/rfc5280
29. Coronado, L.C., Buchmann, J., Carlos, L., Garcia, C., Dahmen, E., Klintsevich, E., Darmstadt, T.U.: CMSS – An Improved Merkle Signature Scheme Johannes Buchmann (2006), http://www.cdc.informatik.tu-darmstadt.de/~dahmen/papers/BCDDK06.pdf
30. Witschnig, H., Patauner, C., Maier, A., Leitgeb, E., Rinner, D.: High speed RFID lab-scaled prototype at the frequency of 13.56 MHz. E & I Elektrotechnik und Informationstechnik 124, 376–383 (2007), doi:10.1007/s00502-007-0485-9
31. Infineon Technologies AG: SLE78 Product Data Sheet, http://www.infineon.com/cms/en/product/channel.html?channel=db3a30431ce5fb52011d47b166342af0
32. Gay, O., http://www.ouah.org/ogay/sha2/
33. Kocher, P.C., Jaffe, J., Jun, B.: Differential Power Analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
34. Overbeck, R., Sendrier, N.: Code-based cryptography. In: Bernstein, D., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 95–145. Springer (2009)
35. Finiasz, M.: Parallel-CFS: Strengthening the CFS McEliece-Based Signature Scheme. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 159–170. Springer, Heidelberg (2011)
36. Otmani, A., Tillich, J.-P.: An Efficient Attack on All Concrete KKS Proposals. In: Yang, B.-Y. (ed.) PQCrypto 2011. LNCS, vol. 7071, pp. 98–116. Springer, Heidelberg (2011)
37. Bernstein, D.J.: List Decoding for Binary Goppa Codes. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) IWCC 2011. LNCS, vol. 6639, pp. 62–80. Springer, Heidelberg (2011)

100% Connectivity for Location Aware Code Based KPD in Clustered WSN: Merging Blocks

Samiran Bag[1], Aritra Dhar[2], and Pinaki Sarkar[3]

[1] Applied Statistics Unit, Indian Statistical Institute, Kolkata-700108, India
samiran_r@isical.ac.in
[2] Guru Nanak Institute of Technology, Kolkata-700114, India
[3] Mathematics Department, Jadavpur University, Kolkata-700032, India
{aritra.dhar7,pinakisark}@gmail.com

Abstract. Key management in wireless sensor networks (WSN) is a challenging task because of the stringent resource available to the nodes. As such key predistribution (KPD) is regarded as one of the best option for key management in WSN. This work analyzes an existing work by Simonova et al. that makes use of deployment knowledge where the deployment zone consists of clusters of nodes. Transversal design (TD) based KPD scheme of Lee and Stinson is used to distribute the keys in these clusters. However Simonova et al. points out that any KPD could have been used. This leads the current authors to investigates the applicability of Ruj and Roy's Reed Solomon (RS) code based KPD , similar to the TD based KPD in distributing the key in each cluster. Much like the TD based KPD, the RS code based KPD does not offer full connectivity among nodes in the clusters amounting to lack of full connectivity in the amalgamated network. Full connectivity among nodes in each cluster and thus the entire network can be achieved by a deterministically merging strategy using exactly two nodes. This merging strategy is certainly better than a random approach by Chakrabarti et al. where the exact number of nodes being merged is not specified and does not ensure full connectivity. Since the scheme of Simonova et al. uses too many keys after amalgamation, a modified approach using Cluster Head (CH) is proposed to provide full communication in the network. Comparative study establishes the proposed Cluster Head design perform better than the KPD of Simonova et al. while proving the efficiency of the merging strategy.

Keywords: Security in WSN, Key predistribution (KPD), Reed Solomon (RS) Code, Combinatorial Designs, Deterministic Merging Blocks, Connectivity.

1 Introduction

Wireless sensor networks (WSN) is a two tier network consisting of many tiny sensing devices at the lower tier and a Key distribution server(KDS) at the upper tier. However depending on the application WSNs can be made to have more tiers. which are normally used to provide a sort of *hierarchy* to a network by subdividing it into (small) clusters. These sensor networks gather information about the environment. Sensor nodes have their inbuilt memory module, a small processor system and a very limited power source that allows the sensors to process data and wireless transceiver to communicate to other sensors of the network as well as to the KDS.

A heterogeneous wireless sensor network can have different types sensor nodes differing in resourcefulness. Such a network can have ordinary nodes having less resource in terms of low amount of memory, less computational power and very limited battery power and some nodes called cluster heads which are rich in resources. As the name suggests, communication in wireless sensor networks is achieved using radio frequencies. Resource constrained nodes can communicate with each other only within a limited range having center as the node and small radius termed as *Radio Frequency range* or *radius of communication* or *physical layer of* [10]. This range or radius is generally same for ordinary sensors and may be varied for CHs. On the other hand, the KDS has quite a large *radius of communication*.

In spite of all the weaknesses in the basic building blocks of WSNs, these networks have several military applications like monitoring enemy movements, etc. Besides they are utilized for other scientific purposes like smoke detection, wild fire detection, seismic activity monitoring etc.

Towards secure communication, it is important that any two sensor nodes should communicate in an encrypted manner using a common secret key. Due to resource constraints symmetric encryption is preferred over public key techniques. Symmetric key cryptographic techniques requires the nodes to possess the secret key beforehand. Here also standard techniques like online key exchange using public parameters or the concept of using trusted authorities are generally avoided. Alternatively key predistribution (KPD) involving preloading and later establishing the keys among the nodes is an effective solution.

Combinatorial designs have been being used for KPD in WSN for almost a decade. Çamptepe & Yener [1] first proposed the use of such designs for predistribution of keys in WSN which inspired other researchers to follow suit. Some research work on the use of design theory in KPD in WSN can be found in [2,5,12,10].

In some applications sensor networks are deployed in a grid group fashion where the entire deployment zone is split into smaller zones and sensor nodes are deployed in this smaller zones such that all the sensors falling into one smaller region forms a group among themselves. Sometimes one or more CHs are deployed into each of these smaller zones. Such schemes can be found in [5,12,14,15,20].

1.1 Related Works

In most of their applications, WSNs once deployed works unattended for long duration of time while its constituent nodes deals with lot of sensitive information. During the message exchange, symmetic key cryptosystems are preferred to public key cryptography as the later incurs lots of computations. In symmetric key cryptography, both the communicating parties possess the same key prior to message exchange. Standard online key exchange technique involving public parameters are generally avoided due to their heavy and costly computations. Treating a node as Trusted Authority is highly risky as capture of that node will make the entire system vulnerable. Hence schemes like Kerberos [16] can not be implemented for security in WSN. This leads one to adopt various *Key Predistribution (KPD)* techniques.

Eschenauer & Gligor in their work [8] suggested the pioneering idea of *predistribution of keys* into the sensors. Their idea can broadly be divided into two steps:

– Keys are (randomly) preloaded into the sensors prior to deployment.
– *Key establishment*: this phase consists of
  • *Shared key discovery:* establishing shared common key among the nodes; and
  • *Path key establishment*: establishing path via other node(s) between a given pair of nodes that do not share any common key.

Random preloading of keys means that the *key rings* or *key chains* are formed randomly. *Key establishment* is done in [8] using *challenge and response* technique. Schemes that follow similar random preloading and later probabilistically establishing strategy are called *random key predistribution schemes*. Some more examples of such schemes are [4,6,12]. Çamptepe & Yener presents an excellent survey of such schemes in their technical report [2].

On the other hand there exists KPD schemes based on deterministic approach involving *Mathematical* tools. Çamptepe & Yener [1] were first to propose a deterministic KPD scheme where keys are preloaded and later established in an deterministic manner. Following their initial work, numerous deterministic KPD schemes based on combinatorial designs have been proposed as can be found in [9,10,11,13].

Simonova et al. in their paper [15], discussed two KPD schemes(homogeneous and heterogeneous) using deployment knowledge. In both the schemes the entire deployment zone is considered as the integration of some grids. There are two types of key pools original key pool($OKP_{i,j}$ for a grid $h_{i,j}$) and deployment key pool(DKP). For the OKP, they consider the transversal designed by Lee and Stinson [11]. It is to be noted that all the OKPs are disjoint.To construct the DKP, they considered augmented square grids made of m^2 number of normal grids. The DKP is defined as $DKP_{i,j} = \{\bigcup_{x,y} OKP_{x,y} | x = 1,2,\ldots,(i+m), y = 1,2,\ldots,(j+m)\}$. So the grid is augmented by $m-1$ cells on both horizontal and vertical sides before deployment.

1.2 Contributions in This Paper

This work suggests an improvement the research work by Simonova et al. [15]. The generic nature of [15] allows a KPD using Reed Solomon code [13] to form the basis of the clusters of [15]. However both the original scheme in [15] and this new scheme lacks full connectivity among nodes and selective node attack is actually feasible. As for lack of common key, when two nodes, not sharing a key wants to communicate, a common path has to be sought. This leads to increased cost of communication and trusting other (intermediate) nodes. Thus one is prompted to think of a remedial strategy to combat this problem.

The problem can be addressed applying a deterministic strategy where exactly two nodes are merged to form fully connected blocks. Corollary 1 of Theorem 2 of section 3 establishes that merging two nodes of Ruj & Roy's scheme [13] in a certain fashion results in full connectivity among the newly formed (merged) blocks. Hence the equivalent Simonova et al. design based on RS code has full connectivity. Issue of selective node attack is dealt with by proposing a new hierarchical design in section 5 where cluster heads (CH) are used as connecting media between clusters.

In one odd work, merging block concept was proposed by Chakrabarti et al. where 'z' nodes of TD based KPD [10] were randomly merged to form blocks. The resultant network thus possessed '$\lfloor \mathscr{N}/z \rfloor$' many blocks where $\mathscr{N}$ is the number of nodes in the original TD model in [10]. Apart from failing to give theoretical justification to most aspects of their design, their model fails to provide full connectivity among the blocks of the merged network.

1.3 Terminologies

'Uncompromised nodes' mean nodes that are not captured by the adversary. 'Communication model/scheme' and 'Key predistribution (KPD) model/scheme' mean the same and abbreviated to 'KPD'. 'TD' is used as abbreviation of Transversal Design. Unless otherwise stated, the notation and symbols introduced here will be reserved for the terms mentioned here.

1.4 Notations

- Connectivity between nodes: Two nodes within communication radius of each other share common key, which is not compromised are said to be connected.
- Full–connectivity of the network: If any two nodes of the network share a secret key then the network is said to have full connectivity. Here the individual node's RF range is not taken into account as the deployment is random.
- TD based KPD: Transversal Design based key predistribution (KPD) of [10].
- RS code based KPD: Reed Solomon code based KPD of Ruj & Roy in [13].
- The set $\mathbb{N} = \mathbb{Z}_+ = \{1,2,3,\ldots\}$ is the set of all natural numbers or positive integers.
- $\mathscr{N}$: denotes total number of nodes for both the KPD in [13,10].
- p : prime, $q = p^r-$prime power, the number of elements in the finite field, $\mathbb{F}_{p^r}-$ finite field with $char(\mathbb{F}_q) = p, r \in \mathbb{N}$.
- k: number of keys in the key ring of each node as in Lee & Stinson work [10].
- l : the dimension of the Reed Solomon code denoted by Ruj & Roy in [13].

Note that in [13], Ruj & Roy reserved the symbol k to denote the dimension of the Reed Solomon code. In this paper, we used the symbol l to denote the same as in [10], for the sake of maintaining consistency of notation with the pioneering work. Notations related to combinatorial designs and resiliency are defined in sections 1.5 and 7 respectively.

1.5 Basics of Combinatorial Design

This section briefly describes some basic notion of *combinatorial design* necessary for understanding Ruj & Roy [13] scheme. Elaborate discussions can be found in any standard book on *Combinatorics* like Stinson's [18] or [19].

Consider a finite set $\mathscr{X}$. Then a *set system or design* is a pair $(\mathscr{X}, \mathscr{A})$ where $\mathscr{A}$ is a set of subsets of $\mathscr{X}$ called blocks. Elements of $\mathscr{X}$ are called varieties. A $(v,b,r,k)-1$ design based on $\mathscr{X}$ satisfies the following conditions:

- $|\mathscr{X}| = v$, $|\mathscr{A}| = b$.
- Each subset in $\mathscr{A}$ contains exactly k elements (*rank*).
- Each variety in $\mathscr{X}$ occurs in r many blocks (*degree*).

Further a $(v,b,r,k)-1$ design is called a *configurations* if any two of it's blocks intersect in at most one point.

Group-divisible design of type g^u and block size k: is a triple $(\mathscr{X},\mathscr{H},\mathscr{A})$, where

1. $\mathscr{X}$ is a finite set with $|\mathscr{X}| = gu$.
2. $\mathscr{H}$ is a partition of $\mathscr{X}$ into u parts, that is, $\mathscr{H} = \{H_1,H_2,H_3,\ldots,H_u\}$ with $\mathscr{X} = H_1 \cup H_2 \cup H_3 \cup \ldots \cup H_u$, $|\mathscr{H}_i| = g \ \forall\ 1 \leq i \leq u$ and $\mathscr{H}_i \cap \mathscr{H}_j = \phi \ \forall\ 1 \leq i \neq j \leq u$.
3. $\mathscr{A}$ is the collection of blocks of $\mathscr{X}$ having the following properties: $|H \cap A| \leq 1 \ \forall\ H \in \mathscr{H}$, $\forall\ A \in \mathscr{A}$, given any pair of varieties $x \in H_i, y \in H_j$ with $i \neq j\ \exists$ unique $A \in \mathscr{A}$ such that $x,y \in A$.

Transversal Designs TD(k,n): are special type of group-divisible designs with $g = n, u = k$, and as well as $(nk,n^2,n,k)-$ configuration. Relation between BIBDs, group-divisible designs and transversal designs can be found in books on Combinatorial Designs like [19,18] while [10, section III] briefs the topic nicely.

Common Intersection Design (CID): maximal CID – Suppose that $\mathscr{X},\mathscr{A}$ is a v,b,r,k–configuration. $\mathscr{X},\mathscr{A}$ is said to be a μ–common intersection design (μ–CID) if:

$$|\{A_\alpha \in \mathscr{A} : A_i \cap A_\alpha \neq \phi \text{ and } A_j \cap A_\alpha \neq \phi\}| \geq \mu$$

whenever $A_i \cap A_j = \phi$. For the sake of consistency, one defines $\mu = \infty$ whenever $A_i \cap A_j \neq \phi, \forall\ i,j$.

For any given set of parametric values of v,b,r,k, such that a configuration can be obtained with them, one would like to construct a configuration with maximum possible μ. This *maximal value of* μ will be denoted μ^*. *Theorem 14.* of [10, section IV] basically establishes such $TD(k,n)$ designs are $k(k-1)^* - CID$. To ensure the existence of a 3^{rd} node sharing separate common keys with both the given non–communicating nodes, it is important that the value of is μ^* reasonably high. When $k = q$ and $q-1$, one can readily see that $T(q,q)$ and $TD(q-1,q)$ possesses pretty high value of μ^*.

2 KPD Using Reed Solomon Codes

This section is devoted to the description of key predistribution scheme proposed by Ruj and Roy in [13]. The scheme uses Reed Solomon codes to predistribute and establish the communication keys among the sensor nodes. The construction of Reed Solomon codes has been given in [13]. For the sake of completeness, salient features are being sketched here:

To construct (n,q^l,d,q) Reed Solomon code having alphabet in the finite field $\mathbb{F}_q$ (q: prime or prime power > 2), consider the following set of polynomials over $\mathbb{F}_q$:

$$\mathscr{P} = \{g(y) : g(y) \in \mathbb{F}_q[y], deg(g(y)) \leq l-1\} \tag{1}$$

Thus the number of elements in $\mathscr{P}$ denoted by $|\mathscr{P}| = q^l$. Let $\mathbb{F}_q^* = \{\alpha_1,\alpha_2,\alpha_3,\ldots,\alpha_{q-1}\}$ be the set of non-zero elements of $\mathbb{F}_q$. For each polynomial $p_m(y) \in \mathscr{P}$, define

$$cp_m = (p_m(\alpha_1), p_m(\alpha_2), \ldots, p_m(\alpha_{q-1}))$$

to be the m^{th} codeword of length $n = q-1$. Let $C = \{cp_m : p_m(y) \in \mathscr{P}\}$ be the collection of all such code words formed out of the polynomials over $\mathbb{F}_q$. This results in a

(n, q^l, d, q) Reed Solomon Code. Since the number of code-words is q^l, the system can support up to q^l nodes.

Now the polynomial p_m and the corresponding codeword cp_m are given to the m^{th} node. For the codeword $cp_m = (a_1, a_2, \ldots, a_n)$, one assigns the keys having key-identifiers $(a_1, \alpha_1), (a_2, \alpha_2), \ldots, (a_n, \alpha_n)$ where $a_j = p_m(\alpha_j), j = 1, 2, \ldots, n$ to the m^{th} node. The node id of the m^{th} node is obtained by evaluating the polynomial p_m at $x = q$ and taking only the numerical value. That is the m^{th} node has the node id $p_m(q)$ (without going modulo 'p'). Alternatively one could have considered taking tuples formed from each node's polynomial's coefficients as its node id. Such considerations of taking tuples as node ids can be found in [10].

A WSN with 16 nodes based on Reed Solomon parameters $q = 4, n = 3$ & $l = 2$ is presented in Table 1. Here '2' means the polynomial 'x' and '3' means the polynomial '$x+1$' modulo the irreducible polynomial $x^2 + x + 1$ over $\mathbb{F}_2[x]$ which are commonly referred to as $\overline{x}$ and $\overline{x+1}$. Thus $0, 1, 2, 3$ forms the finite field $\mathbb{F}_4$. The nodes' polynomials $i + jy \in \mathbb{F}_4[y]$ for $0 \le i, j \le 3$ are given in 2nd row of Table 1. By evaluating these polynomials at non-zero points, the keys $(p_m(b), b)$ where $0 \le i, j \le 3$ have been derived and tabulated in the corresponding columns.

Table 1 constructed by similar computations is being presented in a slightly different manner from Ruj & Roy [13]. This *group–divisible design* form of presentation helps one realize the similarity of the Reed Solomon based KPD to a *Transversal Design* $TD(q-1, q)$ with parameters $q-1, q$. However it is important to state that in $TD(q-1, q)$ design which has been explained in Theorem 6 of [10, section III] is slightly different from Reed Solomon code. In $TD(q-1, q)$ design, the evaluation is done for $y = 0, 1, \ldots, q-2$ while in Reed Solomon based design, it is done at non–zero points, i.e., $y = 0, 1, \ldots, q-1$. One must remember for $TD(q-1, q), k = q-1$ while noting that their scheme their scheme [10] can be extended to $\mathbb{F}_q, q = p^r$: a prime power.

N_0 to N_{15} denotes the nodes with ids ranging from 0 to 15 whose polynomials are represented in the column immediately below it. Key ids contained in a node are presented in the columns below each node. $V - C$ denoted the distinct *Variety Classes* H_1, H_2, H_3, where $H_d = \{(i, d) : 0 \le i \le 3\}$ for $d = 1, 2, 3$. One notes that the scheme under consideration is a $(q-1)(q-2)-$CID as the number of keys per node $= k = q-1$ (see section 1.5). Thus for nodes not sharing any key, there are enough nodes which can play the role of the intermediate node in multi-hop (2-hop) process. This encourages one to search for a deterministic design with exactly two merged nodes per block yielding full connectivity among the blocks.

Table 1. Polynomials, Node and Key identifiers for $q^2 = 16$ nodes. Table adapted from section 3.1 of Ruj and Roy [13]. Alternative presentation: Group–Divisible Design form.

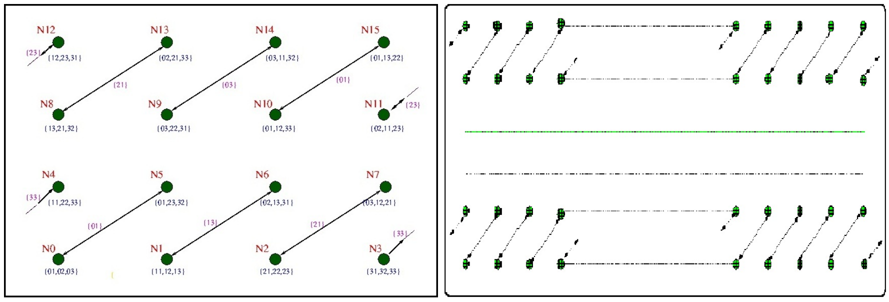
Nodes	N_0	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}
$V-C$	$0y+0$	1	2	3	y	$y+1$	$y+2$	$y+3$	$2y$	$2y+1$	$2y+2$	$2y+3$	$3y$	$3y+1$	$3y+2$	$3y+3$
H_1	(0,1)	(1,1)	(2,1)	(3,1)	(1,1)	(0,1)	(3,1)	(2,1)	(2,1)	(3,1)	(0,1)	(1,1)	(3,1)	(2,1)	(1,1)	(0,1)
H_2	(0,2)	(1,2)	(2,2)	(3,2)	(2,2)	(3,2)	(0,2)	(1,2)	(3,2)	(2,2)	(1,2)	(0,2)	(1,2)	(0,2)	(3,2)	(2,2)
H_3	(0,3)	(1,3)	(2,3)	(3,3)	(3,3)	(2,3)	(1,3)	(0,3)	(1,3)	(0,3)	(3,3)	(2,3)	(2,3)	(3,3)	(0,3)	(1,3)

2.1 Weakness of the above KPD Scheme [13,15]

The RS code based KPD in [13] lacks full connectivity among nodes (see section 3) and selective node attack is feasible (refer section 5). This results in multi-hop communications among the nodes which increases cost of communication as well as enhancing the chances of adversarial attacks on such communication. Thus the energy efficiency and security of message exchange of each cluster and hence the entire network of RS code based KPD in [15] might be grossly affected.

3 Remedy: Deterministic Merging of Nodes

Lack of direct communication for any arbitrarily chosen pair of nodes can be tackled by merging certain number of nodes yielding a network with fewer blocks, each possessing larger key rings. With this increased number of keys per block, one may expect improved communication between any given pair of blocks. Chakrabarti et al. [3] suggested the novel idea of random merging z nodes of a *TD(k,n)* based KPD proposed by Lee & Stionson [10] having similar weakness. Though *connectivity of the resultant model was much improved, full connectivity was not assured*. Other than this, the authors of [3] have not satisfactorily explained many aspects of their design, like the basic concept of merging, choice of nodes while merging, the heuristic in [3, section 4] etc.



(a) Deterministic Merging Strategy with $q = 4 = 2^2$.

(b) Deterministic Merging strategy: general even case: $q = 2^r$.

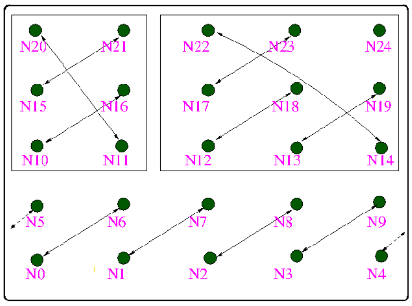
Fig. 1. Deterministic Merging Blocks Strategy on RS Based KPD scheme even cases of $q = 4, 2^r, r \in \mathbb{N}$

These observations prompt the current authors to think of deterministic merging of nodes of the scheme in [13] for the case $l = 2$. In this regard, it is worth noting that Table 1 indicates the network having 16 nodes can be partitioned into 4 classes each containing 4 nodes on the basis of their key sharing. These classes are separated by double column partitioning lines after each set of 4 nodes: N_0, N_1, N_2, N_3; N_4, N_5, N_6, N_7; N_8, N_9, N_{10}, N_{11}; & $N_{12}, N_{13}, N_{14}, N_{15}$. Every class has the property that the coefficient of y in their respective polynomials is same. Equating each other's polynomials $i + jy$ with $0 \le i \le 3$ for some fixed $j = 0, 1, 2$ or 3 results in no common solution, hence no common key. As for example with $j = 1$ with $i = 0, 1, 2, 3$, the corresponding 4 polynomials are $0+y, 1+y, 2+y, 3+y$, which does not have any common solution. Hence no shared keys for corresponding nodes.

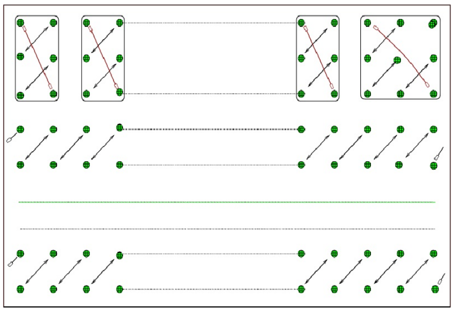
The other case for any pair of nodes not sharing any common key is whenever their constant term is same since only non–zero values of y are allowed. This gives rise to alternative type of partition: N_0, N_4, N_8, N_{12}; N_1, N_5, N_9, N_{13}; N_2, N_6, N_{10}, N_{14}; & N_3, N_7, N_{11}, N_{15}. This motivates one to visualize the key sharing of the 16 nodes, N_0 to N_{15}, like a 'square-grid' as presented in Figure 1(a). Any pair of nodes, other than the ones lying in the same row or column shares exactly 1 key as the equations: $(j-j')y = (i'-i)$ has unique solution over non–zero points of $\mathbb{F}_4$ (since $q=4$) that is with $0 \leq i \neq i', j \neq j' \leq 3$.

Merging of nodes in pairs for the case $q = 4, 2^r$ can be now achieved as indicated by the slanted line in Figure 1. In Figure 1(a) for $q = 4$, the nodes: $N_{i,j}$ & $N_{i\oplus 1, j\oplus 1}$ are merged. $j = 0, 2$. For $q = 2^r$, Figures 1(b) demonstrate the strategy. Idea in Figures 1(b) is to break up the network into pairs of rows, i.e. $\{1,2\}; \{3,4\}, \ldots, \{2^{r-1}, 2^r\}$ and apply similar process. Here $\oplus$: addition modulo 2 for both cases.

Before explaining the general odd case, it is useful to visualize the case when $q = 5$, i.e. a network with $q^2 = 5^2 = 25$ nodes as presented in Figure 2(a). Rest of the discussion is similar to that of the case $q = 4 = 2^2$ except for the merging of last three rows. As usual the arrows indicate the merging strategy. The strategy indicated in Figure 2(a) is same for the first and second rows while differs in the last three rows when compared to Figure 1(a). This is because a similar strategy like previous cases would imply one row is left out. Of course for $q = p = 5$ all arithmetic operations are 'modulo 5' arithmetic operations as done in $\mathbb{F}_5$.



(a) Deterministic Merging Strategy with $q = p = 5 \implies 25$ nodes.



(b) Deterministic Merging strategy: general odd case: $q = p^r, r \in \mathbb{N}$.

Fig. 2. Deterministic Merging Blocks Strategy on RS Based KPD scheme for $p = 5, r = 1 \implies q = 5$ and all odd cases $q = p^r, r \in \mathbb{N}$

The general case of $q = p^r$:odd prime power, q^2 nodes ($l = 2$) can visualized as a $q \times q$ 'square-grid' as in Figure 2(b) where the merging is indicated. Nodes in same row or column having do not share common key while any other pair of nodes have exactly one common key. Nodes occurring at the ends of a slanted line are merged. Like for $q = 5$, the idea is to look at two rows at a time and form blocks containing one node of each except for last 3 rows. For fixed $0 \leq i \leq q-2$, merge the nodes $N_{i,j}$ & $N_{i\oplus 1, j\oplus 1}$($\oplus$: addition modulo q), for $0 \leq j \leq q-3$ (with increment of 2) $\forall\ q > 4$. The last node of every odd row is merged with the first node of the row above it. Taking combination of two row would have left out one row, so top three row are combined separately. Figure 2(b) explains the merging for this case.

Note that in case merging of nodes is done randomly, one may end up with merged pairs like $N_{0,0} \cup N_{0,1}$ and $N_{0,2} \cup N_{0,3}$, (for $q \geq 4$) which do not share any common key, thus not be able to communicate even after merging.

3.1 Assured Full Connectivity: Theoretical Results

Equating the polynomials of the 4 nodes constituting any two merged blocks, one readily sees that:

Theorem 1. *The proposed deterministic Merging Block Strategy where two nodes of the Reed Solomon based KPD scheme in [13] are clubbed to form the merged blocks results in full connectivity among the merged blocks.*

Proof. Consider any two arbitrary blocks A and B. It is evident from the construction that at least node from block A will never lie in the horizontal line as well as the vertical line of either of the two nodes the other block B (refer figures 1(a), 1(b) 2(a) and 2(b) for $q = 4, 2^r, 5$ and for general case respectively). This implies that these two nodes will have a common key as discussed in section 3. Hence the blocks A and B can communicate through this key. As the two blocks were arbitrarily chosen, one is assured of full connectivity in the new network consisting of blocks constructed by merging two nodes in the manner explained above (again refer figures 1(a), 1(b) 2(a) and 2(b) for $q = 4, 2^r, 5$ and for general case respectively).

Theorem 2. *The resulting Merged Block Design has a minimum of one* (1) *to a maximum of four* (4) *common keys between any two given pair of (merged) blocks.*

Proof. Any two nodes can share at most one key in original Reed Solomon based KPD in [13]. So there are at most 4 keys common between two blocks. This situation occurs only if both nodes of the 1st block shares two (2) distinct keys with each node of the 2nd block.

Remark 1. Some important features of the merging block design are as follows:

- Thus the resultant merged block design (as suggested here) has full connectivity among the blocks through at least one common keys between any two given pair of (merged) blocks.
- Full communication can not be assured when nodes are merged randomly to form larger blocks. Probably this is the main reason why authors of [3] could not justify several issues in their random merging model. Table 2 indicates that even the resiliency ($E_1(s)$) of the current stragy is better than those ($E_{CMR}(s)$) of [3].
- The current authors feel that it is mandatory to have inter nodal communication otherwise the entire concept to achieve full connectivity by this technique is lost. The essence of the merging concept is that any communication is always received by

either of the two constituent nodes of a block. Now if required it can pass the information down to the other node comprising its block and hence make the other node connected. This consideration was given importance while proposing the merged design. Evaluation was done at $y = 2, \frac{p-1}{2}, p^t - 1, 1 \le t \le r$.

– It should be highlighted that merging does not mean that the nodes physically combine to become one. Just that they are to be treated as one unit.
– So, the total number links in the merged scheme is same as that of the original Reed Solomon based KPD of Ruj & Roy in [13]. This fact will be recalled later while discussing resiliency of the system.

4 RS Code Based Simonova et al. KPD

In their work [15], Simonova et al. considered a network of p^4 nodes for any prime p and subdivided them into cluster of p^2 many. Each cluster was modeled using Lee & Stinson classic Transversal design of [10]. Inter cluster connectivity was ensured by amalgamating the key rings of each nodes. They had denoted the original key pool as K_{orig} and the new key pool after amalgamation as K_{depl}. However there was a small error in calculating the group–divisible form. Of course one can utilize RS code based KPD [13] to distribute the keys in each cluster. Accordingly their strategy get modified for the current application which is explained below:

– For each cell deployment key pool $\mathscr{X}$ is the union of the original key pools of $m \times m$ cells. This means that the size of the finite set $\mathscr{X}$ is $K_{depl} = |\mathscr{X}| = m^2 K_{orig}$.
– Each original key pool contributing to the deployment key pool serves as a group H_g, so the set of groups consists of $k_{depl} = m^2(q-1)$ groups.
– Each group H_g consists of $K_{orig} = q$ elements.

Rest of the details can be looked up in [15, section 3.3].

4.1 Full Connectivity: Minimum Keys

As TD design based KPD was provoked for inner cluster communication, full connectivity was not achieved inside each cluster, hence not in the entire network. Merging block design presented in section 3 addresses this issue assuring full inner cluster communication among blocks and hence full connectivity in the whole network. However the network size is halved consisting of blocks of 2 nodes with $2k_{depl} - 1, k_{depl} = m^2(q-1)$ keys per ring.

5 New Heterogeneous Design

Wireless sensor nodes are generally very resource constrained and their cost varies proportionately to the amount of resource inbuilt into them. It may not be affordable to use costly sensor nodes having more amount memory. The key predistribution model of Simonova *et al.* may become very demanding when block merging technique is applied to it. Though minimum key concept using merging block may reduce this storage problem, better solution is necessary. In Simonova *et al.* the network size is correlated to number of keys one node can contain. To circumvent this trade one may think of using a heterogeneous design. A heterogeneous network consists of different types of nodes varying mainly in resources they possess. One may think of implanting exactly one cluster head (CH) into the q^2 many clusters of the network giving rise to a hierarchical structure with KDS at top of the ladder, followed by a set of q^2 many CHs in the 2^{nd} tier beneath each lies q^2 many nodes.

In each cluster any KPD may be used like the same one based on TD of [10]. In addition to the $q-1$ keys corresponding to the TD design, each node is preloaded with exactly one key that links it to its CH. Thus each CH have q^2 keys one for each of the node in the lower tier. For communication with other CHs at the same tier, each CH holds another set of q^2-1 keys and one additional key to link it with the Base Station. Thus CH contains a total of $2q^2$ many keys. While the nodes now have $(q-1)+1=q$ many keys in them.

This scheme similarly does not have full connectivity, which can be addressed by the merging technique. Nodes in each cluster are merged to form blocks in absolutely the same manner as before. However, now each block contains $2(q-1+1)-1=2q+1$ keys. Communication of nodes of different cluster takes place via the CHs of the respective clusters. So one obtain a fully connected network till the RF range of the CH. As the CHs are more resourceful and can be given extra security as they are less in number, the overall communication is fast and secure. Key establishment protocols in section 6 of both design, viz. Simonova *et al.* and this elaborate how 'selective-node-attack' is possible in former but gets eradicated here. The current concept differs from the one proposed in [17] as the notion of separating connectivity and communication is not needed here. The extra key for CH does the job.

6 Key Establishment Protocol

During key establishment phase, in Simonova *et al.* model [15], the nodes first exchange their identifiers. Since these identifiers are public information, they can be transmitted in unencrypted form. Upon receiving ids of the other nodes, the nodes compute the key-id of the shared key in a similar way as in the TD model of [10]. Here one notes the the adversary can do the same and can compute the common id of the common key between the two nodes. However the adversary can not know the actual key until he compromises some node containing that key. This makes 'selective node attack' feasible.

This inspires the authors to think of encrypting the node ids during transmission in the new design. Each node $N_{(a,b)} = N_{aq+b}$ is deployed with following data preloaded into them. These are utilized for key establishment according to the protocol described in Algorithm 1:

- The extra key in each node meant for its CH;
- KPD keys compromising their key ring;
- key ids corresponding to each key in their key ring;
- their respective polynomials and the node ids.

Establishing keys for nodes automatically establishes common keys between the merged blocks.

```
1 Immediately on deployment, the nodes send their ids to their CH encrypted with
  common key shared with their CH's;
2 On receiving each of the node ids $aq+b$ (without being reduced modulo $q$), the CH
  form the polynomial $ay+b$ corresponding to the node ids;
3 These polynomials are equated for common key ids in much the same way as was
  done by nodes in Transversal Design [10] or its amalgamated design Simonova et al.
  [15];
4 The common keys are indicated back to the nodes using some indicator in an
  encrypted manner. Again the node-CH keys are used. ;
5 Nodes decrypts the encrypted indicator using the share node-CH keys ;
6 Knowledge of the merging technique is applied for establishing shared keys between
  nodes, automatically establishes shared keys between any pair of blocks;
7 Keys thus established are used for message exchange;
```

Algorithm 1. Key Establishment Protocol

Since the CHs are doing most the operations in this protocol, the process is fast and secure. Hence 'selective node attack' is ruled out in the new scheme.

7 Resiliency: Comparative Study

In this section the resiliency of the all the schemes analyzed are discussed in terms of a well known measure called $E(s)$, which helps in comparing the various schemes studied. $E(s)$ is defined to be the fraction of links disconnected when s nodes (in original) or blocks (in merged) are captured by the adversary and all the keys in them are exposed to the adversary.

Let $q-1$ be the number of keys per node. In RS code based KPD in [13], the replication number of a certain key is q. Hence every key occurs in q many nodes. If the adversary compromises s nodes then the total number of keys she obtains is sk all of which may not be distinct. Let s_{dk} be the number of distinct keys that the adversary gets to know by compromising s many nodes. Since one key is common between q many nodes, one key corresponds to $\binom{q}{2}$ many links of the WSN. So, s_{dk} keys contribute to $s_{dk}\binom{q}{2}$ many links. Hence the number of links exposed is $s_{dk}\binom{q}{2}$. Since, there are $\binom{q}{2}$ distinct links per key and and total of $q(q-1)$ keys in the key pool of RS code design [13], the total number of link $= q(q-1)\binom{q}{2}$ for each cluster. Hence, $E(s) = \frac{s_{dk}\binom{q}{2}}{q(q-1)\binom{q}{2}}$ for original RS code based KPD in [13].

From the construction of the merging blocks, it is clear that a merged block network of size N corresponds to a original KPD of size roughly $2N$. Capture of s merged blocks is equivalent to capturing $2s$ nodes of original KPDs. So here the ratio of links broken $E(s) \approx$ old $E(2s)$ of each of the original KPDs. The tabulated values in Table 2 confirm these observations.

As for the resiliency of new design in section 5 and Simonova *et al.* scheme [15], suppose $S = s_1 + s_2 + \ldots + s_{q^2}$ be the number of compromised nodes, where s_i =number of compromised nodes in cluster $i, 1 \leq i \leq q^2$. The for the new model of section 5,

$E(S) = \sum_{i=1}^{q^2} E(s_i)$, where $E(s_i)$ =resiliency of RS code based KPD in [13] design on compromise of s_i nodes. This is due to the fact that each cluster is independent of the other in terms of connectivity and are linked by their respective CH. Whereas in Simonova *et al.* scheme [15], capture of s_i nodes affects minimum of 1 to maximum of m^2 other nodes. The exact value, dependent on m is difficult to predict. Simulation results are given in Table 2.

8 Connectivity, Scalability and Communication Overhead

Connectivity is defined to be the probability that any given pair of nodes (in original KPDs) or blocks (in their merged designs) can communicate directly, i.e. share common cryptographic key(s). From corollary 2 of Theorem 1, *connectivity is* 1.

As only linear, quadratic and cubic equations have general solution, key establishment involving solving $(l-1)^{th}$ degree can be done till $l = 4$ for the KPD in [13]. Thus this KPD scheme with $l = 2 \implies$ having q^2 nodes *scales* to q^3 to q^4 nodes for $l = 3,4$ respectively. Accordingly one has to think of new merging strategy.

Communication overhead measures the computational complexity of first the key establishment and then message exchange. It is well known that linear, quadratic and cubic equations have general solution. Key establishment involving solving $(l-1)^{th}$ degree can be done till $l = 4$. Though quadratic and cubic equations can be solved in constant time, the *complexity or communication overhead* of finding the solution to these equations is quite high. Thus with increasing number of nodes in the original KPD, the *resiliency drops abruptly* and the *complexity of key establishment* increases rapidly. So, practically $l = 2$ case is considered for the basic design [13] which is used to distribute keys in the clusters of both Simonova et al. and the new design.

9 Simulation and Comparative Results

Simulation is performed to monitor the achieved effectiveness of the proposed model in this paper in terms of resiliency over the model proposed by Simonova et al. [15] and the original KPD [13].

The run results after 100 runs for each data set are tabulated in table 2 where $\mathcal{N}_{mb} = \lfloor \frac{\mathcal{N}_{RS}}{2} \rfloor, l = 2 \implies k = q-1$. $\mathcal{N}_{RS}$ $(= q^2)$ and $\mathcal{N}_{mb}$ denotes total number of nodes and merged blocks in the network respectively. s_{mb} is the total number of *nodes compromised*. As noted, the effect of compromising $2s$ nodes of the original network is equivalent to s compromised blocks in the merged network. $E_1(s), E_{RS}(s)$ denotes the resiliency coefficient in the merged and original network respectively. While $E_{CMR}(s)$ denoted the corresponding data adapted from [3]. $E_2(s), E_3(s)$ are the resiliency coefficients of the merging block design over Simonova et al. [15] and the original KPD [13] respectively. The entire network size is accordingly taken to be $\mathcal{N}_{mb}^2$ and $\mathcal{N}_{RS}^2$ respectively.

10 Conclusions and Future Research Directions

This paper investigates the connectivity issue of a KPD scheme by Simonova et al. [15] where the network is subdivided into clusters. The classic TD based KPD in [10]

Table 2. Simulation & comparative results for $E(s)$ for the original KPDs and their merged design

$\mathcal{N}_{RS}$	$\mathcal{N}_{mb}$	s_{mb}	$E_1(s)$	$E_{RS}(s)$	$E_2(s)$	$E_3(s)$	$E_{CMR}(s)$
841	420	5	0.296121	0.296707	0.182103	0.186332	0.317145
1681	840	5	0.219025	0.219159	0.216666	0.155608	0.234575
2401	1200	5	0.186576	0.186709	0.157999	0.139810	0.199822
2401	1200	10	0.338858	0.338890	0.231038	0.199857	0.362916
5041	2520	10	0.247530	0.247352	0.201631	0.170041	0.265104
10201	5100	10	0.180536	0.190590	0.179905	0.136963	0.193354
10201	5100	20	0.328524	0.329059	0.261768	0.200380	0.351849
10609	5304	10	0.177394	0.177280	0.188190	0.136963	0.189988
10609	5304	30	0.443347	0.443746	0.315809	0.233972	0.4748246

was utilized for predistribution of keys in each cluster of [15] which is replaced by the RS code based KPD in [13]. Hence the work here can be thought of amalgamating q^2 networks (treated as clusters) build with the same KPD of [13] with $q-1$ keys per node. The deficiencies like lack of connectivity and feasibility of 'selective node attack' the original RS code based KPD are naturally carried over to the equivalent of Simonova et al. Deterministic merging of nodes forming blocks within clusters address the issue of connectivity of not only the clusters, but of the entire network. Combinatorial approach helps one view the similarity between the original design in [10] while a new hierarchy based KPD using RS code for clusters solves the node attack problem. This new model requires q^2 many CHs and has q keys per node as opposed to $m^2(q-1)$ keys per node of Simonova et al. model using RS code. Combing merging blocks with the new strategy addresses both the issues.

One readily visualizes some immediate future research directions. Application of similar deterministic merging concept to network based on other KPDs like [12,9] lagging full connectivity among its nodes like may result in interesting works. The reason of preferring such merging strategy over its random counterpart has been sketched. More detailed generic survey of deterministic verses random merging of nodes yielding fully communicating networks can be nice topic of future research. This paves a direction which may fulfill the target of achieving fully communicating deterministic schemes having high resiliency. A priori one must look to design scheme having good node support, small key rings, high resilience and scalability. Mathematical solutions to such fascinating problems will be interesting.

Acknowledgement The authors would like to extended their gratitude towards the Ph.D. supervisors of Mr. Pinaki Sarkar, viz. Dr. Indranath Sengupta, Department of Mathematics, Jadavpur University and Prof. Subhamoy Maitra, ASU, ISI Kolkata, both of whose inspiration has helped this work to become a success. This work is meant to be a part of Ph.D. thesis of Mr. Pinaki Sarkar. The inputs from Mr. Sabyasachi Datta of Pure Mathematics Department, University of Calcutta and Miss. Amrita Saha are worth mentioning and hence need to be thanked heartily. The authors would also like to sincerely express their gratitude towards Mr. Sumit Kumar Pandey and Mr. Sourav Sengupta for their constant motivation and help related to technical issues during the drafting of the paper.

References

1. Çamtepe, S.A., Yener, B.: Combinatorial Design of Key Distribution Mechanisms for Wireless Sensor Networks. In: Samarati, P., Ryan, P.Y.A., Gollmann, D., Molva, R. (eds.) ESORICS 2004. LNCS, vol. 3193, pp. 293–308. Springer, Heidelberg (2004)
2. Çamtepe, S.A., Yener, B.: Key distribution mechanisms for wireless sensor networks: A survey 2005. Technical Report, TR-05-07 Rensselaer Polytechnic Institute, Computer Science Department (March 2005)
3. Chakrabarti, D., Maitra, S., Roy, B.: A key pre-distribution scheme for wireless sensor networks: merging blocks in combinatorial design. International Journal of Information Security 5(2), 105–114 (2006)
4. Chan, H., Perrig, A., Song, D.X.: Random key predistribution schemes for sensor networks. In: IEEE Symposium on Security and Privacy. IEEE Computer Society, Los Alamitos (2003)
5. Du, W., Deng, J., Han, Y.S., Chen, S., Varshney, P.K.: A key management scheme for wireless sensor networks using deployment knowledge. In: INFOCOM (2004)
6. Du, W., Deng, J., Han, Y.S., Varshney, P.K.: A Pairwise Key Pre-distribution Scheme for Wireless Sensor Networks. ACM Trans. Inform. Syst. Secur. 8, 228–258 (2005)
7. Di Pietro, R., Mancini, L.V., Mei, A.: Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. Wireless Networks 12(6), 709–721 (2006)
8. Eschenauer, L., Gligor, V.D.: A key-management scheme for distributed sensor networks. In: ACM Conference on Computer and Communications Security, pp. 41–47 (2002)
9. Lee, J.-Y., Stinson, D.R.: Deterministic Key Predistribution Schemes for Distributed Sensor Networks. In: Handschuh, H., Hasan, M.A. (eds.) SAC 2004. LNCS, vol. 3357, pp. 294–307. Springer, Heidelberg (2004)
10. Lee, J.Y., Stinson, D.R.: A combinatorial approach to key predistribution for distributed sensor networks. In: IEEE Wireless Communications and Networking Conference, WCNC 2005, New Orleans, LA, USA (2005)
11. Lee, J.Y., Stinson, D.R.: On the construction of practical key predistribution schemes for distributed sensor networks using combinatorial designs. ACM Trans. Inf. Syst. Secur. 11(2), 5:1–5:35 (2008)
12. Liu, D., Ning, P.: Establishing pairwise keys in distributed sensor networks. In: ACM Conference on Computer and Communications Security, pp. 52–61. ACM, New York (2003)
13. Ruj, S., Roy, B.: Key Predistribution Schemes Using Codes in Wireless Sensor Networks. In: Yung, M., Liu, P., Lin, D. (eds.) Inscrypt 2008. LNCS, vol. 5487, pp. 275–288. Springer, Heidelberg (2009)
14. Ruj, S., Roy, B.K.: Key predistribution using combinatorial designs for grid-group deployment scheme in wireless sensor networks. TOSN 6(1) (2009)
15. Simonova, K., Ling, A.C.H., Wang, X.S.: Location-aware key predistribution scheme for wide area wireless sensor networks. In: SASN, ACM, pp. 157–168 (2006)
16. Steiner, J.G., Neuman, B.C., Schiller, J.I.: Kerberos: An authentication service for open network systems. In: USENIX Winter, pp. 191–202 (1988)
17. Sarkar, P., Saha, A., Chowdhury, M.U.: Secure Connectivity Model in Wireless Sensor Networks Using First Order Reed-Muller Codes. In: MASS 2010, pp. 507–512 (2010)
18. Stinson, D.R.: Combinatorial Designs: Construction and Analysis. Springer, New York (2004)
19. Street, A.P., Street, D.J.: Combinatorics of Experimental Design. Clarendon Press, Oxford, ISBN 0198532555
20. Yu, Z., Guan, Y.: A key management scheme using deployment knowledge for wireless sensor networks. IEEE Trans. Parallel Distrib. Syst. 19(10), 1411–1425 (2008)

Learning Fine-Grained Structured Input for Memory Corruption Detection

Lei Zhao[1,2], Debin Gao[2], Lina Wang[1]

[1] Computer School of Wuhan University, Wuhan, China
[2] School of Information Systems, Singapore Management University, Singapore
zhaolei.whu@gmail.com, dbgao@smu.edu.sg, lnwang@whu.edu.cn

Abstract. Inputs to many application and server programs contain rich and consistent structural information. The propagation of such input in program execution could serve as accurate and reliable signatures for detecting memory corruptions. In this paper, we propose a novel approach to detect memory corruptions at the binary level. The basic insight is that different parts of an input are usually processed in different ways, e.g., by different instructions. Identifying individual parts in an input and learning the pattern in which they are processed is an attractive approach to detect memory corruptions. We propose a fine-grained dynamic taint analysis system to detect different fields in an input and monitor the propagation of these fields, and show that deviations from the execution pattern learned signal a memory corruption. We implement a prototype of our system and demonstrate its success in detecting a number of memory corruption attacks in the wild. In addition, we evaluate the overhead of our system and discuss its advantages over existing approaches and limitations.

Keywords: memory corruption, dynamic taint analysis.

1 Introduction

Memory corruption exploits usually involve overwriting significant memory segments such as return addresses and function pointers [21]. Typical memory corruption exploits include control-hijacking attacks (e.g., [10]) and non-control data attacks (e.g., [8]). Despite having a long history, memory corruption exploits are still one of the biggest challenges to computer security [19].

Many techniques have been proposed to fight against memory corruption exploits, e.g., secure language [20,14], bug detection [17], safe library [25], bounds checking [26], etc. Some of these techniques require access or even changes to the source code (e.g., [3,7,20,14]), which might not be suitable when dealing with commercial off-the-shelf applications. Dynamic approaches which do not require source code of the program include canary-based techniques [10], probabilistic defenses [4], runtime enforcement [25], dynamic taint analysis [21], control flow integrity [2], etc. These binary-level techniques are powerful and efficient against many attacks, however they also suffer from some limitations [3].

For example, dynamic tainting such as TaintCheck cannot detect non-control data attacks [21]. Pointer tainting [18] may result in a large number of false alarms because of legitimate use of input as pointers [23].

In this paper, we investigate how fine-grained taint analysis and propagation of program inputs could fight against memory corruptions. Program inputs usually contain rich structural information, which has been shown to be useful in a number of security applications [13,16]. Intuitively, programs usually parse an input into various fields, which contain independent semantics and are subsequently processed by different instructions [6,15]. A consistent and reliable pattern on the fine-grained structure of an input and the corresponding processing of it by different instructions could be used to capture normal execution of the program, since memory corruption exploits usually violate program semantics, e.g., the overflowed bytes are not processed by intended instructions. Thus the processing of exploits may not be consistent with that in benign executions. With this observation, memory corruptions could be detected by monitoring deviations of program execution from the pattern of input processing.

There are some difficulties in realizing such an intuition. First, the fine-grained structural information of the input might not be known. This could be due to a proprietary protocol used or lack of documentation. Moreover, even if a protocol description is available, it is usually not implementation specific, which might introduce noise to the detection (e.g., two independent fields in an input might be processed by the same instructions in a similar way in an optimized implementation). Second, it is unclear how to model the propagation and processing of the fine-grained structural input and to define patterns to catch the deviations. Third, existing dynamic tainting systems do not support the monitoring of fine-grained structural inputs.

We propose a novel technique called FiGi to monitor fine-grained input information for detecting memory corruptions. FiGi extends existing dynamic taint systems to enable precise monitoring of the propagation of individual input bytes (and their corresponding taint tags) during program executions. It learns and extracts structural information in program inputs by analyzing the execution context of every input byte. To normalize the input structures for specific inputs, FiGi uses a tree structure to model the input as well as its propagation in program executions, and constructs normal patterns by monitoring benign executions of the program. We implement FiGi and demonstrate its success in detecting a number of memory corruption exploits in the wild, include both control-hijacking and non-control data attacks. We additionally evaluate the overhead in using FiGi, and discuss its advantages over existing dynamic techniques in detecting memory corruption exploits as well as its limitations.

2 Related Work

The closest related work to FiGi is dynamic taint analysis which has been proposed to detect attacks, to diagnose vulnerabilities, and to generate attack signatures. TaintCheck [21] detects attacks that overwrite return addresses, function pointers, format vulnerabilities, and in general control-related memory corruptions [8]. Pointer tainting detects non-control data attacks [18], but over-tainting causes a large number of false alarms [23]. FiGi differs from these existing dynamic tainting systems in the granularity of tainting in that FiGi assigns a different taint tag to each individual byte of the input and monitors the propagation of every taint tag. As we will show in Section 5, FiGi is able to detect both the control-hijacking attacks and non-control data attacks.

Clause et al. proposed tainting memory allocation to improve memory safety [9,12]. Instead of tainting the input of the program, they taint the memory segments and the corresponding pointers for dynamic memory allocations on the heap and stack. However, a drawback of their approach is that compiler optimization may consolidate multiple memory regions (especially for local variables) into a single allocation request, which hides the granularity required for effective detection of memory corruptions. FiGi, on the other hand, does not suffer from this limitation because our tainting is fine-grained and not limited to memory allocations.

A key step in FiGi is to learn the structure of inputs, which is closely related to protocol reversing [6,15]. FiGi employs a similar idea as these protocol and format reversing techniques. However, the different scenarios in which the recovered structure is used pose very different requirements on the discovery of the input structure. FiGi uses the input structure to detect memory corruptions exploits, which requires very fine-grained input structure to be learned; on the other hand, protocol and data structure reversing could benefit from grouping multiple fields of the input together as long as the execution context is similar [15].

3 A Motivating Example and the Challenges

In this section, we present a motivating example to demonstrate the idea of FiGi as well as the challenges involved.

```
char ifname[MAX_PATH_LEN]; /* input file name */
char ofname[MAX_PATH_LEN]; /* output file name */

if(make_ofname() != OK) return;
ifd = OPEN(ifname, ascii && !decompress, RW_USER);

local int make_ofname() {
strcpy(ofname, ifname); //memory corruption
```

Fig. 1. A motivating example of vulnerable gzip

Fig. 1 shows a vulnerable code segment of `gzip-1.2.4` with an unsafe string copy from `ifname` to `ofname`. If the length of the filename (part of the input) is larger than 1024 bytes, `strcpy` will result in an overflow of the buffer for `ofname`. Note that FiGi works on the binary level. The source code and the corresponding memory layout of variables in Fig. 1 and Fig. 2 are presented for clearer explanation only. FiGi does not need to know anything about the high-level symbols such as `ifname` and `ofname`.

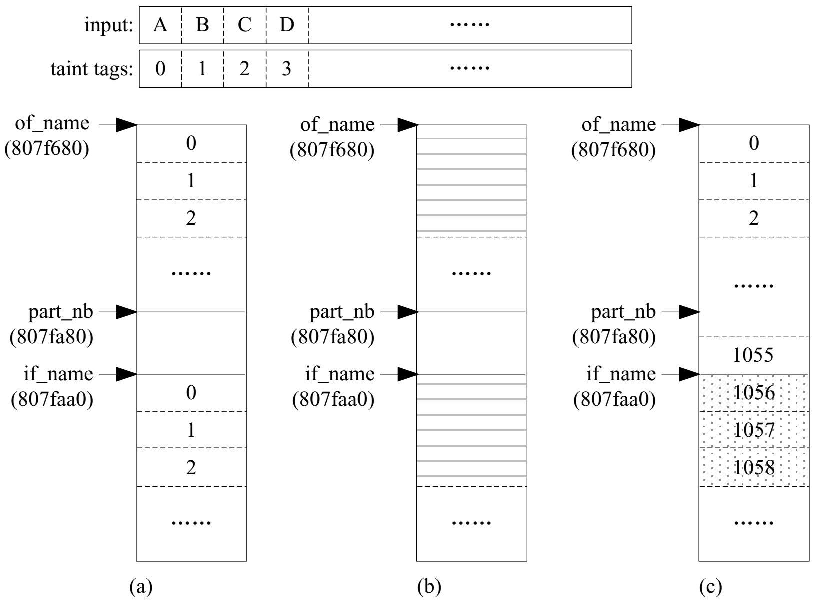


Fig. 2. Memory layout of gzip. `0x807f680`, `0x807fa80` and `0x807faa0` are the beginning memory addresses of `ifname`, `part_nb` and `ofname`, respectively.

We first examine what FiGi could learn from the fine-grained structural input in benign executions of `gzip`, when each byte of the command line input to `gzip` is assigned a different taint tag. Fig. 2(a) shows a portion of the memory layout in one particular execution of `gzip`, and we can see that each byte on the stack comes with a unique taint tag. At a closer look into the instructions processing the tainted input, we also realize that the tainted input bytes are stored into two buffers of which the memory addresses start from `0x807f680` and `0x807faa0`, respectively. Moreover, the bytes are all processed by the same instructions (`strcpy` and `open`) continuously during program executions. That is, the tainted input is processed as one unit. Therefore, FiGi could derive that the entire input contains only one field (see Fig. 2(b)).

To see the advantage of monitoring the fine-grained structural input, we consider a typical memory corruption exploit with inputs of 1200 bytes long at offset 0–1199 as shown in Fig. 2(c). When the input is longer than 1056 bytes, the unsafe string copy will overwrite segments of `part_nb` (the memory address starts from `0x807fa80`) and `ifname` (the memory address starts from `0x807faa0`). Bytes with tags 1056–1199 would be accessed when `open` executes. FiGi would realize that the input portion with offset 1056–1199 is accessed by two instructions, namely `open` and `strcpy`, while the input portion with offset 0–1055 is accessed by `strcpy` only. Given the assumption that the input fields contain independent semantic and are subsequently processed by different instructions, from the exploit execution, the two portions with offset 1056–1199 and 0–1055 form two independent fields. This constitutes a deviation from what FiGi had learned from training of `gzip`, and triggers an alarm. Note that this deviation could not have been detected without a fine-grained monitoring of the structural input. Also note that such a memory corruption cannot be detected by TaintCheck [21] because tainted data does not change any function pointers or format characters.

Although this motivating example shows some intuition as to how the fine-grained structural input and its propagation during program executions could help detecting memory corruptions, there are some challenges we face.

1. The structural information of the input might not be known, either due to a proprietary protocol in use or lack of documentation. FiGi assumes that the only information available is the binary program as well as some training inputs. Therefore, we need to design FiGi in such a way that the structural information is learned via training.
2. It is unclear how to best model the propagation and processing of the fine-grained structural input to catch memory corruptions and to minimize false alarms. As we shall explain in the next section, FiGi keeps multiple patterns for each program to minimize false alarms.

4 System Design

4.1 Patterns and Deviations

We design two types of patterns in FiGi based on the observations that 1) the input structure is well-defined and consistent; and 2) fields accessed by different instructions are independent of one another. With these two observations, we capture the independent fields and the input structures. At the same time, for each independent field, we capture the execution context in which the field is processed as well.

Definition 1. *An independent field of the input consists of several continuous bytes which are always accessed as one unit in the program execution.*

Definition 2. *The accessing location of an independent field is the program execution context accessing the bytes of this field, which include the current call stack as well as the accessing instructions.*

With the two patterns, a program execution processing one input could be represented as follows.

Definition 3. *An input processing is denoted as $R(i) = \langle S, F, E \rangle$. In this formula, $R(i)$ refers to the program execution with the input i. S refers to the structure of i. $F = \{f_1, f_2, ..., f_n\}$ refers to the set of fields. $E = \{e_1, e_2, ..., e_m\}$ refers to the set of accessing locations.*

The input structure is specific with i (e.g., the offset interval of fields). With this impact, the field sets cannot be directly compared, and a normalized representation is required to generally represent the structured input. We will demonstrate this part in Sec. 4.5. Beside, there are mapping relationships between F and E. f_i is mapped with with more than one e_j, which indicates that one field could be accessed by several execution contexts, respectively.

For an unknown execution $R(i)$, the deviation is detected if 1) S is not matched with any of benign inputs, or 2) even S is matched, but the e_j of f_i cannot be matched. The first condition makes sense because the abnormal execution violates the input structures due to the misuse of corrupting bytes by unintended instructions. However, it is possible that the exploit could break and interrupt the input parsing, especially when the program is control hijacked and jumps to illegal instructions. These cases are common in multiple vulnerable programs (e.g, `AT-Tftp`, `ghttpd` in Sec. 5). As a result, we could only get a partial structure and the difference of structures may not be clear. In such cases, the second condition makes sense because a specific fields can only be accessed in several specific accessing locations. In Sec. 5.2, we will use the `ghttpd` program to illustrate.

4.2 Overview

As demonstrated in Sec. 3, we cannot expect the structural information of inputs is known, and we need to extract and learn patterns on input processing from benign executions. To overcome these challenges, the basic idea of FiGi is to model the propagation and processing of the fine-grained structured input through dynamic execution monitoring. Fig. 3 shows an overview of FiGi.

1. We adopt the dynamic taint analysis to dynamically monitor the program executions as well as the propagation of inputs. In addition, we extend the dynamic taint analysis to enable precise monitoring of the propagation of individual input bytes (and their corresponding taint tags), such fine-grained propagation could be used to capture the structured input.
2. To model the input processing and learn some patterns, we identify fields of the input via execution context comparison. This approach is similar with the protocol/format reversing techniques [15], except the difference that the field identification is performed online for the purpose of emergency responses to memory corruption attacks. Moreover, for exploits, the input structure may not be sufficient for deviation detection, because the memory corruption attack could break and interrupt the input parsing, from which the deviation cannot be observed and lead to false negatives. For effectiveness enhancement, we also collect the corresponding processing contexts for each

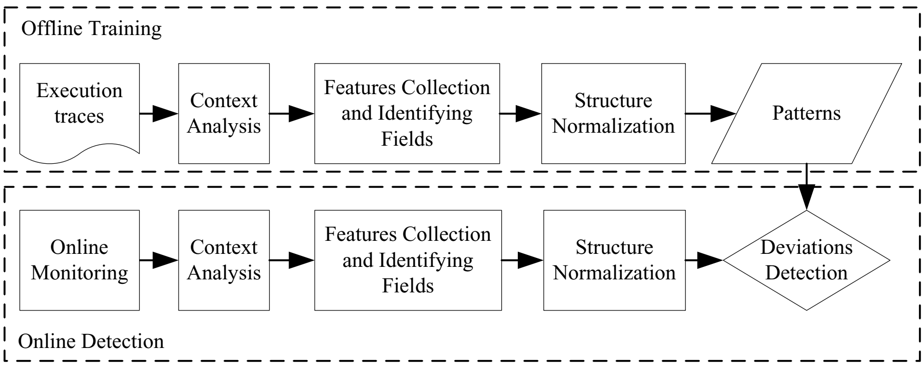


Fig. 3. The overview framework of our model

independent field based on the intuition that one field could be accessed in several specific execution contexts.

3. The identified fields are represented by the taint tags of a specific input. Since some fields are not length-fixed and some fields are optional, we need a normalized representation of structural inputs to compare the structures and figure out the deviations. We design to use the tree structure to model the structured input based on the observation that the input is processed subsequently and the input parsing looks like the construction of a tree.
4. By analyzing the execution context and identifying fields of benign executions, we generate patterns of the input processing and propagation, and use them to detect the deviation of executions with unknown inputs. The deviation detection is performed online. In details, we dynamically analyze the execution context and the propagation of fine-grained structured inputs to identify fields and collect patterns, whenever a field is identified, update $R(i)$ and perform deviation detection on $R(i)$ with trained ones.

4.3 Execution Monitoring and Context Tracking

To monitor the propagation of inputs and precisely capture the execution context, we extend the dynamic tainting [21,6] to give each byte a unique taint tag. The taint tag includes two items, the taint source and the taint offset, which refer to the source of the input and the byte offset, respectively. Among the taint propagation, we make the destination operand has the union taint record for multiple source operands. For example, suppose the instruction is `ADD %eax, %ecx`. The taint records of `%eax` and `%ecx` are $\{1001, 3\}$ and $\{1001, 7\}$, respectively. After the execution, the taint record of `%ecx` will be $\{\{1001, 3\}\{1001, 7\}\}$.

During execution monitoring, we record two types of execution context information: the run-time call stack and the address of instructions that access tainted bytes [15]. To acquire the run-time call stack, we monitor the the function call and return instructions, as well as the stack frame balancing. For each taint byte, we make every tainted instructions as well as the call stack to form an execution sequence of this taint byte. Then we compare the execution sequences of taint bytes with continuous offsets, if the execution sequences could be matched, we regard the continuous bytes belong to the same field.

4.4 Identifying Fields and Collecting Patterns

In the dynamic protocol reversing techniques, the execution context analysis are performed off-line [6] and the inputs are assumed benign [6,15]. For our problem scope, we cannot wait for the program to exit and should sponsor quick response to the anomaly as early as possible. That is, the execution context comparison and identifying fields should be performed online.

To identify fields online, an intuitive approach is to compare the execution context of a captured taint byte with those of its continuous bytes whenever the taint byte is captured. However, the taint bytes are dynamically processed and the order of accessed taint bytes is an undecided problem (e.g., the program could access another field such as a separator between the period of accessing two continuous bytes). To overcome the problem, we design to perform the context comparison whenever a function returns that causes the call stack changes. For every calling of a function, we allocate a data structure to store the execution sequences of taint bytes which are accessed within the current call stack. When

Algorithm 1. Identifying Fields

```
   Data: InstAddr($t_b$): the instruction address accessing the taint byte $t_b$
   Data: ExecSeq[index]: the data structure storing the execution sequences
   Data: ExecSeq[index][$t_b$]: the execution sequence of the taint byte $t_b$
   Result: ⟨ offset intervals, accessing locations ⟩
1  while instruction i do
2      if i is tainted then
3          for taint byte $t_b$ in i do
4              Insert InstAddr($t_b$) into the ExecSeq[index][$t_b$];
5          end
6      end
7      Call Stack Analysis;
8      if call stack changes and a function returns then
9          ContextComparison(ExecSeq[index]);
           // compare the execution sequences and identify fields
10         delete ExecSeq[index];
11         index−−;
12     end
13     if call stack changes and a function starts then
14         index++;
15         ExecSeq[index] = new ExecSeq;
16     end
17 end
```

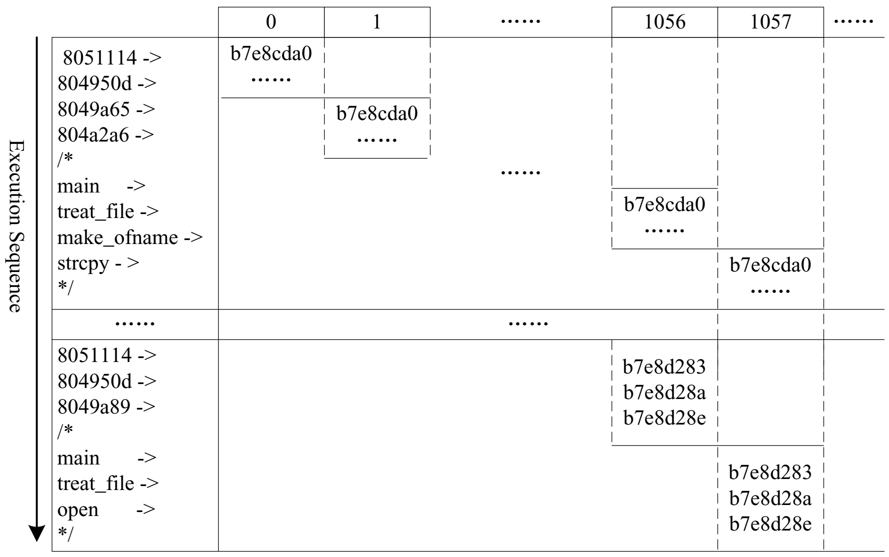


Fig. 4. The execution trace of the vulnerable gzip. The header refers to taint tags of input bytes, the left column refers to the call stacks, and the hex numbers refer to the address of instructions accessing the taint bytes.

a function returns, we compare the execution context for the bytes that are only processed in this function, and the offset interval could be identified. The algorithm is shown in Algorithm 1.

This scheme leads to little impact on the effectiveness of identified fields but a little impact on the detection of memory corruptions. First, it is rarely that several parts of one field are processed with different functions. Second, memory corruption attacks could only be detected when the call stack changes, which causes a little delay.

Whenever a field is identified, we record the current call stack as well as the instruction address accessing this field, and make them as the accessing location of this field. Note that the field could be accessed by several instructions, we only regard the current call stack and the beginning instruction address as the accessing locations.

We still take the `gzip` example for detailed illustration. Fig. 4 shows a segment of execution contexts. When the program calls `strcpy`, we create a new object of data structure to store the execution sequences for every taint byte. Among the instructions of `strcpy`, we record the tainted instructions, and insert the instructions into the execution sequences of taint bytes. When `strcpy` returns, we find that the taint bytes share the same execution context and then are grouped as one unit in `strcpy`. The field with the offset interval [0, 1199] (the length of the exploit is 1200) is identified. Another call stack is captured when program calls `open`. The bytes from offset 1056 to 1199 are accessed by the instructions of which the addresses are `b7e8d283`, `b7e8d28a`, and `b7e8d28e`. But the bytes from offset 0 to 1055 are not accessed in this function. As a result, a new field of which the offset interval is [1056, 1199] is identified.

For the two fields identified in this trace segment, [0, 1199] and [1056, 1199], the accessing locations identifying these two fields are shown in Fig. 5. The two fields are accessed by several instructions, we only record the beginning instruction addresses as the accessing locations.

offset interval	execution context
0-1199	8051114 -> 804950d -> 8049a65 -> 804a2a6 -> b7e8cda0 /* main -> treat_file -> make_ofname -> strcpy */
1056-1199	8051114 -> 804950d -> 8049a89 -> b7e8d283 /* main -> treat_file -> open */

Fig. 5. The accessing locations identifying the two fields

4.5 Structure Normalization

The identified fields are represented with the offset intervals. These offsets are specific to an input, because the values of some fields are user-defined and not every fields is length-fixed. For two inputs, the offset intervals are likely not identical and cannot directly to compare. We need to normalize the specific offsets with an abstract structure.

The tree structure is a well fit data structure to represent the input formats because the input could be a flatten or hierarchical structure [6,15]. In our approach, we also employ the tree to normalize the structures of specific inputs, where nodes refer to the fields and edges refer to the flatten or hierarchical relationships.

During online monitoring and detection, the tree is dynamically built and initialized with a root node. Whenever a new field is identified, we search its parent node of which the offset interval is the smallest yet covers the offset interval of the new field, and then insert a new node into the tree as a child node of its parent node. If no parent is found, we will make the root node as the parent node. At the same time, for a parent node has several child nodes, we order these child nodes with a increasing order of their offset interval.

5 Implementation and Evaluation

We implement FiGi on the platform of `Bitblaze` [24]. `TEMU`, the taint tracking component of `Bitblaze`, develops sophisticated data structures on the QEMU and could be very slow with an overhead at several hundreds [24].

Table 1. Vulnerable Programs

Programs	Published Date	Vulnerable Description	Detected
3CTfpdSvc-0.11	2006-05	stack overflow	√
AT-Tftp-1.9	2008-05	stack overflow	√
knftpd-1.0.0	2011-10	stack overflow	√
tftpd32-2.21	2010-09	format string	√
nginx-0.6.38	2010-08	heap overflow	√
wu-ftpd-2.6.0	2001-01	format string (non-control)	√
ghttpd-1.4.3	2002-10	stack overflow (non-control)	√
floatFTP	2011-09	stack overflow (ROP attack)	√
gzip-1.2.4	2002-06	stack overflow (DoS)	√

`Bitblaze` uses shadow memory and a data structure of taint record to represent the taint tags for every bytes. FiGi extends the data structure of such taint record to an array to store the multiple tags, and modifies the taint propagation for instructions with multiple source operands. Compared with the original `Bitblaze`, the performance of FiGi is impacted by a litter larger memory movement of taint records and the multiple taint propagation for instructions that have multiple tainted source operands. FiGi results in the similar overhead as `Bitblaze`.

The off-line training is a stand-alone program that is performed on the execution traces collected from FiGi. The online monitoring, context analysis, and deviation detection are implement as a plugin of `TEMU`. Both the off-line training and online monitoring are implemented in C++ with about 3000 lines of code.

5.1 Evaluation on Attacks

We select several vulnerable programs to evaluate the effectiveness of FiGi. The selected programs are shown in Table 1. The exploits of vulnerable programs cover memory corruptions on stacks, format strings, and heaps. The execution of such exploits could result in both control-hijacking and non-control data attacks. In addition, we also select an attack exploit based on the Return-Oriented Programming (ROP), which is a research focus during recent years, and a DoS attack of which the memory corruption could lead the program to crash but hard to exploit.

As shown in Table 1, FiGi could detect all these memory corruptions. In general, memory corruptions are the root-cause of control-hijacking, non-control data attacks, and many memory errors. The approach behind FiGi is to detect the misuse of user inputs by unintended instructions. Therefore, we claim that FiGi is transparent to the type of attacks, no matter the memory corruption is used for control-hijacking, non-control data attacks, or just deny of service.

5.2 A Case Study

We use an example to show a case study on how FiGi works to detect both the control-hijacking and non-control data attacks. There is a stack overflow vulnerability in the `log()` function in `ghttpd-1.4.3`. This vulnerability could be triggered when the `GET` package contains too many bytes. In [8], Chen et al. proposed a non-control data attack to overwrite a significant pointer and force the program to execute the "$\backslash bin \backslash sh$". We use both control-hijacking exploit and non-control exploit to compromise this program.

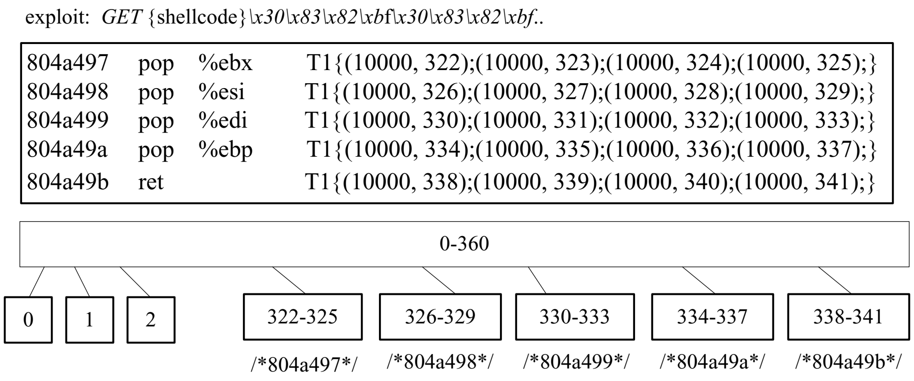
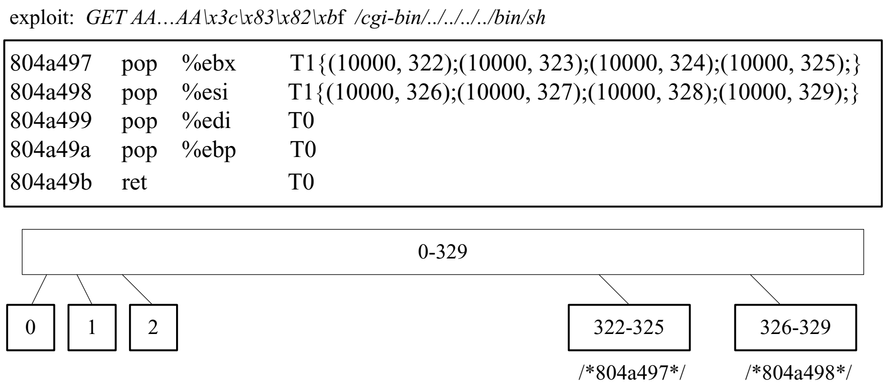

exploit: GET {shellcode}\x30\x83\x82\xbf\x30\x83\x82\xbf..
804a497 pop %ebx T1{(10000, 322);(10000, 323);(10000, 324);(10000, 325);}
804a498 pop %esi T1{(10000, 326);(10000, 327);(10000, 328);(10000, 329);}
804a499 pop %edi T1{(10000, 330);(10000, 331);(10000, 332);(10000, 333);}
804a49a pop %ebp T1{(10000, 334);(10000, 335);(10000, 336);(10000, 337);}
804a49b ret T1{(10000, 338);(10000, 339);(10000, 340);(10000, 341);}
0-360
0
1
2
322-325
326-329
330-333
334-337
338-341
/*804a497*/
/*804a498*/
/*804a499*/
/*804a49a*/
/*804a49b*/

Fig. 6. The detection of the control-hijacking attack on ghttpd. `T1` refers to tainted, and `T0` refers to non-tainted. The taint tags of individual byte are separated with ";". Each taint tag is represented as `(source, offset)`, where `source` refers to the input source and `offset` refers to the offsets of every bytes.

Let us first examine the control-hijacking attack shown in Figure 6. We observe that operands of these 5 instructions are tainted. By comparing execution contexts, 5 fields are identified ([322, 325], [326, 329], [330, 333], [334, 337], and [338, 341]), of which the accessing locations are `804a497`, `804a498`, `804a499`, `804a49a`, `804a49b`. After the normalization, we compare the pattern of the exploit with trained ones and find that the accessing locations of the 5 fields are never present in the benign executions.

During the non-control attack, as shown in Figure 7, the exploit only overwrites the value of `ebx` and `esi`. FiGi detects that 2 fields are identified and their accessing locations are not present in benign executions.

Note that the execution of `log()` function is much earlier before the `ghttpd` parsing the entire `GET` request package, thus the tree structure at this moment is simple. However, the tree structure of a benign input is much more complicate, and a similar result could be seen in [15]. There is no difference between this partial tree structure and a benign tree because this partial tree is a sub-tree of that. The deviation in this example is detected by matching the accessing locations.


exploit: *GET AA…AA\x3c\x83\x82\xbf /cgi-bin/../../../../bin/sh*

```
804a497   pop   %ebx    T1{(10000, 322);(10000, 323);(10000, 324);(10000, 325);}
804a498   pop   %esi    T1{(10000, 326);(10000, 327);(10000, 328);(10000, 329);}
804a499   pop   %edi    T0
804a49a   pop   %ebp    T0
804a49b   ret           T0
```

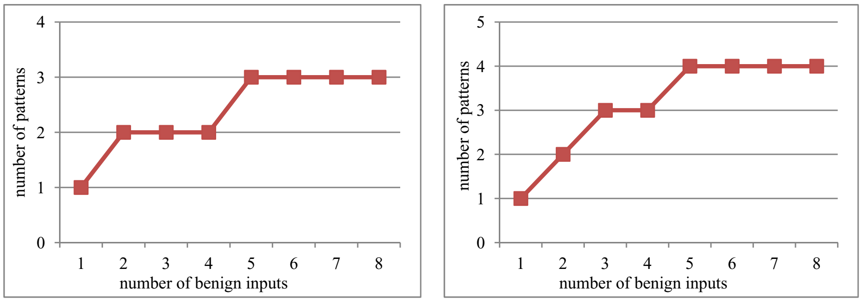
Fig. 7. The detection of the non-control data attack on ghttpd

5.3 False Negative and False Positive

In our experiments, we encounter no false negatives. It means that we missed no attacks among these control-hijacking, non-control data attacks, and memory corruption errors. Among these attacks, all the exploits try to overwrite some significant data structures, including the internal data structures (such as the return addresses) and significant variables. After the corruptions, some bytes would further be misused by unintended instructions, which may lead to the deviation of input structures or the deviation of execution context for a specific field of user input. These anomalies could be captured by the patterns of input processing. However, there is no guarantee that the false negatives will not occur.

The false positives is a big challenge in FiGi. The main reason to arise false positive is the coverage. That is, if the coverage is not sufficient, then we could miss some input processing patterns. For example, record sequences [11] are common in some user inputs such as images, videos and others. Generally, the processing of such sequences will be a loop. In such cases, the number of loop paths is large and hard to be full covered, and FiGi may generate false positives for benign executions covering untrained paths.

In addition, there is no close relationships between the input structure and the execution paths. Inputs with the same structure but different values of fields could also generate different patterns. This scenario is not rare because some significant input values could affect the program behaviors. We are interested in the training overhead of inputs with identical structures. In the two case studies, we only change the value of some fields but keep the structure unchanged, and then construct 8 benign inputs for `gzip` and `nginx`, respectively. The training overhead is shown in Fig. 8, and the numbers of trained patterns are 3 and 4, respectively.



(a) gzip (b) nginx

Fig. 8. The training overhead of the gzip and nginx

5.4 Comparison with Peer Techniques

We perform a theoretical comparison between FiGi and peer techniques on their abilities to detect attacks. Since FiGi works on the binary level, we just select peer techniques works on binaries (e.g., TaintCheek [21], PointerTainting [18], CFI [2], Clause2007 [9,12]) and transparent protection systems (e.g., StackGuard [10], Data Execution Protection (DEP for short) [1], ALSR [4]).

Table 2. Comparison with Peer Techniques

	Attacks								Performance	Errors
	S(C)	F(C)	H(C)	F(N)	S(N)	H(N)	ROP	DoS		
StackGuard [10]	√	•	•	×	×	×	•	×	≈ `1x`	FN
DEP [1]	√	√	√	×	×	×	×	×	≈ `1x`	FN
ALSR [4]	√	√	√	√	√	√	•	•	≈ `1x`	FN
TaintCheck [21]	√	√	√	×	×	×	√	×	`1.5x-30x`	FN
PointerTaint [18]	√	√	√	√	√	√	√	×	`1.5x-30x`	*FP/FN*
CFI [2]	√	√	√	×	×	×	√	×	`1x-2x`	FN
Clause2007 [9,12]	√	√	√	√	√	√	√	•	`1x-500x`	FN
FiGi	√	√	√	√	√	√	√	√	`100x-1000x`	FP

In Table 2, we list the control-hijacking and non-control data attacks which may be caused by stack overflow, format string, and heap overflow. For short representation, we use `S`, `F`, `H` to denote the stack overflow, format string vulnerability, and heap overflow, respectively. We use `C` and `N` to denote the control-hijacking and non-control data attacks, respectively. The symbol `S(C)` refers to the control-hijacking attacks caused by stack overflow. Other symbols could be similarly explained. In addition, we also select ROP and DoS memory corruptions as comparison features. ROP could be caused by stack overflow, format string, and heap overflow, and in Table 2, we just use ROP to represent attacks passing DEP.

In Table 2, we use $\surd$, $\times$, $\bullet$ to denote that the attack can be detected, cannot be detected, and uncertain. $\bullet$ means the protection mechanism may have variants, which could be the improvement of previous techniques, or a different implementation. Attacks could be detected by some of them, but may not be detected by other variants. For example, StackGuard [10] places a hard-to-predict canary before the return address on the stack, which could detect stack smashing but cannot detect control-hijacking attacks caused by format string and heap overflow. As an improvement, StackShield protects the stack by copying the return address to a "secure" location. Some ASLR [4] only randomize the load memory but keep the relative addresses un-randomized, than it cannot detect the memory corruptions between variables (e.g., `wu-tftp` and `gzip`). As an improvement, some fine-grained ASLR could also randomize the relative addresses (e.g., ASLR through binary transformation [4]), which could mitigate more attacks. As we discussed in Sec. 2, Clause2007 [9,12] may lose its ability for non-control memory corruptions if the symbol table is unavailable.

FiGi could detect all these memory corruptions based the consistent and reliable foundation that memory corruption exploits violate the program semantics, leading some input bytes being misused by unintended instructions, and the anomaly could be captured through patterns on the input processing. We encounter no false negatives (FN for short in Table 2), but other peer techniques could result in false negatives more or less. It means that FiGi misses the least number of attacks. However, FiGi could result in false positives (FP for short in Table 2).

From Table 2, we could also observe that performance is the shortage of FiGi. The transparent protections are very fast. The prototype of TaintCheck [21] could result in the overhead about `30x`. Recent advances on dynamic taint analysis could reduce the overhead to `1.5x` [5]. There is no quantified performance in PointerTainting [18], but it should has the similar overhead with dynamic taint analysis. The overhead of CFI is about `1x-2x`. The Clause2007 [9,12] prototype based on software emulation could cause the overhead of `100x-500x`, and the improved overhead with hardware-assistant is about `1x-2x`. We build FiGi on TEMU [24], of which the overhead is more than several hundreds. To improve the performance, several optimizations could be used, such as simplifying taint tracking instructions [22], designing novel memory layout to reduce the overhead caused by taint propagation [5], faster emulator [5], and hardware assistant [12]. Although these optimizations may not be directly used, we could learn from these and design similar schemes.

6 Limitations and Discussions

In this paper, the fields mainly include semantic independent units, and we do not recognize the higher level data types. For the complex and higher level data types such as structs and unions, every item in the struct or union is treated as one independent field, if each item is processed as one unit among the execution.

Comparing the execution contexts could lead to over-fine granularity [15]. For example, `strcmp` could return after accessing the first few bytes. In such cases, the first few bytes will be regarded as one field and FiGi may divide the string into several parts. Fortunately, such over-fine granularity shall occur in all executions, both benign and abnormal ones. Therefore, this limitation bring little noise to the deviation detection.

7 Conclusion

In this paper, we propose a novel approach FiGi to detect memory corruptions at the binary level. FiGi identifies individual parts in an input and learns the pattern in which they are processed. We implement a prototype of FiGi and demonstrate its success in detecting a number of memory corruption attacks in the wild. The experiments shown that FiGi is effective to detect memory corruptions.

Acknowledgment. This work is partially supported by the National Natural Science Foundation of China (no. 60970114), the Doctoral Priority Development Projects granted by the Chinese Ministry of Education (no. 20110141130006), the Research Award for Excellent Doctoral Student granted by Chinese Ministry of Education, and the Northeast Asia Grant Program by SafeNet.

References

1. Data execution protection, `http://technet.microsoft.com/en-us/library/cc738483(WS.10).aspx`
2. Abadi, M., Budiu, M., Erlingsson, U., Ligatti, J.: Control-flow integrity principles, implementations, and applications. ACM Transactions on Information and System Security 13(1), 1–40 (2009)
3. Akritidis, P., Cadar, C., Raiciu, C., Costa, M., Castro, M.: Preventing Memory Error Exploits with WIT. In: 2008 IEEE Symposium on Security and Privacy, pp. 263–277 (2008)
4. Bhatkar, S., DuVarney, D.: Address obfuscation: An efficient approach to combat a broad range of memory error exploits. In: Proceedings of USENIX Security (2003)
5. Bosman, E., Slowinska, A., Bos, H.: Minemu: The World's Fastest Taint Tracker. In: Sommer, R., Balzarotti, D., Maier, G. (eds.) RAID 2011. LNCS, vol. 6961, pp. 1–20. Springer, Heidelberg (2011)
6. Caballero, J., Yin, H., Liang, Z., Song, D.: Polyglot: Automatic extraction of protocol message format using dynamic binary analysis. In: Proceedings of CCS, pp. 317–329 (2007)
7. Castro, M., Costa, M., Harris, T.: Securing software by enforcing data-flow integrity. In: Proceedings of OSDI, pp. 147–160 (2006)
8. Chen, S., Xu, J., Sezer, E.C., Gauriar, P., Iyer, R.K.: Non-control-hijacking attacks are realistic threats. In: Proceedings of USENIX Security (2005)
9. Clause, J., Doudalis, I., Orso, A., Prvulovic, M.: Effective memory protection using dynamic tainting. In: Proceedings of ASE, pp. 284–292 (2007)
10. Cowan, C., Pu, C., Maier, D., Hintony, H., Walpole, J., Bakke, P., Beattie, S., Grier, A., Wagle, P., Zhang, Q.: StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In: Proceedings of USENIX Security (1998)

11. Cui, W., Peinado, M., Chen, K., Wang, H., Irun-Briz, L.: Tupni: Automatic reverse engineering of input formats. In: Proceedings of CCS, pp. 391–402 (2008)
12. Doudalis, I., Clause, J., Venkataramani, G., Prvulovic, M., Orso, A.: Effective and Efficient Memory Protection Using Dynamic Tainting. IEEE Transactions on Computers 61(1), 87–100 (2012)
13. Godefroid, P., Kiezun, A., Levin, M.Y.: Grammar-based whitebox fuzzing. In: Proceedings of PLDI, vol. 43, pp. 206–215 (2008)
14. Jim, T., Morrisett, G., Grossman, D., Hicks, M., Cheney, J., Wang, Y.: Cyclone: A safe dialect of C. In: Proceedings of USENIX ATC, pp. 275–288 (2002)
15. Lin, Z., Jiang, X., Xu, D., Zhang, X.: Automatic protocol format reverse engineering through context-aware monitored execution. In: Proceedings of NDSS (2008)
16. Lin, Z., Zhang, X.: Reverse Engineering Input Syntactic Structure from Program Execution and Its Applications. IEEE Transactions on Software Engineering 36(5), 688–703 (2010)
17. Livshits, V.B., Lam, M.S.: Tracking pointers with path and context sensitivity for bug detection in C programs. In: Proceedings of FSE, vol. 28, pp. 317–326 (2003)
18. Nakka, N., Kalbarczyk, Z., Iyer, R.: Defeating Memory Corruption Attacks via Pointer Taintedness Detection. In: Proceedings of DSN, pp. 378–387 (2005)
19. National Institute of Standards and Technology: National vulnerability database statistics, http://web.nvd.nist.gov/view/vuln/statistics
20. Necula, G.C., McPeak, S., Weimer, W.: CCured: type-safe retrofitting of legacy code. In: Proceedings of POPL, pp. 128–139 (2002)
21. Newsome, J., Song, D.: Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: Proceedings of NDSS (2005)
22. Qin, F., Wang, C., Li, Z., Kim, H.S., Zhou, Y., Wu, Y.: LIFT: A Low-Overhead Practical Information Flow Tracking System for Detecting Security Attacks. In: Proceedings of Micro, pp. 135–148 (2006)
23. Slowinska, A., Bos, H.: Pointless tainting? Evaluating the practicality of pointer tainting. In: Proceedings of EuroSys, pp. 61–74 (2009)
24. Song, D., Brumley, D., Yin, H., Caballero, J., Jager, I., Kang, M.G., Liang, Z., Newsome, J., Poosankam, P., Saxena, P.: BitBlaze: A New Approach to Computer Security via Binary Analysis. In: Sekar, R., Pujari, A.K. (eds.) ICISS 2008. LNCS, vol. 5352, pp. 1–25. Springer, Heidelberg (2008)
25. Tsai, T., Singh, N.: Libsafe: transparent system-wide protection against buffer overflow attacks. In: Proceedings of DSN, pp. 541–550 (2002)
26. Younan, Y., Philippaerts, P., Cavallaro, L., Sekar, R., Piessens, F., Joosen, W.: PAriCheck: an efficient pointer arithmetic checker for C programs. In: Proceedings of AsiaCCS, pp. 145–156 (2010)

Dynamic Anomaly Detection for More Trustworthy Outsourced Computation

Sami Alsouri, Jan Sinschek, Andreas Sewe, Eric Bodden, Mira Mezini, and Stefan Katzenbeisser

Technische Universität Darmstadt
Center for Advanced Security Research Darmstadt - CASED
Mornewegstraße 32, 64293 Darmstadt, Germany
{sami.alsouri,jan.sinschek,andreas.sewe,eric.bodden}@cased.de,
mezini@st.informatik.tu-darmstadt.de,
katzenbeisser@seceng.informatik.tu-darmstadt.de

Abstract. A hybrid cloud combines a trusted private cloud with a public cloud owned by an untrusted cloud provider. This is problematic: When a hybrid cloud shifts computation from its private to its public part, it must trust the public part to execute the computation as intended. We show how public-cloud providers can use dynamic anomaly detection to increase their clients' trust in outsourced computations. The client first defines the computation's reference behavior by running an automated dynamic analysis in the private cloud. The cloud provider then generates an application profile when executing the outsourced computation for its client, persisted in tamper-proof storage. When in doubt, the client checks the profile against the recorded reference behavior. False positives are identified by re-executing the dubious computation in the trusted private cloud, and are used to re-fine the description of the reference behavior. The approach is fully automated. Using 3,000 harmless and 118 malicious inputs to different Java applications, we show that our approach is effective. In particular, different characterizations of behavior can yield anything from low numbers of false positives to low numbers of false negatives, effectively trading trustworthiness for computation cost in the private cloud.

Keywords: Cloud security, dependability, dynamic analysis, anomaly detection, hybrid clouds.

1 Introduction

Cloud computing allows companies to outsource part of their computations to server farms, usually owned by a cloud provider. It promises many benefits, such as reducing infrastructure investments, the ability to quickly adapt its compute power according to the demands (the so-called "elastic cloud"), or the adoption of a pay-as-you-go billing model [22].

But cloud computing comes at a risk. While a company controls its private computer servers, it has limited control over resources rented in the cloud.

This motivates the so-called "hybrid cloud" scenario, in which a company owns a private cloud of trusted compute servers, while at the same time this private cloud shares data with a public cloud service owned by a public cloud provider, executing another set of computations. Which computations are performed in the private and which ones in the public cloud depends on the company's preferences and policies.

All instances of hybrid clouds share the common problem that, when shifting computation from their private to their public parts, they must trust the public part to execute the computation as intended. But what justifies that trust?

In this work, we introduce *behavior compliance control* , in which a cloud provider uses methods from dynamic anomaly detection to provide clients trustworthy evidence about the absence of "abnormal" executions caused by incorrect server configurations, version mismatches, hardware glitches or malicious attacks by third parties [20]. Providing such evidence is very important in scenarios where faults or attacks occur through invalid program inputs such as incorrect or compromised configuration files.

Our approach starts with a learning phase in which the client uses an automated tool to learn the behavior of an application by running it in the trusted private cloud on a collection of representative inputs. This process, conducted before outsourcing the application, results in a so-called application model. The model is considered to characterize the application's intended behavior. In this work we study and compare models at different levels of granularity.

After the application has been outsourced into the public cloud, the outsourced application uses runtime monitoring techniques to log critical runtime information into a securely sealed storage, thus yielding trusted evidence on the application's remote behavior. Next, the client verifies if the observed log information, according to this evidence, complies with the application model learned in the learning phase. If the run is found to be compliant, the outsourced computation is assumed to have executed correctly. If the run is not compliant this can be either due to an actual anomaly in the public cloud, or due to a false positive caused by an imprecise application model. We restrict ourselves to deterministic programs, which the client can re-execute in the private cloud to tell both cases apart. If this trusted re-execution yields the same result then the client has identified a false positive, and can use this false positive to refine the application model. If not, then the client has found an anomaly, i.e., an actual piece of evidence of a faulty or maliciously influenced computation in the public cloud.

In this work, we present the first work leveraging dynamic anomaly detection for the scenario of hybrid cloud computing. In particular, we present the following contributions: We present an abstract architectural framework for behavior compliance control. The framework is defined in terms of its abstract security requirements, and hence independent of any concrete implementation. In addition, however, we present and make publicly available[1] a concrete instantiation

[1] Our implementation is available, in source, along with all our raw data and scripts to reproduce our empirical results, at `http://seceng.de/research/projects/bcc`

of this framework for the Java platform. In this instantiation, we implement a sealed storage using Trusted Computing technologies.

Another main contribution is an empirical evaluation showing how the efficacy of our approach depends on the choice of application model. We evaluate three kinds of models that abstract from an application's dynamic behavior with increasing granularity, by recording (1) the set of called methods, (2) a dynamic call graph, or (3) a dynamic calling context tree. We used our Java-based implementation to produce and evaluate application models for three different open-source applications, applied to 3,000 publicly available documents we believe to be harmless and 118 known malicious documents containing web exploits. Our results show that our approach is effective. In particular, different choices of models can yield anything from low numbers of false positives to low numbers of false negatives. This gives clients a large degree of freedom in trading increased trustworthiness for increased computation cost in the private cloud.

The remainder of this paper is structured as follows. In Section 2, we describe our three choices of behavioral abstractions. Section 3 defines our architectural framework for behavior compliance control, while Section 4 describes our concrete instantiation for Java. In Section 5, we discuss our empirical evaluation, assessing the usefulness of our three abstractions for the purpose of behavior compliance control, as well as the performance of our approach. We discuss related work in Section 6 and our conclusions in Section 7.

2 Characterizing Behavior

Behavior compliance control builds on techniques from dynamic anomaly detection [8,9,11,13,18], a methodology that attempts to detect anomalous executions by comparing certain execution characteristics with those known to be characteristic for correct and "compliant" executions. Our technique significantly extends traditional anomaly detection by a means to conduct the detection process in a distributed but nevertheless trustworthy fashion. Yet, an important design decision that both previous approaches as well as ours have to make is how to best characterize an application's runtime behavior.

Since all previous approaches describe behavior at one level of granularity and have therefore some disadvantages, we decided to not restrict ourselves to a single mind set: Instead of fixing one given classification of behavior upfront, we decided to implement three white-box abstractions on different levels of abstraction, and to compare their relative usefulness for the behavior compliance control of outsourced applications. Clients can then choose which abstraction best fits their needs.

Behavior Models. We regard function calls as a main ingredient for characterizing behavior.[2] We have evaluated three approximations of behavior by tracing which

[2] We use the term "function" instead of "method" because our approach is not bound to Java. Our functions are not "functional" in the strict sense: They may have side-effects.

functions a program calls during its execution, and in which contexts. Each approximation thereby induces a different kind of application model for our behavior compliance control approach. We distinguish models according to the amount of information they contain (from least to most):

– **Functions:** A set of functions F the application called during the execution.
– **Call graph:** A call graph, with nodes representing functions, and an edge from f to f' if f calls f' at least once during the execution.
– **Calling context tree:** A calling context tree [1], with the root node representing the program's entry point and a node f' as child of node f if f calls f' in the same context at least once during its execution.

To illustrate these abstractions, consider the example program in Figure 1. Figure 2a shows the "Functions" representation of the example program. Herein, the model just consists of the set of all functions called during the program's execution. Figure 2b, on the other hand, shows the program's dynamic call graph. Note that in a call graph, every function, such as `bar`, is represented by exactly one node, no matter in how many different contexts the function is invoked. Figure 2c shows the program's calling context tree. In this representation, calling contexts are kept separate: Because `bar` is called by two different functions, once by `main` and once by `foo`, it appears twice in the tree, just under the appropriate contexts.

We chose these three different characterizations of behavior carefully, so that one can construct a model of higher abstraction from a model of lower abstraction. This allows us to compare the models directly to each other, based on the very same data set. The fact that the three different abstractions form such a total order allows us to evaluate different characterizations of behavior at opposite ends of the granularity spectrum: The "Functions" abstraction is quite coarse-grained but can be computed very efficiently. Yet, by its nature it may have the tendency to yield false negatives, i.e., to miss anomalies. Hence, the amount of trustworthiness that this abstraction provides is relatively low. The calling context trees at the other end of the spectrum are very fine-grained. Their computation consumes more time, and by their nature they tend to cause a relatively large number of false positives, increasing the necessary computation cost in the private cloud. But nevertheless, this may still be a price worth paying for the additional trustworthiness they provide. In Section 5, we present an extensive evaluation demonstrating the absolute and relative utility of those abstractions for the task of behavior compliance control. We formalize our abstractions in the appendix.

Model Generation in the Learning Phase. For the purpose of behavior compliance control, models should be sensitive to malicious inputs, where faults or attacks occur through them, such as incorrect or compromised configuration files.

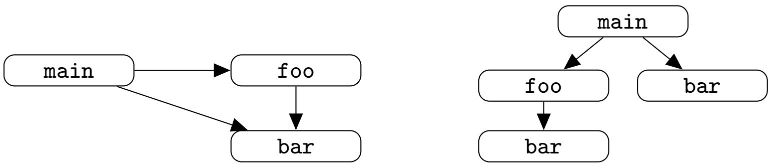
We therefore opt for a dynamic approach that collects an application model as a union of a set of runtime execution profiles. Within the trusted private cloud,

```
public static void main(String args[]) {
    foo();
    bar();
}

static void foo() { bar(); }

static void bar() { }
```

Fig. 1. Example program

{`main`, `foo`, `bar`}

(a) Functions



(b) Call graph **(c)** Calling context tree

Fig. 2. Three abstractions of the example program

the client collects an execution profile for every test run. The application's final model for this training data is then defined as the union of all those individual profiles. In the case of Functions we use simple set union, while in the case of call graphs or calling context trees we define the union in the natural way, by computing the union over the graph's, respectively tree's, node and edge sets. We call the resulting profile the application's *model*. Since the union operation is associative, one can compute the model in a step-wise and iterative way, i.e., after each individual profile is collected, or instead compute the union once over all collected individual profiles.

This property is key to our approach: When a model appears too restrictive, it can easily be expanded by joining the application's current model with new execution profiles. Clients can make use of this property after having identified a false positive. The model is extended accordingly to avoid the same false positive in the future. This process can be fully automated.

3 Platform Architecture

In this section, we present our abstract platform architecture for behavior compliance control. The architecture assumes the presence of a trusted logger that is sufficiently tamper-resistant, as well as securely sealed storage on the host that performs the computation. Assuring the integrity of these components is a problem complementary to the one of behavior compliance control and may be achieved through several means. In Section 4 we will describe a concrete instantiation of the generic architecture that fulfills these requirements, including concrete mechanisms for establishing the integrity of the collected profiles.

As described in Section 2, the client first computes an application model by running the software in a trusted environment, the private cloud. Subsequently, the client outsources the application and executes it in the untrusted public cloud. After execution, the public cloud provides the client with evidence about the application's behavior. The client finally verifies the evidence locally to decide on its trustworthiness, and, if required, refines the application model. In the following, we detail the individual phases of this procedure.

I: Learning phase. As described in Section 2, the client generates an application model m, characterizing the behavior of the application, by running the application in the client's trusted private cloud collecting the generated profiles. Afterwards, the application is outsourced to the public part of the cloud.

II: Runtime phase. Figure 3 shows the abstract platform architecture of the hosting platform. We assume the presence of trusted system measurement components, which assure the load-time integrity of the loaded applications and the trusted logger. The load-time integrity of the public cloud platform can be verified by the client before outsourcing takes place. Those mechanisms in fact assure the client that, at load time, exactly those components are brought to execution that the client intended to execute.

The trusted logger logs the events coming from the application itself (i.e., in case of using instrumented code) or from the middleware (e.g., the Java Virtual Machine, or a business process engine), on which this application runs. As mentioned previously, a core task of the public cloud is to generate logs about the execution of the outsourced application in a trustworthy way. As the generated logs are security critical, secure storage is required.

III: Compliance verification phase. Once the client has obtained the log and verified its integrity, he compares the log against the application model collected in the learning phase. We write $l \models m$ if the log l corresponds to the model m and $l \not\models m$ otherwise. Whenever $l \not\models m$, this means that the log diverged from the model, indicating a dubious execution. Such a divergence could be the effect of an execution anomaly but could also just be a false positive, due to an overfitting application model.

To tell apart a false positive from an actual anomaly, the client would then re-execute the application in the private cloud and record the resulting log l'. We write $l \equiv l'$ if the log l is equivalent to the log l' and $l \not\equiv l'$ otherwise. Whenever $l \equiv l'$, the execution in the public cloud is considered correct and trustworthy, and the divergence was a false positive. In this case, the client expands m by including l. By doing so, m can be continuously improved to decrease the overall false positives rate. (Note that there is a trade-off: By including l in m the model also becomes more permissive, which may yield more false negatives as well.) If otherwise $l \not\equiv l'$, the execution is considered untrustworthy. Concrete implementations of the "$\models$" and "$\equiv$" operators depend on the kind of profile being used. We discuss our concrete instantiation in Section 4.

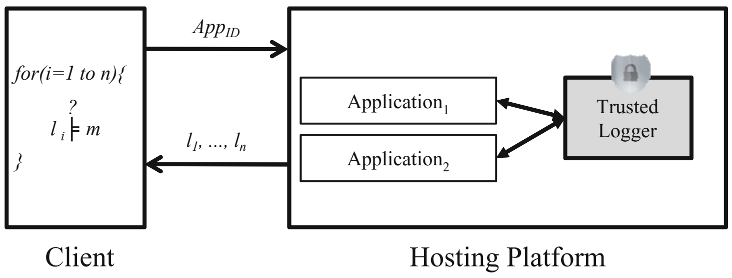


Fig. 3. Our architecture for behavior compliance control

Note that, in contrast to intrusion detection, where a very low false positive rate is imperative, our approach can tolerate higher rates. Essentially, the false-positive rate determines how much outsourced computation has to be re-done in the private cloud. Thus, in the hybrid cloud environment, a false-positive rate in the order of a few percent may well be acceptable, as still the bulk of the computation is performed in the public cloud.

Threat model and limitations. Our approach hinders attacks by the cloud provider but cannot fully prevent them. This is because we demand the existence of a trusted logger, whose integrity can only be assured through specialized hardware. The cloud provider has access to this hardware and thus may have the means to compromise it. Assuming, however, that the trusted logger can indeed be trusted, our architecture guarantees that detected anomalies can be communicated to the client in a tamper-proof way. Our approach can effectively identify execution anomalies caused by malicious program inputs of any kind, of by faults or misconfigurations in the execution environment. The former is particularly useful to identify attacks on such systems in the public cloud that have a public interface, for instance web servers or document servers.

Our approach is passive, i.e., anomalous behavior is detected only after the fact. As such, our approach cannot prevent anomalous behavior from happening. Instead it allows the client to identify the anomaly, and thereby to re-execute the original computation (and thus circumvent the anomaly) in the trusted private cloud, and to take other appropriate measures such as legal actions in case the anomaly was caused by a malicious intruder. While approaches to active compliance control are possible (e.g., by inserting a runtime monitor that checks model compliance just in time), such an approach would greatly suffer from any false positives: When a monitor detects an anomaly at runtime, it must decide whether the anomaly is real or a false positive at that point in time. In most cases, this is impossible. In any case, such active compliance control would not be able to provide the flexible trust/cost trade-off that we see as one of the greatest benefits of our approach.

Last but not least, it should be noted that all security-related approaches to anomaly detection, including our own one, are susceptible to mimicry attacks [30, 36], in which an attacker tries to execute behavior that is malicious, but nevertheless mimics legal behavior in such a way that the malicious behavior remains undetected. This problem can be mitigated somewhat by keeping the application model undisclosed, but to the best of our knowledge no way to absolutely avert mimicry attacks is known to date.

4 Platform Instantiation

We next discuss our instantiations of the generic architecture described in Section 3 to the Java language and platform. As a way to provide secure storage, we base our instantiation on concepts from Trusted Computing. Our full implementation of this instantiation is available online, in source, on our project website.

4.1 Adaption to Java

To generate execution profiles, we use JP2, an open source calling-context-tree profiler for Java [26,27]. This light-weight profiler consists of a small Java agent, which instruments the profiled application at load time, and an accompanying tool to instrument the Java runtime library ahead-of-time. This combination enables us to generate execution profiles which cover not only the application but also the Java runtime library itself. Moreover, JP2's profiles cover not only methods that have a bytecode representation but also method calls made in either direction across the bytecode-native code boundary. The following details are specific to a Java-based setting:

– **Virtual machine-based execution:** The Java platform allows for easy load-time transformation of code. Hence, to introduce a runtime monitor, a client does not need to instrument his application in house. Instead, the application can be transformed remotely, by a custom class loader [23] or transformation agent. Such instrumentation is performed on the level of bytecode and requires no access to source code. JP2 does exactly this.
– **Generated code:** The same class-loader mechanism that makes it easy to introduce a runtime monitor at load time also makes it possible to generate classes at runtime. Such classes frequently bear a randomized name, and that name must be canonicalized to ensure that the same method, up to renaming, can be reliably identified across program runs. To that end, we integrated the hashing facility from TamiFlex [5] with the calling-context-tree profiler described next.
– **Recursion:** When using the Calling Context Tree abstraction, recursive calls can cause the profile to grow very large. One way to address this would be to "fold" those sub-trees in the CCT that exhibit a recursive structure. The generated profiles would hereby be bounded. However, what exactly counts as recursion in a language with dynamic dispatch is not obvious: Do only calls to the same target method count or also calls to a different target method of the same call site? Calls of the latter kind are frequent, e.g., when operating on objects structured using the Composite pattern [10]. Moreover, mutual recursion or, more generally, larger cycles of calls could be considered

recursive as well and maybe thus subject to folding. For the purpose of this paper we restrict the discussion to the straight-forward calling context tree abstraction produced by JP2 and do not fold recursive calls; thus, the tree structure mirrors the entire computation.

In our current implementation, we always collect full calling context trees, even if we are just interested in call graphs or function sets. Call graphs are computed from a CCT by merging nodes with the same name, and method sets are computed by a simple exhaustive search through the call graph. This methodology is a limitation of our prototype. For efficiency, a realistic implementation would record only the information required for the chosen behavior characterization. We implement the "$\models$" operator from Section 3 by simply checking whether the calling context tree, call graph or function set collected on the server is a sub-tree, sub-graph or sub-set of the respective application model. For the "$\equiv$" we define that $l \equiv l'$ if the respective trees or graphs are isomorphic, or in the case of function sets if they are equal. We store calling context trees and call graphs in a normalized fashion that allows us to decide $l \equiv l'$ in time linear in the size of the operands.

4.2 Integrity of Trusted Components and Runtime-Secure Storage

To safeguard not only against anomalies caused by accidental misconfigurations or hardware glitches but also against (certain classes of) malicious attacks, it is necessary to store the runtime information collected in a trustworthy manner.

Our particular choice to instantiate the integrity measurement components and the secure storage relies on the concepts of Trusted Computing. To assure load-time integrity of the trusted logger in the public cloud, we first build a chain of trust, starting from the cloud server's hardware up to the trusted logger itself. For this purpose, our hardware was equipped with a TPM chip. Trusted boot is assured using the Grand Unified Bootloader (GRUB) version 0.97 together with TrustedGrub [35] version 1.1.5. We used the attestation framework IMA (Integrity Measurement Architecture [25]) to allow the client to verify the load-time integrity of the behavior measurement component and the secure storage (effectively comparing cryptographic hashes of the binaries). Clients would typically use the integrity reporting facilities of those components before the outsourcing of computations takes place.

To provide a runtime-secure storage, our logging facilities record a hash chain of all logged events in one fixed Platform Configuration Register (PCR) of the TPM chip. For each logged event, the register's current hash value is replaced by a hash over this current value and the event's own payload data. A client can then validate the integrity of the log by re-performing the same hash operation on the log and comparing the resulting hash values. If they differ, the log has been tampered with, and the computation should be re-performed in the private cloud.

In the general context of outsourced applications, the use of a single hardware TPM is insufficient: Many applications execute in the same remote host, and each can be executed many times. Data measured for different application must be stored separately. We hence use the concept of virtual TPMs (vTPMs), which allows us to assign a (unique) virtual TPM instance to each outsourced process [28]. All vTPMs are managed by a vTPM manager, which provides an interface to create and access vTPM instances; the vTPM manager is notified whenever an application instance is started. We implemented a vTPM manager in Java as a proxy to create and manage vTPM instances. The vTPM instances themselves are implemented using the TPM emulator proposed by Strasser and Stamer [29]. To communicate with vTPMs, we use the tpm4java library [33], which facilitates using the cryptographic functionalities of vTPMs in our Java applications. In detail, one chooses a particular vPCR i to hold a hash chain of all recorded events. Whenever a new log entry is generated, the vPCR i is extended by hashing the log entry using SHA-1 and running the `TPM_Extend` command of the corresponding vTPM instance as described in the TPM specification [34]. The log entry itself is stored in external (untrusted) storage. Thus, after the outsourced application terminates, the vPCR register i of the vTPM associated to the application contains a (securely stored) hash chain of all recorded events; further, the log l of all events is available on storage.

Subsequently, remote attestation is performed to securely transfer the log, which is signed by the vTPM, to the client. After verifying the log's integrity, the client verifies the compliance of each single log entry (i.e., each call edge) with the application model as described in Sections 2 and 3.

5 Evaluation

In this section we evaluate our three behavior abstractions from Section 2, function sets, call graphs, and calling-context trees, with respect to following four research questions:

RQ1 (Feasibility): In the learning phase, do the collected profiles converge to a stable model of legal inputs with low false-positive rates?
RQ2 (Effectiveness): To what extent is the application model able to discriminate between legal and illegal inputs?
RQ3 (Scalability): Is the profile size independent of the application's runtime?
RQ4 (Efficiency): Can our approach be implemented efficiently enough to induce a sufficiently low runtime overhead?

5.1 General Experimental Setup

One restriction of our approach is that, to produce representative models, it requires a representative set of program inputs. This restricted us in our choice of evaluation subjects; we had to opt for applications for we would be able to obtain large sets of abnormal/malicious as well as legal/harmless inputs. We chose the following subjects:

1. *Apache pdfbox:* A PDF manipulation framework [2].

2. *POI-HSLF:* A Java API to extract data from PowerPoint documents [3].
3. *POI-HWPF:* A Java API to extract data from Word documents [3].

All applications operate on popular file types (Adobe PDF, Microsoft PowerPoint `.ppt`, and Microsoft Word `.doc`), all of which can be obtained in large numbers from the web. Moreover, all three file types are well-known attack vectors. For the PDF file type there further exist repositories of malicious inputs, which serve us to simulate possible manipulations by the cloud provider (details below).

5.2 RQ1: Feasibility

For behavior compliance control to be feasible, it must be possible to automatically generate a useful application model from only a number of representative inputs small enough not to be prohibitive. Moreover, the generated application models must yield false positive rates low enough for the approach to pay off. Remember that any false positive induces increased computation cost in the private cloud.

For our evaluation, we used the top 1,000 results of a Google search for `filetype:pdf`, `filetype:ppt`, and `filetype:doc`, respectively. The resulting corpus of inputs allowed us to generate application models from various numbers of input documents. We believe those 3,000 documents to be harmless, legal documents.[3] Therefore, if a model classified any run as abnormal that was induced by one of those inputs, we count this classification as a false positive.

To generate the application models, we first used the JP2 profiler (cf. Section 4.1) to obtain a calling context tree for each of the applications and inputs. From the resulting CCTs we then derived both dynamic call-graph and function-set profiles. This ensures that, for a given input document, all three abstractions of the application's behavior are consistent.

We then used ten-fold cross-validation [21] to determine the false-positive rate that can be expected of the collected models. For each of the three file types, the 1,000 profiles were first divided into ten subsets of 100 profiles each. Then, each profile from one of the subsets was checked for compliance with application models derived from an increasing number of (randomly chosen) profiles in the other nine subsets, up to all 900 profiles in the end. Every compliance check yields either the answer "compliant" or an anomaly warning. Since we consider our training set to only contain compliant input documents, we consider all warnings to be false positives.

Figure 4 shows the resulting false positive rates, averaged over the 10 subsets, for various training set sizes. Because we used ten-fold cross-validation, at most 900 out of the 1000 available inputs were used for model generation. As Figure 4 shows, for both the Function and Call Graph abstractions it suffices

[3] This is because Google has put in place filters to remove invalid or potentially malicious documents from its search index. In fact we tried to find malicious documents using Google but failed.

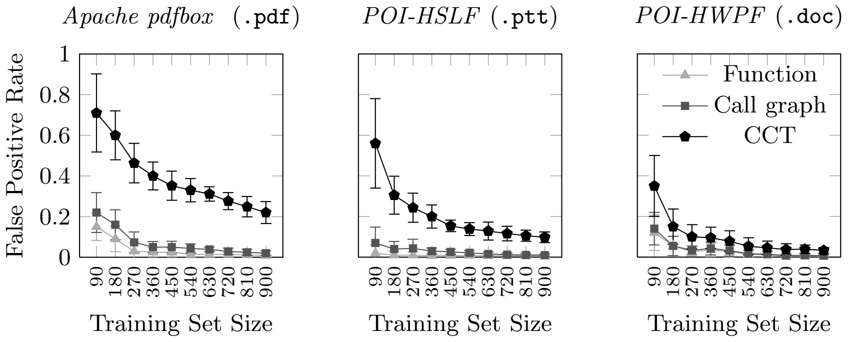


Fig. 4. False positive rate for differently-sized training sets (arithmetic mean ± standard deviation of 10 training sets each)

to use only a few hundred inputs for model generation to obtain a model with a false-positive rates below 5 %. Using the calling-context-tree (CCT) abstraction, however, requires a larger number of inputs to achieve low false-positive rates. Even when using 900 inputs to generate the application model, an average of about 22 %, 10 %, and 3 %, respectively, of the remaining 100 profiles are deemed non-compliant. We also observe that at least for the Calling Context Tree abstraction the false-positive rates very much depend on the program under evaluation.

5.3 RQ2: Effectiveness

To increase trustworthiness, behavior compliance control must be able to detect abnormal execution behavior. For the purpose of our evaluation we consider an execution to be abnormal if it executes on an abnormal program input. In reality, there could be other sources of abnormality such as glitches in the hardware or execution environment. We obtained abnormal inputs from two distinct sources: from dedicated repositories of malicious inputs for the file types in question and from applying fuzzing techniques to legal inputs.

To simulate a targeted attack by a third party, we have used a set of 118 PDFs that have previously been used in exploits.[4] For this experiment, we used application models computed by including all 1,000 profiles for PDF file type. As Table 1 shows, all abnormal executions were classified correctly when using the Calling Context Tree abstraction. When using the more coarse-grained Call Graph and Function abstractions, however, only 34 % respectively 11 % of inputs were classified correctly. We therefore conclude that it is essential to use information-rich profiles to detect targeted attacks reliably. This is the main trade-off at the heart of this paper: Increased trust requires an increase investment to counter-balance the increased rate of false positives caused by such information-rich profiles.

[4] Test data taken from `http://contagiodump.blogspot.com/2010/08/malicious-documents-archive-for.html` (Collection 3).

Table 1. Percentage of inputs (exploits or fuzzed) detected as illegal

	Exploits	Fuzzed		
	Apache pdfbox (`.pdf`)		*POI-HSLF* (`.ppt`)	*POI-HWPF* (`.doc`)
Functions	11 %	83 %	100 %	100 %
Call graphs	34 %	89 %	100 %	100 %
CCTs	100 %	97 %	100 %	100 %

As we were unable to obtain a similarly large number of malicious PowerPoint and Word documents to simulate a targeted attack, we commenced on a best-effort basis and resorted to fuzzing techniques to simulate an untargeted attack or a problem caused by a faulty data transmission. For each file type, we randomly picked 100 documents from of our corpus of legal documents and applied simple fuzzing techniques to them.[5] This process yields 100 documents each which we define to be abnormal inputs. For each of these inputs we then ran the corresponding application and compared its behavior, abstracted as Functions, Call Graph, or Calling Context Tree, with the application model of legal inputs used before.

Table 1 shows the percentage of fuzzed inputs that were successfully detected as illegal. As these results show, false negatives created by this simple fuzzing algorithm are easy to recognize. It follows that abnormal program runs induced by inputs corrupted in this manner will most likely be detected using behavior compliance control; the abstraction chosen (Functions, Call Graph, Calling Context Tree) has little influence on the detection rate. Those observations hold for the particular fuzzing approach we consider. More targeted fuzzing approaches, taking advantage of the input document's internal structure, may be harder to recognize, but from a security perspective would probably also be less capable of exploiting a vulnerability in the outsourced application.

5.4 RQ3: Scalability

For behavior compliance control to pay off, checking for compliance must be affordable, and must scale to large, long-running applications. We thus evaluate whether the size of the model correlates with the runtime of the application. If this were the case, the compliance check could be as expensive as re-performing the actual outsourced computation, hence defeating the purpose of outsourcing.

For the Function and Call Graph abstractions it is immediately obvious that no such correlation can exist. This is because the number of functions, and consequently the number of call-graph edges, is statically bounded. For the Calling

[5] 10 random single-byte changes beyond the first 1024 bytes of data; the latter avoids corrupting the main document header, a case that is particularly easy to identify as abnormal.

Context Tree abstraction, however, this is not the case. In particular, the use of recursion can cause an application's calling context tree to to any size.[6] Figure 5 visualizes the relation between application runtime (with CCT logging enabled) and and the number of nodes in the resulting CCT profile. Interestingly, in our benchmark longer-running applications do *not* induce significantly larger profiles; thus, our approach scales well over time.

5.5 RQ4: Efficiency

We comment on the runtime overhead caused by the instrumentation necessary for profile generation and on the overhead induced by using securely sealed storage.

For the experiments mentioned above, we used a setup as described in Section 2: we collected calling context trees in all cases, and in a second step computed call graphs and method sets based on the collected trees. This procedure is inefficient. In a real-world setting one would rather opt for a customized instrumentation that emits the respective representation directly, as this can safe a significant amount of execution time. While computing full CCTs will generally incur a significant runtime overhead (10 times or more), one can bring overheads down to under 5 % by using probabilistic calling context trees [6]. Such probabilistic CCTs appear quite sufficient for our purposes, and we plan to evaluate their utility in future work. Method sets and call graphs are statically bounded and can therefore be indexed ahead-of-time, which makes instrumentation possible that produces little to no observable runtime overhead [16]. We thereby conclude that sufficiently efficient implementations are possible given the state of the art in dynamic program analysis. While such implementations are outside the scope of this paper, we plan to investigate them in future work.

We measured the runtime cost of our runtime-secure storage on a machine equipped with an AMD Phenom II X2 555 processor and 4 GiB RAM under GNU/Linux (kernel 2.6.32) and the TPM emulator version 0.7.2. Our tests show that the most expensive operation is to create a vTPM instance, which takes 1 second on average. However, this operation is only invoked once, at application startup time. The overhead is caused by the expensive `TPM_TakeOwnership` operation, which creates the Storage Root Key (SRK) key-pair.

The average total cost of storing a CCT profile depends on the average node number. For pdfbox, POI-HSLF and POI-HWPF those are 120,850, 78,568 and 48,239 respectively. Hashing the unique identifier (8 bytes) of every node takes about 6 μs. The instruction `TPM_Extend`, which extends a PCR register with a hash, takes 400 μs. That is, we estimate the overhead of securely storing a full CCT profile for pdfbox, POI-HSLF and POI-HWPF at about 50, 32 and 20 seconds respectively. When using the more coarse-grained Call Graph abstraction, only an average 5,313, 2,338 resp. 2,289 nodes must be stored for pdfbox, POI-HSLF and POI-HWPF respectively, lasting approximately 3.1, 1.95 and 1.93 seconds. The most efficient abstraction are Functions. The overhead for Functions is 2.1, 1.53 and 1.52 seconds for 2,577, 1,301 and 1,281 functions respectively.

[6] In practice, the virtual machine's maximum stack size does impose a (large) limit.

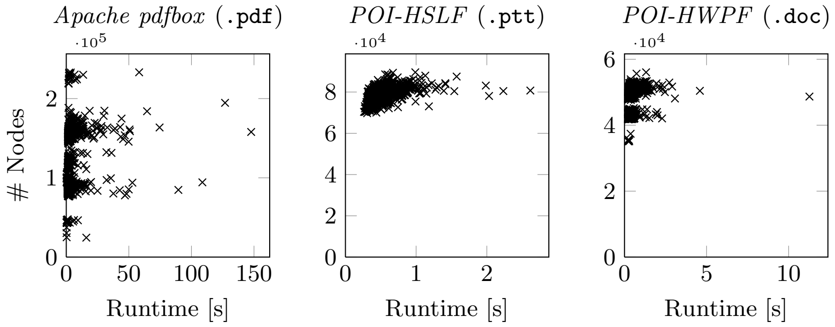


Fig. 5. Relation between application runtime and model size, measured in number of calling context tree nodes

Our results show that the cost of secure storage becomes an issue with CCTs but appears low enough for the other two abstractions. In any case, note that storage can be performed asynchronously on a separate processor core (or even a set of those).

6 Related Work

There is has been a significant amount of previous work on automated property inference [7,24,31,32] and anomaly detection [13,18] on many different levels, both static and dynamic, all with their relative strengths and weaknesses. Many of those approaches could be integrated into our generic architecture defined in Section 3. We decided to define our own set of three behavior abstractions because this setup would allow us to evaluate the relative properties of those abstractions. Our approach extends all previous approaches to anomaly detection by allowing anomalies to be identified in a distributed but trustworthy manner.

Our approach is not the first to capture program behavior in terms of calling-context information. Ammons et al. [1] show how to generate context-sensitive performance profiles efficiently, using hardware performance counters. Dynamic sandboxing, proposed by Inoue et al. [19], shows similarities with behavior compliance control. Like behavior compliance control, dynamic sandboxing relies on dedicated training runs to determine a set of legal behaviors. However, Inoue et al. only consider profiles at function granularity and validate them in two very limited scenarios; in particular, they do not provide a detailed, quantitative evaluation and do not consider a broader applicability of dynamic sandboxing beyond runtime monitoring.

Our approach builds on ideas from intrusion detection. In the mid-nineties, Forrest et al. [9] addressed an important problem in intrusion detection, the definition of what they call “self”, in other words a system’s normal behavior. The authors propose a method to define “self” for privileged Unix processes by recording short sequences of system calls. Behaviors that deviate from these patterns are flagged as anomalous and considered untrustworthy.

None of those approaches considers the scenario of behavior compliance control in cloud computing. In addition, all approaches are black-box approaches (in addition to other similar works mentioned in Section [2]). Compared to our approach, this gives them the advantage of being independent of any programming language or compiler. On the other hand, white-box approaches such as ours yield higher flexibility (as they can obtain more information) and finer granularity.

Other authors have proposed enforcement architectures to control access to data objects distributed to remote systems [37]. Such architectures control how outsourced applications can access outsourced objects at runtime, assuming that these applications are trusted after verifying their load-time integrity. As we discussed before, behavior compliance control goes well beyond such load-time based measures.

Trusted Computing allows to remotely attest the integrity of computing platforms. Behavior compliance control goes beyond binary attestation by not only considering the integrity of the application's code at load-time, but its actual runtime behavior. Gu et al. [15] propose an approach to remote attestation that can be seen as complementary to ours. Behavior compliance control is focused on assessing the compliance of a single application's execution to its model. Gu et al.'s approach, on the other hand, rather focuses on system-wide attestation; the authors attest behavior by measuring the ways in which different processes call each other. In an approach called Semantic Attestation, Vivek et al. [17] propose to use a trusted virtual machine for remote attestation. The core idea is that such a trusted virtual machine is capable of performing code analysis and runtime monitoring. In the approach, the appropriate property checkers need to be programmed manually, though. This is in stark difference to behavior compliance control, in which application models are automatically generated from legal executions. In more recent work, Gu et al. [14] propose an architecture to attest the execution of single mission-critical subroutines of an outsourced application. The authors use the debug facilities of certain CPUs to track the execution of a specific function. The execution of the function is then transferred to a secure environment prepared by a secure kernel.

Finally, some effort has been spent on the construction of schemes for verifiable computation [4, 12], which aim at outsourcing computations to a third party, while offering a proof of correctness for the result. At the moment, these constructions are rather impractical and cannot cope with side-effects of the program execution.

7 Conclusion

We have presented behavior compliance control, a novel approach to increase the trust in the validity of executions of outsourced applications. The approach goes beyond load-time based systems for compliance control by considering the application's runtime behavior. This allows the client outsourcing the application to detect abnormal executions even in cases where the application's code remains unaltered after loading. Such anomalies can for instance be caused by faulty or malicious inputs, misconfigurations, version mismatches or hardware glitches. We have presented a reference architecture for behavior compliance control, and an instantiation for the Java platform which is available as open source.

We have implemented and evaluated our approach based on three different abstractions of runtime behavior: function sets, call graphs and calling context trees. Using a large-scale empirical evaluation we could show that the former two abstractions yield few false positives, while still being able to identify a significant number of abnormal executions caused by malicious inputs, and all cases of abnormal executions caused by fuzzed inputs. Those abstractions can also be implemented efficiently. Calling context trees identify all malicious inputs but also yield a larger number of false positives, causing additional computation cost in the client's private cloud. It is hence up to the client to decide whether this additional cost is justified by the increased trust that this abstraction offers.

An interesting piece of future work would be to evaluate optimized implementations of behavior abstractions in the apparent sweet spot between full calling context trees and call graphs. An approach with bounded context strings paired with probabilistic calling contexts [6] appears like a potentially optimal candidate in this solution space.

Acknowledgements. This work was supported by the German Federal Ministry of Education and Research (BMBF) within EC SPRIDE and by the Hessian LOEWE excellence initiative within CASED.

References

1. Ammons, G., Ball, T., Larus, J.R.: Exploiting hardware performance counters with flow and context sensitive profiling. In: Proc. of the 10th Conference on Programming Language Design and Implementation (PLDI), pp. 85–96 (1997)
2. Apache Software Foundation. The Apache Java PDF Library (PDFbox), `http://pdfbox.apache.org/`
3. Apache Software Foundation. The Java API for Microsoft Documents (Apache POI), `http://poi.apache.org/`
4. Benabbas, S., Gennaro, R., Vahlis, Y.: Verifiable Delegation of Computation over Large Datasets. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 111–131. Springer, Heidelberg (2011)
5. Bodden, E., Sewe, A., Sinschek, J., Oueslati, H., Mezini, M.: Taming reflection: Aiding static analysis in the presence of reflection and custom class loaders. In: Proc. of the 33rd International Conference on Software Engineering (ICSE), pp. 241–250 (2011)
6. Bond, M.D., McKinley, K.S.: Probabilistic calling context. In: Proc. of the 22nd Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), pp. 97–112 (2007)
7. Ernst, M.D., Cockrell, J., Griswold, W.G., Notkin, D.: Dynamically discovering likely program invariants to support program evolution. In: Proc. of the 21st International Conference on Software Engineering (ICSE), pp. 213–224 (1999)

8. Feng, H.H., Kolesnikov, O.M., Fogla, P., Lee, W., Gong, W.: Anomaly detection using call stack information. In: Proc. of the 2003 IEEE Symposium on Security and Privacy (S&P), pp. 62–75 (2003)
9. Forrest, S., Hofmeyr, S.A., Somayaji, A., Longstaff, T.A.: A sense of self for Unix processes. In: Proc. of the 1996 Symposium on Security and Privacy (S&P), pp. 120–128 (1996)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional (1994)
11. Gao, D., Reiter, M.K., Song, D.: Gray-box extraction of execution graphs for anomaly detection. In: Proc. of the 11th Conference on Computer and Communications Security (CCS), pp. 318–329 (2004)
12. Gennaro, R., Gentry, C., Parno, B.: Non-interactive Verifiable Computing: Outsourcing Computation to Untrusted Workers. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 465–482. Springer, Heidelberg (2010)
13. Gruska, N., Wasylkowski, A., Zeller, A.: Learning from 6,000 projects: Lightweight cross-project anomaly detection. In: Proc. of the 19th International Symposium on Software Testing and Analysis (ISSTA), pp. 119–130 (2010)
14. Gu, L., Cheng, Y., Ding, X., Deng, R.H., Guo, Y., Shao, W.: Remote Attestation on Function Execution (Work-in-Progress). In: Chen, L., Yung, M. (eds.) INTRUST 2009. LNCS, vol. 6163, pp. 60–72. Springer, Heidelberg (2010)
15. Gu, L., Ding, X., Deng, R.H., Xie, B., Mei, H.: Remote attestation on program execution. In: Proc. of the 3rd Workshop on Scalable Trusted Computing (STC), pp. 11–20 (2008)
16. Gutzmann, T., Löwe, W.: Custom-made instrumentation based on static analysis. In: Proc. of the 9th International Workshop on Dynamic Analysis, WODA (2011)
17. Haldar, V., Chandra, D., Franz, M.: Semantic remote attestation: a virtual machine directed approach to trusted computing. In: Proc. of the 3rd Conference on Virtual Machine Research and Technology Symposium, pp. 3–20 (2004)
18. Hangal, S., Lam, M.S.: Tracking down software bugs using automatic anomaly detection. In: Proc. of the 24th International Conference on Software Engineering (ICSE), pp. 291–301 (2002)
19. Inoue, H., Forrest, S.: Anomaly intrusion detection in dynamic execution environments. In: Proc. of the 2002 Workshop on New Security Paradigms (NSPW), pp. 52–60 (2002)
20. Karabulut, Y., Kerschbaum, F., Massacci, F., Robinson, P., Yautsiukhin, A.: Security and trust in IT business outsourcing: a manifesto. ENTCS 179, 47–58 (2007)
21. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proc. of the International Joint Conference on Artificial Intelligence (IJCAI), pp. 1137–1143 (1995)
22. Lacity, M.C., Khan, S.A., Willcocks, L.P.: A review of the IT outsourcing literature: Insights for practice. The Journal of Strategic Information Systems 18(3), 130–146 (2009)
23. Liang, S., Bracha, G.: Dynamic class loading in the java virtual machine. In: Proc. of the 13th Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), pp. 36–44 (1998)
24. Pradel, M., Gross, T.R.: Automatic generation of object usage specifications from large method traces. In: Proc. of the 24th International Conference on Automated Software Engineering (ASE), pp. 371–382 (2009)
25. Sailer, R., Zhang, X., Jaeger, T., van Doorn, L.: Design and implementation of a TCG-based integrity measurement architecture. In: Proc. of the 13th USENIX Security Symposium, pp. 1–16 (2004)

26. Sarimbekov, A., Sewe, A., Binder, W., Moret, P., Mezini, M.: JP2: Call-site aware calling context profiling for the Java Virtual Machine. Science of Computer Programming (2012), doi:10.1016/j.scico.2011.11.003
27. Sarimbekov, A., Sewe, A., Binder, W., Moret, P., Schöberl, M., Mezini, M.: Portable and accurate collection of calling-context-sensitive bytecode metrics for the Java Virtual Machine. In: Proc. of the 9th Conference on the Principles and Practice of Programming in Java (PPPJ), pp. 11–20 (2011)
28. Scarlata, V., Rozas, C., Wiseman, M., Grawrock, D., Vishik, C.: Tpm virtualization: Building a general framework. In: Pohlmann, N., Reimer, H. (eds.) Trusted Computing, pp. 43–56. Vieweg+Teubner (2008)
29. Strasser, M., Stamer, H.: A Software-Based Trusted Platform Module Emulator. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968, pp. 33–47. Springer, Heidelberg (2008)
30. Tan, K., McHugh, J., Killourhy, K.: Hiding Intrusions: From the Abnormal to the Normal and Beyond. In: Petitcolas, F.A.P. (ed.) IH 2002. LNCS, vol. 2578, pp. 1–17. Springer, Heidelberg (2003)
31. Thummalapenta, S., Xie, T.: Alattin: Mining alternative patterns for detecting neglected conditions. In: Proc. of the 24th International Conference on Automated Software Engineering (ASE), pp. 283–294 (2009)
32. Thummalapenta, S., Xie, T.: Mining exception-handling rules as sequence association rules. In: Proc. of the 31st International Conference on Software Engineering (ICSE), pp. 496–506 (2009)
33. The tpm4java library, http://sourceforge.net/projects/tpm4java/
34. Trusted Computing Group, Inc. TPM Main Specification Level 2 Version 1.2, Revision 116 (March 2011)
35. The TrustedGRUB extension to the GRUB bootloader, http://sourceforge.net/projects/trustedgrub/
36. Wagner, D., Soto, P.: Mimicry attacks on host-based intrusion detection systems. In: Proc. of the 9th Conference on Computer and Communications Security (CCS), pp. 255–264 (2002)
37. Zhang, X., Seifert, J.-P., Sandhu, R.: Security enforcement model for distributed usage control. In: Proc. of the Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), pp. 10–18 (2008)

Appendix

Definition 1: (Function set) Let r be a monitored program run. Then the function set of r, which we denote by $\mathbf{functionSet}(r)$, is the smallest set fulfilling the following property: For any invocation $f \rightarrow f'$ of function f' from function f on r, it holds that $\{f, f'\} \subseteq \mathbf{functionSet}(r)$.

Definition 2: (Call graph) A call graph is a directed graph (V, E) with V a set of nodes representing functions, and $E \subseteq V \times V$ a set of directed edges. Then the call graph of r, $\mathbf{cg}(r)$, is a call graph that fulfills the following constraints. V is the smallest set such that for any invocation $f \rightarrow f'$ of function f' from function f on r, it holds that $\{f, f'\} \subseteq V$. E is the smallest subset of $V \times V$ such that for each such f, f' it holds that $(f, f') \in E$.

Definition 3: (Calling context tree) Let F be the set of all function identifiers. Then C_M, the set of all calling contexts over F, is defined as $C_M := F^+$. The set C_M is closed under concatenation: we define a concatenation function "$\cdot$" on calling contexts such that for any context $c \in C_M$ and function $f \in F$ it holds that $c \cdot f \in C_M$. A calling context tree is a tree (V, E) with $V \subseteq C_M$ a set of nodes representing calling contexts and $E \subseteq V \times V$ a parent-child relationship. We further demand that there exists a unique root node v_0 which has no parents, i.e., for which it holds that $\neg\exists v \in V$ s.th. $(v, v_0) \in E$. Let r be a monitored program run. Then $\mathbf{cct}(r)$ is a calling context tree for which the following holds. V is the smallest set such that for any invocation $c \rightarrow f$ of function f from within context c on r, it holds that $\{c, c \cdot f\} \subseteq V$. E is the smallest subset of $V \times V$ such that for each such f, c it holds that $(c, c \cdot f) \in E$.

An Empirical Study of Dangerous Behaviors in Firefox Extensions

Jiangang Wang[1], Xiaohong Li[⋆,1], Xuhui Liu[2], Xinshu Dong[2], Junjie Wang[1], Zhenkai Liang[2], and Zhiyong Feng[1]

[1] Department of Computer Science, Tianjin University
xiaohongli@tju.edu.cn
[2] School of Computing, National University of Singapore

Abstract. Browser extensions provide additional functionality and customization to browsers. To support such functionality, extensions interact with browsers through a set of APIs of different privilege levels. As shown in previous studies, browser extensions are often granted more privileges than necessary. Extensions can directly threaten the host system as well as web applications, or bring in indirect threats to web sessions by injecting contents into web pages. In this paper, we make an empirical study to analyze extension behaviors, especially the behaviors that affect web sessions. We developed a dynamic technique to track the behaviors of injected scripts and analyzed the impact of these scripts. We analyzed the behaviors of 2465 extensions and discussed their security implications. We also proposed a solution to mitigate indirect threats to web sessions.

1 Introduction

Browser extensions are widely adopted by modern web browsers to allow users to customize their web browsers. Browser extensions can change display of web pages, improve browsers' behaviors, and introduce new features. To support such functionalities, extensions need to monitor and modify web contents, change browser behaviors and appearances, access stored website passwords, cookies, etc. They can issue HTTP requests on behalf of users, and open sockets to listen to connections or connect to remote servers. They can even access the local file system and launch processes. On one hand, all these privileges enable extensions to extend the functionality of browsers; on the other hand, they increase the attack surface of users' systems.

Threats from extensions can be categorized into two types: *direct threats* and *indirect threats*. *Direct threats* arise from extensions' direct access to critical browser resources or the local system. Extensions can read and write Document Object Module (DOM) of any web page by which page contents and user keystrokes are visible to extensions. By inserting new elements to DOM, extensions can easily push contents to users at the extension developer's will. They also can access browser components directly, such as Firefox's password manager. Extensions can also access network and access local files with current system user's privileges.

Indirect threats are threats to web sessions. By injecting code into a page, browser extensions can easily take full control of a web session. Although this is an intended

[⋆] Corresponding author.

functionality of browser extensions, malicious extensions can take advantage of it to carry out attacks against web sessions.

To limit extensions' high privileges in browsers, several solutions have been developed. Ter Louw et al. proposed a solution [1,2] to enhance browser extension security by protecting *the integrity of browser code base* and *the confidentiality and integrity of user data*. It provides an isolation mechanism to prevent one malicious extension from compromising another. Google Chrome has adopted an extension system designed with *least privilege, privilege separation*, and *strong isolation* [3]. Each extension is granted a set of privileges at installation time and it cannot exceed the granted privileges during execution time. A recent proposal [4] further divides privileges of Chrome extensions into micro-privileges, to restrict extensions' capabilities in cross-site requests and DOM element accesses. It also introduces resources requesting new origins. To mitigate threats from over-privileged extensions, Mozilla has developed a new extension development framework Jetpack [5] to assist developers in building extensions following the principle of least authority (POLA).

To gain a better understanding of dangerous behaviors in Firefox extensions, we performed an empirical study of real-world Firefox extensions. Instead of looking for vulnerabilities in browser extensions [6], our focus is on dangerous runtime behaviors of browser extensions. We specifically study indirect threats to web sessions, and discuss what are missing from existing solution in dealing with them.

We developed an automatic testing system based on instrumented Firefox to dynamically investigate extension behaviors. Using this system, we studied Firefox extensions hosted in the Mozilla Addons repository [7]. In total, 2465 extensions had been tested, which were distributed in 13 different categories. We summarize our findings of dangerous behaviors in Firefox extensions, and discuss potential improvements in mitigating indirect threats while maintaining usability.

This paper makes the following contributions:

1. We perform an empirical study of Firefox extension behaviors by monitoring their runtime behaviors, with a focus on indirect threats. From our analysis, we propose improvement to current solutions in mitigating indirect extension threats to web sessions.
2. We designed and implemented an automatic testing system to monitor the browser extension behaviors.

2 Threats from Extension

2.1 Browser Extensions

Browser extensions are tools to extend browsers to enhance their functionality. To obtain highly customizable features, extensions are granted high privileges. They can access almost all components in a browser. The extension privileges can be summarized as follows:

– Monitoring and modifying web contents. All contents displayed in web page are visible to extensions including user inputs. Extension can modify the page by modifying DOM.

- Accessing browser components. Extensions can access browsers' password managers, cookie files, etc.
- Accessing network. Extensions also have full access to network and sockets. It can easily issue HTTP requests.
- Accessing local file systems. Extensions can access local file system with the privileges of the browser process.
- Launching processes. Extensions can launch processes through certain browser API.

As discussed, extensions have powerful privileges. In the following subsections, we will discuss what a malicious extension can do with these powerful privileges. In the rest of the paper, we focus on Firefox extensions.

2.2 Direct Threats

Direct threats are threats from browser extensions, through directly accessing critical resources of browsers or host systems.

- **Direct DOM Access.** Extensions can access all DOM structures in any page, including contents browsed by users, contents in all forms filled and submitted by users, and events on keystrokes and mouse clicks. They can also access all information submitted to the server, including account names and passwords, personal information, finance information, etc. They can not only read DOM, but also modify DOM to create new page contents.
- **Browser Component Access.** Extensions are capable to access all Firefox components, including the password manager and the browsing history, etc. Consequently, they can easily collect account information, such as passwords stored in the password manager and the browsing history.
- **Arbitrary File Access.** Extensions have the ability to access local file systems without any restriction. Once a malicious extension is installed, the whole system is controlled by attackers with the browser user's privilege. Since extensions can access arbitrary files and the extension system does not provide a mechanism to protect the integrity of installed extensions, it is possible for a malicious extension to change other benign extensions' behaviors by modifying their installed files.
- **Network Access.** Extensions have variable ways to access the network, such as sending XMLHttpRequest directly, or requesting resources like images or scripts from arbitrary servers. By attaching information to requests, malicious extensions can send out collected information.
- **Launching Process.** Launching a local process is considered highly dangerous, because by running malicious code, an attacker can take full control of victims' computers and make them part of a botnet.
- **Dynamic Code Execution.** Executing dynamic code is an important feature in the JavaScript language, which enables the program to execute dynamically generated strings as JavaScript code. The strings may originate from an attacker, allowing them execute malicious code.
- **Listening to Keyboard Events.** Listening to keyboard events is common in extensions, but an attacker can use this feature to steal users' inputs.

The root cause of direct threats is the high privileges that a browser grants to extensions. Different browser vendors proposed different mechanisms to mitigate these threats. Firefox browser adopts a Sandbox Review System to force all submitted extensions to be manually reviewed before they are released to the public. Google Chrome adopts a new extension system [3] with "least privileges" principle. The new system requires developers to claim the minimal privileges their extensions need. When an extension is installed, a prompt dialog pops up to warn the users what privileges the extension claims, whether to continue installing or not is determined by users. At run time, an extension cannot exceed privileges it claimed.

2.3 Indirect Threats to Web Sessions

Extensions introduce indirect behaviors through two steps. It first injects contents (script code or HTML elements) into web sessions; The injected contents then result in subsequent behaviors, which may launch attacks against the web sessions. Extensions have various ways to inject contents to web pages. They can call the $write$ or $writeln$ methods of DOM objects, $appendChild$ or $insertBefore$ methods of HTMLElement objects.

Unfortunately, current browsers lack the abilities to distinguish injected contents from original ones. The indirect threats completely bypass permission checking on the extension itself, and can launch attacks to web sessions. These attacks can be launched in all pages, regardless they are vulnerable or not.

Let us use a scenario to demonstrate how it can be exploited by attacks. An extension claims for privileges to access the browsing history and DOM, but it does not require the privilege to access the network. It seems that this claim should be allowed because although the extension may access users' private data, it cannot send them to untrusted third party through network APIs. However, direct access to network has been limited, but this extension can still access network in an indirect way. The extension can modify the DOM tree of benign web pages and inject <script>, <img> or <iframe> tags into pages, which use a malicious site as its source, and attach users' private data or cookies of target web sites as parameters. Such indirect threats have not been well studied by existing research work, and are also not constrained by existing browser extension systems.

All the behaviors conducted by extensions are within the context of browsers. So, it's almost impossible for traditional general protection mechanisms (e.g., anti-virus software or firewall) to judge certain behaviors are conducted by extensions or browsers themselves, let alone to tell these behaviors are benign or not. The protection mechanism again malicious extensions can only be deployed within browsers.

3 Design and Implementation

In this section, we present the design and implementation of our extension testing system that can automatically download, install, and test browser extensions and obtain their behavior information. We first illustrate the architecture of the system, then introduce the design and implementation of modules in our system in detail, and finally explain the testing process of browser extensions.

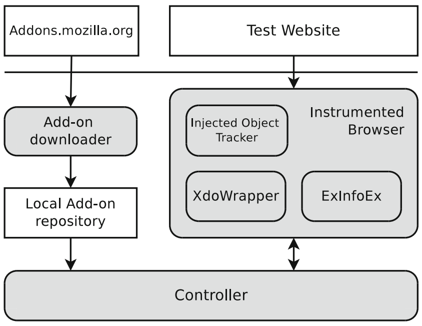


Fig. 1. System architecture of our Firefox extension testing system. The main components are in grey color.

Although our approach and study are based on Firefox, our approach does not depend on specific feature of Firefox, and our study is performed assuming the availability of protection mechanism on all browsers. Our solution and study can be straightforwardly extended to other browsers.

3.1 System Architecture

The system architecture is shown in Figure 3.1. There are six modules in our extension testing system. **Instrumented Browser** monitors the execution of browser extensions. It contains three sub-modules, **Injected Object Tracker** is a module to track whether an HTML element or a piece of JavaScript is originated from web page or injected by extensions; **ExInfoEx** is used for extracting the position information of items (such as menu items) added by tested extension and offering them to simulating module for triggering the events; **XdoWrapper** is a wrapper of Xdotool [8] that makes it easy to simulate many kinds of events at the window level. **Addon-Downloader** is responsible for downloading all the analyzable browser extensions to the local system. **Controller** is a module that connects other modules together and makes the whole test process automatic. We also created a test website to drive the testing of extensions.

3.2 Design and Implementation

We give the detailed introduction about the design and implementation of the major components in our system.

Instrumented Browser. In order to investigate the tested extension's behaviors, we intercept the interfaces or functions that are accessed by the tested extension at runtime using an instrumented Mozilla Firefox 3.5.

In this customized browser, for detecting the direct threats of an extension, we inserted a series of hooks into the browser's source code. We mainly hook four modules of Firefox: the XPConnect module, the DOM module, the Content module, and the

JavaScript engine. XPCOM makes it possible for JavaScript to invoke methods provided by the browser. The XPConnect module is responsible for the communication between JavaScript and XPCOM components. By inserting hooks into this module, we can intercept the events of browser extension accessing XPCOM interfaces, and extract the parameters. In the DOM module and the Content module, we inserted hooks to intercept event listener registration and removal issued by a browser extension. SpiderMonkey [9], the JavaScript engine in Firefox, is also instrumented to intercept several security-related JavaScript native functions for analysis.

The instrumented browser contains three sub-modules:

Injected Object Tracker. To investigate the indirect behaviors of an extension, it is necessary to track the contents injected into web pages by extensions and monitor their subsequent behaviors. For this purpose, we design the Injected Object Tracker to track the source of injected contents. We intercept interfaces used by extensions to inject new contents to web pages. Once extensions inject any object into web pages, the injected object is marked with the source "extension". During behavior monitoring, we can find out whether the behavior is from the original web page or from objects injected by extensions.

ExInfoEx. ExInfoEx (Extension Information Extractor) is a browser extension that identifies the browser user interface elements that are added by the tested extension and transfer the position information of these elements to XdoWrapper module to trigger the events targeted on them. However, due to the technology of "overlay", how to distinguish the elements added by an extension from the elements belonging to browser itself becomes the biggest challenge we faced.

To resolve this problem, we modify the process of overlay loading. When the browser creates elements defined in overlays in the process of loading, an extra attribute is added to each element. The value of this attribute is set to the extension's identifier that is still retained by the browser at this time and can be easily extracted. With this attribute, ExInfoEx can identify those elements by traversing the DOM tree of the browser user interface.

XdoWrapper. For exposing as many behaviors of a browser extension as possible, we need to simulate users' behaviors to trigger the events in web pages and in the chrome area. XdoWrapper is such a module that receives specific instructions from the ExInfoEx module to simulate these users' behaviors, and provides the relevant information about the windows of browser to the Controller module. It is a wrapper of xdotool [8], which can conveniently generate various keyboard and mouse events at the window level and easily manage and manipulate all the windows opened in the operating system.

Controller. The Controller module is used to connect other modules together and coordinate the entire testing process. Specifically, it is mainly responsible for the following work:

– Install/uninstall extensions. Controller is responsible for the installation and uninstallation of each extension.

- Start/shutdown the browser. During the whole test process, there are many cases need to start or shutdown the browser.
- Configure/clean the testing environment. Before or after an extension is tested, the testing environment needs to be configured with the corresponding information or cleaned up by the controller module.
- Coordinate the whole process. All these processes (install, configure, test, and uninstall) should be connected together smoothly, and this is one of the main tasks of Controller.
- Handling exceptions. Controller also handles unexpected issues, such as browser crashes.

Addon Downloader. The Addon Downloader module is responsible for downloading all the testable browser extensions in the online repository. We utilize Htmlcxx [10] (a lightweight HTML and CSS parser for C++) to parse an extension's page and determine whether the extension is testable, and then with the help of Libcurl [11](a client-side file transfer library), download it to our system.

3.3 Testing Process

We first download all the testable browser extensions in the repository. Each extension downloaded to the local system is then installed on the instrumented browser for testing. After installation, essential information about the extension, such as name, ID (if applicable) and installation path, is extracted from the browser profile and recorded into a configuration file, which is used by the browser in the phase of monitoring extension's behaviors. We then restart the browser to begin the behavior monitoring process. At the beginning of restart, the browser reads the information about the tested extension from the configuration file. This information is indispensable for browsers to detect the behavior information of the tested extension. Once the browser obtains this information during restart, the behavior monitoring process is automatically started. For exposing more behaviors of the tested extension, the system first leads browser to visit a particular website designed by ourselves and simulates a variety of events in web pages, such as click on links, keyboard input, and form submission. Then, the system goes through the browser interface, detects the elements (such as menu items, context menu items and status bar items) added by the tested extension, and generates events corresponding to clicks on each these elements. If new windows pop up after clicks, they are simply closed. After that, the browser is shut down, and the tested extension is removed. Relevant configuration information related to this extension is also cleaned up. After all the above procedures are over, the testing process for one extension is completed, and the process for next extension can start.

4 Evaluation and Analysis

The experiment was conducted on a computer with Intel(R) Core(TM)2 Duo CPU at 2.33GH, 250GB 7200RPM disk, and 4GB RAM. Its operating system is Ubuntu 10.04, with the instrumented Firefox 3.5 installed.

Table 1. The ratings of extension behaviors

Rating	Behaviors
high	Arbitrary file access; Process launching; Download; XPinstall; Network access via XPCOM APIs; Update; DOM injection
medium	Password; Login; Cookie; Network access via XMLHttpRequest; Addons management; Changing Firefox preferences; Profile
low	History; Bookmark; Clipboard; Dynamic code execution
none	Accessibility; Browser core; Auto complete; Log to console; Searching; Spell checking; DOM; Editor; Internationalization; Offline cache; XML parser; Network utilities; RSS/RDF; Data types and structures; Streams; Memory management; Thread management; Component management; Additional XPCOM services; JavaScript core; JavaScript debugger; XPConnect; Authentication; Certification; Cryptograph; Additional security interfaces; Document handling; Transaction management; Web worker; Window management; Print; Database access; Images; Zip/jar process; JVM; Plugins

We first investigated the behaviors of Firefox extensions by analyzing the critical XPCOM interfaces and JavaScript functions exposed to extensions, and classified them into different levels according to their potential risks. Then we further studied Firefox extension behaviors by testing 2465 extensions hosted on the Mozilla Addons website [7] using our extension testing system, and analyzed the experimental data from various perspectives.

4.1 Studying and Classifying Extension Behaviors

Through the analysis of critical XPCOM interfaces and JavaScript functions used by Firefox extensions, we summarize them into 54 different behaviors. To investigate extension behaviors from a security perspective, all these behaviors are rated into four levels according to their potential risks. This work is mainly based on Firefox security severity ratings [12], Chromium's severity guidelines for security issues [13] and the work of Barth et al [3]. The ratings are described in Table 1.

Rating *High* includes the behaviors that can possibly download, install, or execute a program. These behaviors are considered the most privileged and dangerous ones, because they could be utilized by an attacker to compromise users' entire operating system. "DOM injection" is also classified into *High* because it can inject DOM contents into web applications that in turn initiate network access. Rating *Medium* consists of behaviors that may access users' private information (e.g., the password or cookie), or access network in a relatively safe way (like through XMLHttpRequest). The behaviors that may modify the critical data of the browser are also rated as this level. The behaviors in Rating *Low* are those that may obtain limited information of users, such as browsing history or bookmark, and those that are likely to cause a vulnerability with low possibility, like executing a dynamically generated string. All other non-sensitive behaviors are classified into Rating *None*. These ratings are used to investigate how many browser extensions exhibit high privileged behaviors and how the distribution of the usage of different behaviors looks like.

The Statistics of Extension Behaviors. To better understand the behaviors of Firefox extensions, we performed a study of statistics on their behaviors according to their ratings. We four that 33% of them had the behaviors belonging to Rating High. The extensions with the behaviors of Rating Medium accounted for nearly 16% of the total and only 1% of the tested extensions possessed the behaviors of Rating Low. A large number of the tested extensions, approximately 50% of them, only demonstrated the behaviors with Rating None, which are invariably benign to users. Note that one extension with higher privileged behaviors does not indicate that it will definitely cause a damage, but means that the extension has such capabilities that, if abused, will bring severe threats to users' data and/or the underlying operating system. We have known that Firefox provides the browser extensions with excessive privileges, and from our experiments we found that the extensions on Mozilla indeed utilize these high privileges widely.

To investigate the extensions' behaviors in detail from a security perspective, Table 2 lists the frequency of security-related behaviors of Ratings *High*, *Medium* and *Low*. In Rating *High*, there were 37 (1.50%) extensions found to access files outside their installation directory. Four (0.16%) extensions launched a process on the local system, and 10 (0.41%) use the download API provided by XPCOM. None of the extensions were found to install other addons through XPInstall API. The most widely used interfaces in this group are XPCOM interfaces for network accessing, accounting for 29.66% of the entire test set. Only one extension in our experiment uses the Update system. Additionally, there were 108 extensions (4.38%) found to inject new contents into web pages.

In Rating *Medium*, we found no extensions that access password information, while the login and cookie information were found to be accessed by some extensions, accounting for 1.78% and 2.80%, of the total number of extensions. There were 441 (17.89%) extensions issuing HTTP requests through the XMLHttpRequest object. The number of extensions that used XPCOM APIs to manage addons (e.g., to search and install addon from the repository or to locate the installation location) were 139 (5.64%). Although changing Firefox preferences may greatly annoy users, there were still 31 (1.26%) extensions doing so. In addition, one extension was observed to access the profile information of the Firefox.

In Rating *Low*, there were 43 (1.74%) and 44 (1.78%) extensions found to access the limited information, the browsing history and bookmarks. 49 of the tested extensions, about 1.99%, were found to manipulate the clipboard through XPCOM APIs. To our surprise, although Mozilla has highly recommended not to execute dynamic generated strings with "eval()" or "Function()", and several safer alternatives are also available, there were still 145 extensions showing such behaviors, accounting for 5.88% of the total number of extensions examined.

4.2 Extension's Indirect Threats to Web Sessions

To investigate the indirect threats from browser extensions, our experiment paid special attentions to *indirect behaviors* of extensions that inject new web contents into web pages. As shown in Table 3, there were 108 browser extensions that inject new contents

Table 2. Frequency of security-related behaviors

Rating	Behavior	Quantity	Frequency
High	Arbitrary file access	37	1.50%
	Process launching	4	0.16%
	Download	10	0.41%
	XPInstall	0	0.00%
	Network access via XPCOM API	731	29.66%
	Update	1	0.04%
	DOM injection	108	4.38%
Medium	Password	0	0.00%
	Login	44	1.78%
	Cookie	69	2.80%
	Network access via XMLHttpRequest	441	17.89%
	Addons management	139	5.64%
	Changing Firefox preferences	31	1.26%
	Profile	1	0.04%
Low	History	43	1.74%
	Bookmark	44	1.78%
	Clipboard	49	1.99%
	Dynamic code execution	145	5.88%

Table 3. Statistics of contents injected into webpages

Injected Element	Quantity	Total	Ratio
script	14	108	12.96%
iframe	8	108	7.41%
a	8	108	7.41%
img	14	108	12.96%
object/embed	3	108	2.78%
others	95	108	87.96%

into web pages, among which 14 extensions inserted "script" tags into pages, eight extensions inserted "iframe" tags into pages, and other 14 extensions inserted "img" tags. There were some other elements that were also found to be injected into pages by extensions, such as "div", "span" and so on. We focus our analysis on those cases that extensions injected elements that are frequently exploited in attacks, including script, iframe and img tags. By manually examining these cases, we conclude that all the extensions that inject new contents into web pages, were exempted from being malicious.

For the 14 extensions that injected "script" elements into web pages, six of them directly injected JavaScript code into pages, while the remaining eight extensions injected "script" elements that request external JavaScript from the developers' websites or third-party ones.

For the six extensions that directly injected code, the injected code's behaviors were rather simple. Three of them simply set certain JavaScript objects' values to 1, as shown

```
    <SCRIPT type=”text/javascript”>
    window.script1309754027588=1;
    </SCRIPT>
```

Fig. 2. JavaScript code to assign an object

```
<SCRIPT id=”afterthedeadline−dispatchDisable”>
  if (window.setTimeout)
    window.setTimeout(function(){
      if (”AtD” in window || ”AtDCore” in window) {
        document.AtDdisabled = true;
        var ev = document.createEvent(”HTMLEvents”);
        ev.initEvent(”disableAtD”,true,false);
        document.dispatchEvent(ev);
      }
  },100);
</SCRIPT>
```

Fig. 3. JavaScript code to disable extension

in Figure 2; two of them injected empty code; and one of them set a timer event to disable itself, as shown in Figure 3. We also closely examined the eight extensions that injected “script” elements with *src* attributes to refer external JavaScript. We found that one of them accesses the cookie that was believed to belong to its own website. In another case, the injected script called the insecure function “eval” to load dynamic code. However, we have confirmed that this case is also benign.

In the eight extensions that injected “iframe” HTML elements into web pages, the requested pages loaded by iframes all came from the corresponding extension’s official website. For the behaviors of these injected pages in iframes, we also find two pages that access the cookie of their own.

Other injection behaviors are also innocuous. All the “a” HTML elements injected into web pages by the eight extensions do not link to other pages. The 14 extensions inject “img” HTML element into web pages, using an image either from their chrome areas or from other sites as the source of “img” tag. All these images used as normal were icons and did not contain malicious event handlers. The three extensions that injected “object” or “embed” HTML elements are to include flash files to web pages.

From the experimental results, we found that subsequent behaviors of web contents injected into web sessions by extensions can be summarized in three groups.

– Network Access. Via $script$, img, $iframe$ or $embedded$ tags.
– Cookie Access. Via $script$ tags.
– Others. Certain injected code assigns simple values to certain objects, or register event listeners. None of these is security related.

Table 4. Statistics of some dangerous functionalities and practices

Dangerous Practice	Quantity	Total	Ratio
Accessing files beyond the directory of its own.	37	2465	1.50%
Launching a process.	4	2465	0.16%
Issuing HTTP requests via XMLHttpRequest.	441	2465	17.89%
Changing the preferences of the browser.	31	2465	1.26%
Capability of dynamic code execution(evel, Function).	145	2465	5.88%
Listening to the keyboard events targeted at web pages.	19	2465	0.77%
Preference names without "extensions." prefix.	266	2465	10.79%

4.3 Direct Threats

Dangerous Functionalities and Practices. In our experimental analysis, we also focused on dangerous functionality and practices. Table 4 lists the statistical results of our experiment.

Accessing files beyond the directory of its own. Browser extensions may have legitimate needs to access local files, so we focused on accesses to files outside the extensions' installation directories by monitoring the invocations to the "initWithPath" function of the "nsILocalFile" interface and its parameter. Among the 37 extensions with such behaviors, 15 extensions access files from "/home/username", and surprisingly there are as many as 7 extensions found to access files from "/usr/bin", 5 accessing "/tmp", and 2 accessing "/bin". After manual inspection, all of them were confirmed to be not malicious.

Launching a process. We monitor process launching by intercepting the "run" method of the "nsIProcess" interface. In these four extensions that launch new processes, two of them (KidZui and KidZui K2) executed the program "/usr/bin/xmodmap" to modify keymaps and pointer button mappings in the X system. Another extension (Zotero) executed the program "/usr/bin/mkfifo" to create a named pipe. Moreover, the extension named "TTS for linux" executed a shell script, which is responsible for converting text to speech by calling KTTSD (Kde Text To Speech Daemon) via DCOP (Desktop COmmunication Protocol).

Issuing HTTP requests via XMLHttpRequest. We found 33 extensions sending current page URLs to remote servers using XMLHttpRequest, which might be due to the fact that browsers do not append the `referrer` header to the HTTP request generated by extensions using XMLHttpRequest by default. Although no abuse of such information was found, it is possible for malicious extensions to use this approach to collect users favors or sense users' behaviors that may not be acceptable for some users.

Changing the preferences of the browser. The modifications to some of the browser preferences can lead to exploit or at least annoying users. To evaluate the situation among Firefox extensions that how many extensions access the critical preferences of the browser, we intercepted the "getBranch" method of the "nsIPrefService" interface and logged its parameters. We found 20 extensions accessed "network.*" preferences, 8 accessed "general.useragent.*" accesses, etc.

Capability of dynamic code execution. We found that 145 (5.88%) extensions still use the dangerous JavaScript features “eval()” or “Function()”, despite the existence of safer alternatives. These dangerous practices could leave the door open to attackers. Their existence showed that Mozilla’s review process does not completely eliminate dangerous and vulnerable coding practices, although Mozilla’s Add-on Review Guide [14] clearly states that any extension using “eval()” to evaluate remote code should be rejected.

Listening to the keyboard events on web pages. We found 19 extensions that monitor the user’s keyboard operations. However, after our manual reviewing, considering the particular functionality these extensions need to implement, such event monitoring is considered essential for their functional goals. For example, a translator extension “Nice Translator” accelerates the query speed by recording the user input in real time. And another extension “gleeBox” provides a keyboard-centric approach to navigating the web, which also needs to monitor the keyboard events.

Preference names without the “extensions” prefix. Mozilla suggests that it is a good practice to name extension preferences with an “extensions” prefix; otherwise, extension preferences’ names may pollute the namespace of the browser’s own preference system and affect the stability of the browser. However, during our experiment, we still found up to 266 extensions violating this rule, accounting for 10.79% of the total.

5 Discussion

5.1 Tracking Principals of Indirect Behaviors

Indirect threats to web sessions come from injected contents by browser extensions. A recent proposal [4] introduces new permissions to have fine-grained control on accesses to DOM elements and the capability to introduce new origins into web sessions. This is a promising direction in mitigating indirect threats from browser extensions. However, there are other cases of indirect threats other than introducing new origins via the `src` attributes of “img” of “iframe” HTML tags. For example, things will get more complicated when a script injected into a web session by a browser extension can dynamically create another script, which in turn modifies the web page’s original JavaScript to tamper with their original XMLHttpRequest destinations.

We argue that instead of imposing an allow or disallow option for browser extensions to introduce new origins into web sessions, we need a more systematic approach, which tracks the principals of contents from different sources in web sessions. When injected contents from browser extensions modify original contents in the web page, the principal of the injected content should be propagated to the modified contents. As a result, additional security mechanisms, such as permission checking or access control can be applied on the dynamic principals of different components in the same web session. This approach would not only mitigate indirect threats to web sessions from browser extensions, but would allow legitimate interactions between browser extensions and web sessions.

5.2 Coverage of Extension Behaviors

In our testing system, we setup a simple web site to simulate the real world web sites, and we use XdoWrapper to simulate users’ click or keyboard strike events. However, some extensions are designed to work only on specific web sites and some behaviors will only be triggered by certain user actions. Since we are not able to simulate the exact environments for them, it is possible that certain behaviors are not triggered in our testing system. As our future work, we will work on solutions to achieve better coverage of extension behaviors with assistance from static program analysis.

6 Related Work

We discuss research work related to browser extension security in the following categories.

A. Study of security and privacy in browser extensions. Martin Jr. et al [15] investigate privacy issues in IE 6 extensions, where they found some extensions monitoring users' behaviors or intercepting and disclosing SSL-protected traffic. A more recent study [16] investigates privileges in 25 Firefox extensions that are necessary for extensions' functionalities, and found that only 3 out of the 25 extension would actually require the most powerful capabilities of the privileges Firefox extensions all have, violating the least privilege principle. Compared to them, our work focuses on dangerous behaviors in Firefox extensions, and our study was conducted with an automatic testing tool.

B. Securing browser extensions. Based on the weaknesses of the old Firefox extension system, Barth et el. [3] propose a new browser extension system for Google Chrome, which is designed with least privilege, privilege separation and strong isolation. However, Liu et al. [4] find that the original design of the Chrome extension framework had still violated the principles of least privileges and privilege separation, and they propose improvements to it with micro-privilege management and fine-grained access control to DOM elements. Similarly, Mozilla develops a new extension framework Jetpack [5] to make it easier to develop more secure browser extensions. The basic idea of the new framework is to isolate extensions into a collection of modules, each of which is expected to follow the principle of least authority (POLA). Karim et al. [17] perform a static analysis on 77 core Jetpack framework modules and 359 extensions, where they find 12 and 24 capability leaks, respectively. Although we propose potential improvement to mitigate indirect threats to web sessions from browser extensions, we do not aim for a redesign of browser extension frameworks in this paper. Our focus is on the existing behaviors of Firefox extensions that are currently used by users today.

Sabre [18] is a system that analyzes the browser extensions by monitoring in-browser information-flow. It produces an alert when an extension is found to access some sensitive information in an unsafe way. A similar approach [19] is also used to detect attacks against privilege escalation vulnerabilities in Firefox extensions. Another recent work by Ter Louw et [1,2] discusses techniques for runtime monitoring of extension behaviors. They try to reduce the threats posed by malicious or buggy Firefox extensions by controlling an extension's access to XPCOM. Similar to this work, to protect users from spy add-ons, SpyShield [20] uses an access-control proxy to control communications between untrusted add-ons and their host application. Some static approaches are also proposed to detect vulnerabilities in JavaScript-based widgets. GATEKEEPER [21] is a static approach for enforcing security and reliability policies for JavaScript programs.

VEX [6] is a framework for identifying potential security vulnerabilities in browser extensions by static information-flow analysis.

Compared to the work above, our focus in this paper is on dangerous behaviors in existing Firefox extensions, rather than detecting vulnerabilities or attacks.

C. Automatic event simulation. Some existing systems for web application testing, such as Selenium [22], Watir [23] and [24] can simulate mouse and keyboard events on webpages. However, they cannot trigger events in Firefox extensions. In this work, we provide additional support for Firefox's internal event simulation.

7 Conclusion

In this paper, we present a large-scale study on dangerous behaviors in Firefox extensions. We focus on investigating the indirect threats posed by extensions by tracking the behaviors of new web content injected by extensions. Through an automatic testing system equipped with an injected object tracker, we tested over 2,465 extensions in 13 different categories from the Mozilla Addon repository. We found that there are 108 extensions in total injecting various contents (such as scripts, iframes, images and so on) into web pages. Although these cases are not malicious, they can be abused to tamper with web sessions and should be tackled with special care. To mitigate this kind of threats, we discuss a solution to apply principal tracking to constrain the behaviors of injected content by extensions, while still maintaining usability for legitimate behaviors.

Acknowledgments. We are grateful to the anonymous reviewers for their insightful comments and suggestions. This research was supported in part by National Natural Science Foundation of China (No.90718023, 91118003), Tianjin Research Program of Application Foundation and Advanced Technology (No.10JCZDJC15700), "985" funds of Tianjin University, and National University of Singapore under NUS Young Investigator Award R-252-000-378-101.

References

1. Ter Louw, M., Lim, J.S., Venkatakrishnan, V.N.: Enhancing web browser security against malware extensions. Journal in Computer Virology 4, 179–195 (2008)
2. Ter Louw, M., Lim, J.S., Venkatakrishnan, V.N.: Extensible Web Browser Security. In: Hämmerli, B.M., Sommer, R. (eds.) DIMVA 2007. LNCS, vol. 4579, pp. 1–19. Springer, Heidelberg (2007)
3. Barth, A., Felt, A.P., Saxena, P., Boodman, A.: Protecting browsers from extension vulnerabilities. In: Network and Distributed System Security Symposium (2010)
4. Liu, L., Zhang, X., Yan, G., Chen, S.: Chrome extensions: Threat analysis and countermeasures. In: Proceeding of the Network and Distributed System Security Symposium, NDSS 2012 (2012)
5. Mozilla. Jetpack, https://wiki.mozilla.org/Jetpack
6. Bandhakavi, S., King, S.T., Madhusudan, P., Winslett, M.: Vex: vetting browser extensions for security vulnerabilities. In: Proceedings of the 19th USENIX Conference on Security, Berkeley, CA, USA, p. 22 (2010)

7. Mozilla add-ons, https://addons.mozilla.org/
8. xdotool, http://www.semicomplete.com/projects/xdotool/
9. Spidermonkey, https://developer.mozilla.org/en/SpiderMonkey
10. htmlcxx - HTML and CSS APIs for C++, http://htmlcxx.sourceforge.net/
11. libcurl - the multiprotocol file transfer library, http://curl.haxx.se/libcurl/
12. Security severity ratings, https://wiki.mozilla.org/Security_Severity_Ratings
13. Severity guidelines for security issues, http://dev.chromium.org/developers/severity-guidelines
14. Add-on review guide, https://wiki.mozilla.org/AMO:Editors/EditorGuide/AddonReviews
15. Martin Jr., D.M., Smith, R.M., Brittain, M., Fetch, I., Wu, H.: The privacy practices of web browser extensions. Communications of the ACM (2001)
16. Felt, A.P.: A survey of firefox extension API use. Technical report, University of California at Berkeley (2009)
17. Karim, R., Dhawan, M., Ganapathy, V., Shan, C.-C.: An Analysis of the Mozilla Jetpack Extension Framework. In: Noble, J. (ed.) ECOOP 2012. LNCS, vol. 7313, pp. 333–355. Springer, Heidelberg (2012)
18. Dhawan, M., Ganapathy, V.: Analyzing information flow in javascript-based browser extensions. In: Computer Security Applications Conference, ACSAC (2009)
19. Djeric, V., Goel, A.: Securing script-based extensibility in web browsers. In: Proceedings of the 19th USENIX Conference on Security, USENIX Security 2010, p. 23. USENIX Association, Berkeley (2010)
20. Li, Z., Wang, X.-F., Choi, J.Y.: SpyShield: Preserving Privacy from Spy Add-Ons. In: Kruegel, C., Lippmann, R., Clark, A. (eds.) RAID 2007. LNCS, vol. 4637, pp. 296–316. Springer, Heidelberg (2007)
21. Guarnieri, S., Livshits, B.: Gatekeeper: Mostly static enforcement of security and reliability policies for javascript code. In: USENIX Security Symposium (2009)
22. Selenium web application testing system, http://seleniumhq.org/
23. Watir automated webbrowsers, http://wtr.rubyforge.org/
24. Balduzzi, M., Egele, M., Kirda, E., Balzarotti, D., Kruegel, C.: A solution for the automated detection of clickjacking attacks. In: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 135–144 (2010)

Collaboration-Preserving Authenticated Encryption for Operational Transformation Systems

Michael Clear, Karl Reid, Desmond Ennis, Arthur Hughes, and Hitesh Tewari

School of Computer Science and Statistics,
Trinity College Dublin

Abstract. We present a flexible approach for achieving user-controlled privacy and integrity of documents that are collaboratively authored within web-based document-editing applications. In this setting, the goal is to provide security without modifying the web application's client-side or server-side components. Instead, communication between both components is transparently intercepted and processed (if necessary) by means of a local proxy or browser plugin. We improve upon existing solutions by securely preserving real-time collaboration for encrypted documents and facilitating self-containment of the metadata (an overhead of encryption) within the same document. An architectural generalization is also presented that permits generic transformations and fine-grained access control. Security is assessed with respect to several threat models, and performance is evaluated alongside other approaches.

1 Introduction

Privacy for data stored in the cloud is a growing concern. Although the ubiquity of TLS for securing the transport layer ensures confidentiality and integrity for data in-transit, reservations arise about the security of data persisted in the cloud. This is especially pronounced for Software as a Service (SaaS) offerings whereby an organization's security policy is incompatible with that adopted by the service provider. Since it is desirable to avail of the convenient functionality and resources supported by the provider, these concerns have often been incautiously ignored.

In recent years, there has been significant research in the area of computing delegation and data delegation as cloud computing has emerged as a predominant paradigm. Advances in fully homomorphic encryption since Gentry's 2009 breakthrough [1] such as [2], along with practical predicate encryption such as [3] proceed towards the realizing the goal of privacy in outsourced computations. However, in these settings, control is assumed over the remote server.

Some SaaS applications can retain their functionality while simultaneously allowing end-to-end data privacy, *without* performing computations homomorphically or relying on cryptographic techniques that require changes to be made to the server. Other applications can retain the bulk of their features under similar conditions. In fact, if control were assumed over customizing the server-side software to accommodate end-to-end data privacy and integrity, the task would be less challenging. By ‘end-to-end’, we mean that the client has exclusive access to any keys required to authenticate or decrypt the application data.

Consider a SaaS-orientated solution comprising two components - a user-facing client and a remote server. For several web applications such as Google Docs [4], there is a tight-coupling between both. Indeed, the client is not guaranteed to be static, and there may or may not be a published API between client and server. In the absence of a published API, the interface i.e. *protocol* must be reverse engineered.

There are several web-based real-time collaborative-editing systems based on the technique of Operational Transformation (OT). In essence, OT manages consistency of shared resources that are acted on concurrently by multiple collaborators. As collaborators relay their edit *operations* to each other (usually via a coordinator), these edits must be correctly *transformed* relative to other concurrent edits to preserve the consistency of the shared resource. Web-based OT applications include Google Docs [4], Etherpad [5] and Apache Wave [6].

We focus on Google Docs in this work for a multitude of reasons. Firstly, it is popular and free, allowing us to readily obtain empirical data from co-operative users. Secondly, alternative approaches for achieving encryption have been explored in the literature, (most notably [7,8]), thus allowing a comparative analysis to be conducted.

Achieving security for services that are operated by untrusted parties, and doing so without control over the server is a difficult challenge. Availing of security functions such as encryption in this scenario often disrupts core functionality of the service. For example in the case of collaborative editing, features such as spell checking and searching that are performed on the server-side are spoiled. Nevertheless, it may be satisfactory for users that the predominant functionality of such applications, namely easy sharing of documents, real-time collaborative editing and granular version control, remain unimpaired by added security functions. In addition, many users may only require encryption for a small portion of the document, for example, sensitive fields such as social security numbers.

The contributions of this paper are as follows:

1. A flexible approach is presented to achieve transparent application-layer security (integrity and authentication) for untrusted OT systems such as Google Docs. Unlike prior work, our solution does not use an out-of-band resource to store auxiliary information nor does it suffer from synchronization issues regarding its consistency. Instead, the presented approach inextricably couples the auxiliary information into the document representation in a manner that allows both efficient *incremental* encryption and preservation of collaborative functionality.
2. A model is formulated that expresses this approach generically for a certain class of OT systems. As a result, the precise preconditions and expected behavioral properties can be established.

3. A proof-of-concept implementation for the Firefox web browser is discussed and is shown to compare favorably against existing implementations. Note that the complex issue of key sharing is not addressed in this particular work.[1]

1.1 Related Work

The target problem exhibits overlap with several areas; in particular incremental cryptography [9] and secure timestamp systems [10].

This work was carried out independent to the contributions of Feldman et al. [11] and thus there are several similarities, although both works have a different focus. The major difference is that we concentrate on providing security for *existing, live* OT systems such as Google Docs whereas Feldman et al. devise a new OT-based framework to be deployed on an untrusted server. Furthermore, they consider protection against a variety of active attacks, such as forking, which we don't address in this paper.

Adkinson-Orellana et al. [8] present a security layer to provide encryption for Google Docs by means of a Firefox extension that mediates traffic to/from the Google Docs server in order to transparently encrypt/decrypt it. Since semantically secure encryption depends on a random component, it is necessary to include additional information with a ciphertext. The authors of [8] propose to employ a hidden document serving as an index table where this information can be stored. This approach introduces a host of synchronization issues - collaborating users must somehow synchronize updates to the shared index in order to maintain consistency. Furthermore, the auxiliary information is externalized; that is, stored out-of-band.

An improved solution appears in a more recent work by Huang and Evans [7]. The authors also develop a Firefox extension to transparently mediate Google Docs traffic. Their solution is based on incremental encryption, introduced by Bellare, Goldreich and Goldwasser [9]. In particular, they use a block cipher mode of operation known as RPC [12] for incremental *authenticated* encryption. A derivative of SkipList data structure is employed to accomodate efficient incremental encryption, and it supports complete document integrity. Its main drawback is synchronization limitations. Synchronization is difficult because it relies on a block table (the chaining of all the blocks together), which may give rise to conflicts if multiple users are editing simultaneously, although the authors cite other technical reasons. In the version of Google Docs targeted in this work, the size of a delta i.e. the number of characters modified in a single edit must be preserved by the browser extension in order to maintain consistency, certainly in the presence of active collaborators. As a result, the approach taken by Huang

[1] Nevertheless, key sharing has been incorporated into our proof-of-concept implementation by intercepting the requests that specifically relate to document sharing, and then enveloping the symmetric key using the target recipient's public key and depositing the envelope in a remote centralized data store. PKI is employed in the present proof-of-concept implementation, and ECC is used for efficiency.

and Evans where a block table is stored in the document content is rendered infeasible by our goal to support fuller collaboration. This reason along with our aim to avoid storing auxiliary data "out-of-band" motivated the revision-based approach adopted here.

1.2 Notation

In this paper, we index strings, sequences and vectors starting from 0. Given a string w in Σ^* for some alphabet Σ, the j-th character is denoted by $w[j]$, and a substring of w with starting index i and ending index (exclusive) j is denoted by $w[i, j]$. The symbol $\|$ denotes string concatenation. As a notational convenience, the length of all strings and sequences is given by the $|\cdot|$ operator. We denote the empty string by ϵ.

For a probability distribution $\mathcal{D}_S$ defined on a set S, we denote by $x \overset{\$}{\leftarrow} \mathcal{D}_S$ the fact that x is sampled from S according to $\mathcal{D}_S$. For any set T, the notation $t \overset{\$}{\leftarrow} T$ indicates that t is uniformly sampled from T.

All vectors and sequences used throughout this paper are denoted in boldface. The set $\mathbb{N}_0$ is considered to be the set of natural numbers with inclusion of 0.

A function is said to be *negligible* (in some parameter n) if for all polynomial functions $g(x)$: $f(n) < 1/g(n)$ for sufficiently large n.

1.3 Paper Organization

This paper is organized as follows: firstly in Section 2, an abstraction of an Operational Transformation system is formulated which models a variety of real-world collaborative editing systems. Our methodology for collaboration-preserving encryption and authentication of document content is then expressed within this model. A proof-of-concept implementation then follows for Google Docs in Section 3 where we instantiate our approach with cryptographic primitives. Performance is evaluated in Section 4. Finally, security properties are analyzed in Section 5.

2 Abstract Collaborative Editing System

An abstraction of a simplified collaborative editing system is presented in this section. The model described here will be referred to as the Simple Transactional Collaborative Editing (STCE) model. We intentionally omit the operational transformation aspects relating to consistency and concurrency control, which have been the subject of considerable research so far [13,14,15], and are outside the scope of this work. Accordingly, these properties are taken as assumptions.

The system is coordinated by a centralized server, denoted by C, referred to as the *coordinator*. The abstraction is simplified to center around a fixed document resource, which therefore obviates the need to reference a particular document. For the document resource implicit in the description, we assume that there are n connected collaborating *sites* labelled $s_1, \ldots, s_n$. The interface exposed to each site by C is outlined in Figure 1.

Service 1 EstablishSession
```
1: siteId ← $n$
2: $n \leftarrow n + 1$
3: Initialize position for site siteId to 0
4: Initialize revision number for site siteId to $|\boldsymbol{H}|$
5: return siteId
```

Service 2 UpdateCursorPosition
```
Input: siteId : $\mathbb{N}_0$
Input: $p : N_0$
Require: $p \leq |\delta_{\boldsymbol{H}}(\epsilon)|$
1: Update cursor position for site siteId to $p$
```

Service 3 Edit
```
Input: siteId : $\mathbb{N}_0$
Input: mutations : $M^+$
Input: revisionNum : $\mathbb{N}_0$
Require: $0 \leq$ siteId $< n$
Require: $\boldsymbol{H}[0, \text{revisionNum}] \parallel \boldsymbol{r} \in \mathbb{H}$    ▷ where
    $\boldsymbol{r} := (\text{position}_{\text{siteId}}, \text{mutations})$
1: $\boldsymbol{r}' \leftarrow$ result of OT processing on $\boldsymbol{r}$ with respect to
    recent revisions (since last revision committed by the
    site)
2: newRevisionNum $\leftarrow |\boldsymbol{H}|$
3: $\boldsymbol{H} \leftarrow \boldsymbol{H} \parallel \boldsymbol{r}'$
4: return newRevisionNum
```

Service 4 LoadRevisions
```
Input: startRevNum : $\mathbb{N}_0$
Input: endRevNum : $\mathbb{N}_0$
Require: $0 \leq$ startRevNum $\leq$ endRevNum $\leq |\boldsymbol{H}|$
1: return $\boldsymbol{H}$[startRevNum, endRevNum]
```

Service 5 GetHistorySize
```
1: return $|\boldsymbol{H}|$
```

Fig. 1. Interface to the Coordinator C

Let Σ be an alphabet. A document's content is typed as a string over this alphabet - an element of Σ^*. Each site s_i maintains a local state q_i, which is a pair (c, d), where $c \in \mathbb{N}_0$ is the site's cursor position within the document content and $d \in \Sigma^*$ is its local copy of the document content. Thus we define the state space $Q \triangleq \{(p, w) \in \mathbb{N}_0 \times \Sigma^* : 0 \leq p \leq |w|\}$. Upon establishing a session with C, a site initializes its cursor position to 0, and any changes to it must be synchronized with C. Therefore, C keeps track of the current cursor positions of all active collaborators facilitating control over synchronization and mutual exclusion.

2.1 Primitive Operations

There are a finite number of prescribed primitive operations that can be performed on a state. Such a primitive operation is termed a *mutation*. It is sufficient to describe two such types in this simplified system, namely for insertion and deletion. Define a set of symbolic tags $T \triangleq \{\tau_{\text{ins}}, \tau_{\text{del}}\}$. Now we define the set of insertion mutations $M_{\tau_{\text{ins}}} \triangleq \{(\tau_{\text{ins}}, v) : v \in \Sigma^*\}$ and the set of deletion mutations $M_{\tau_{\text{del}}} \triangleq \{(\tau_{\text{del}}, k) : k \in \mathbb{Z}\}$. Finally the total set of mutations M is the union $M_{\tau_{\text{ins}}} \cup M_{\tau_{\text{del}}}$.

Some mutations may not be compatible with a particular state. An example is deletion where the number of characters specified for deletion may exceed the range of the document string. Although removing as many characters as possible is satisfactory, an alternative that is useful later for other purposes is to augment Q with a 'failure' state. Thus, we define $\hat{Q} \triangleq Q \cup \{\perp\}$. A mutation description μ (i.e. a pair in M) induces a state transformation function $\delta_\mu : \hat{Q} \to \hat{Q}$. We have that

$$\delta_{(\tau_{\text{ins}}, v)}(q) = \begin{cases} (p + |v|, w[0, p] \parallel v \parallel w[p, |w|]) & \text{if } q := (p, w) \in Q \\ \perp & \text{if } q = \perp \end{cases}$$

and

$$\delta_{(\tau_{\mathsf{del}},k)}(q) = \begin{cases} (p+k, w[0,p+k] \parallel w[p,|w|]) & \text{if } q := (p,w) \in Q, k < 0 \leq p+k < |w| \\ (p, w[0,p] \parallel w[p+k,|w|]) & \text{if } q := (p,w) \in Q, 0 \leq k \leq p+k \leq |w| \\ \bot & \text{otherwise} \end{cases}$$

Informally, the former splices a new string into an existing string while the latter has the effect of removing a portion of the string before or after a specified position (it removes as many characters as possible of the specified number of characters). These transformation functions transform states, updating both cursor position and document content.

2.2 Composite Operations

A mutation is an atomic primitive operation. In order to specify an ordered sequence of mutations to be treated as an atomic unit, the system supports composite operations which behave as transactions. Any valid edit request produced by a site entails a non-empty ordered sequence of mutations termed a *revision* that is to be applied atomically. A revision also specifies the cursor position in the content string at which the mutations are to be applied. Define the set of revisions as $\mathcal{R} \triangleq \mathbb{N}_0 \times M^+$, We distinguish a revision $\boldsymbol{r} \in \mathcal{R}$ by boldface. If $\boldsymbol{r}$ is valid, the coordinator C stores $\boldsymbol{r}$ in its chronology, which is discussed further later. It also relays $\boldsymbol{r}$ to the $n-1$ other sites and takes care of the appropriate operational transformations.

Let $\boldsymbol{r} := (p, \mu_0, \ldots, \mu_{\ell-1})$ be a revision. The transformation function $\delta_{\boldsymbol{r}} : \Sigma^* \cup \{\bot\} \to \Sigma^* \cup \{\bot\}$ induced by $\boldsymbol{r}$ is given by:

$$\delta_{\boldsymbol{r}:=(p,\mu_0,\ldots,\mu_{\ell-1})}(s) = \begin{cases} \mathsf{proj}_1((\delta_{\mu_{\ell-1}} \circ \cdots \circ \delta_{\mu_0})(p,w)) & \text{if } s := (p,w) \in Q \\ \bot & \text{otherwise} \end{cases} \quad (1)$$

where proj_1 is the projection map that returns the second (index 1) component of its vector argument.

We say two revisions $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ are equivalent, written $\boldsymbol{r_1} \sim \boldsymbol{r_2}$, iff $\delta_{\boldsymbol{r_1}} = \delta_{\boldsymbol{r_2}}$. It is easy to show that for every equivalence class in $\mathcal{R}/\sim$, there is a representative $\boldsymbol{r} \in \mathcal{R}$ of the form $(\tau_{\mathsf{del}}, k), (\tau_{\mathsf{ins}}, v))$ where $k \in \mathbb{Z}$ and $v \in \Sigma^*$. Such an $\boldsymbol{r}$ is said to be in normal form, and we make use of this in the security game in Section 2.3. This prompts the question: should a site normalize a revision prior to sending it to C? Alternatively, should C normalize the revision before recording it in its chronology? Clearly this depends on the requirements and feature set of the system. Each mutation corresponds to an editing event and these events may be bundled together periodically to form atomic revisions. Thus in order to facilitate an undo feature with arbitrary granularity, preserving the original sequence of events is necessary. Another motivation to preserve the original sequence of events in a persistent chronology is to allow for fine-grained logging. Normalization would lose any such information. Thus, in the system described here, we assume that a revision submitted by a site remains *structurally invariant* in persistent storage provided by C.

For each document, the coordinator C maintains a history $\boldsymbol{H} \subset \mathcal{R}^*$ of all revisions ordered by the time of commitment. It exposes a facility to make range queries on $\boldsymbol{H}$. We define $\delta_{\boldsymbol{H}} = \delta_{\boldsymbol{r_{m-1}}} \circ \cdots \circ \delta_{\boldsymbol{r_0}}$ where $\boldsymbol{H} = \boldsymbol{r_0}, \ldots, \boldsymbol{r_{m-1}}$. A history is a sequence of *valid* revisions i.e. reconstitution of the document string does not yield the 'failure' state. Therefore, we define the history space as $\mathbb{H} \triangleq \{\boldsymbol{H} \in \mathcal{R}^* : \delta_{\boldsymbol{H}}(\epsilon) \neq \perp\}$ where ϵ is the empty string.

2.3 Transformations

Consider useful types of pre-processing and post-processing that may be performed on revisions stored in a history. To capture the restricted transformations that can be applied to revisions in the system, we define a stateful (dependent on the history preceding a given revision) and reversible transformation as follows:

Definition 1. *Define* $\mathsf{HR} \triangleq \{(\boldsymbol{H}, \boldsymbol{r}) \in \mathbb{H} \times \mathcal{R} : \boldsymbol{H} \parallel \boldsymbol{r} \in \mathbb{H}\}$. *Define* $\hat{\mathcal{R}} \triangleq \mathcal{R} \cup \{\perp\}$. *A revision transformation is a pair* $(\mathsf{Map}, \mathsf{Invert})$, *where* $\mathsf{Map} \subseteq \mathsf{HR} \times \hat{\mathcal{R}}$ *is a relation and* $\mathsf{Invert} : \mathsf{HR} \to \hat{\mathcal{R}}$ *is a function, satisfying*
$\forall (\boldsymbol{H}, \boldsymbol{r}, \boldsymbol{r'}) \in \mathsf{Map} : \boldsymbol{r'} \in \mathcal{R}$, $\forall (\boldsymbol{I}, \boldsymbol{s}, \boldsymbol{s'}) \in (\mathbb{H} \times \mathcal{R} \times \hat{\mathcal{R}}) \setminus \mathsf{Map}$:

1. $\boldsymbol{H} \parallel \boldsymbol{r'} \in \mathbb{H}$.
2. $\mathsf{proj}_0(\boldsymbol{r}) = \mathsf{proj}_0(\boldsymbol{r'})$
3. $\mathsf{proj}_0 \circ \delta_{\boldsymbol{r}} = \mathsf{proj}_0 \circ \delta_{\boldsymbol{r'}}$
4. $\boldsymbol{r} \sim \mathsf{Invert}(\boldsymbol{H}, \boldsymbol{r'})$
5. $\mathsf{Invert}(\boldsymbol{I}, \boldsymbol{s'}) = \perp$

Intuitively, properties 1-3 capture the requirement that Map preserve both the validity of a revision, the cursor position it acts at and the change in cursor position it effects, whereas the final two properties constrain the transformation to be reversible. We will respectively view Map and Invert as a randomized algorithm and a deterministic algorithm. To apply such algorithms to a history $\boldsymbol{H}$, we define

$$\mathsf{InvertH}(\mathsf{Invert}, \boldsymbol{H}) = \begin{cases} \epsilon & \text{if } \boldsymbol{H} = \epsilon \\ \boldsymbol{r'} \parallel \mathsf{InvertH}(\mathsf{Invert}, \boldsymbol{H'}) & \text{if } \boldsymbol{H} = \boldsymbol{H'} \parallel \boldsymbol{r} : \boldsymbol{r} \in \mathcal{R} \text{ and } \boldsymbol{r'} \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

where $\boldsymbol{r'} = \mathsf{Invert}(\boldsymbol{H'}, \boldsymbol{r})$ and

$$\mathsf{MapH}(\mathsf{Map}, \boldsymbol{H}) = \begin{cases} \epsilon & \text{if } \boldsymbol{H} = \epsilon \\ \boldsymbol{H''} \parallel \boldsymbol{r'} & \text{if } \boldsymbol{H} = \boldsymbol{H'} \parallel \boldsymbol{r} : \boldsymbol{r} \in \mathcal{R} \text{ and } \boldsymbol{r'} \neq \perp \text{ and } \boldsymbol{r'} \neq \perp \\ \perp & \text{otherwise} \end{cases}$$

where $\boldsymbol{H''} = \mathsf{MapH}(\mathsf{Map}, \boldsymbol{H'})$ and $\boldsymbol{r'} = \mathsf{Map}(\boldsymbol{H''}, \boldsymbol{r})$

Definition 1 generalizes several transformation functionalities including string isomorphisms. An important characteristic of a revision transformation is that it may rely on the structure of a revision to accomplish invertibility i.e. it may reorder mutations or add 'dummy mutations' to store auxiliary information. For example: suppose the desired transformation is information lossy, which implies that additional information must be stored to reverse it. Due to the property of structural invariance of revisions that is provided by the system, it is possible to map a revision to one with an appended insertion-deletion pair (an insertion of a string immediately followed by its deletion). Such a null sequence has no effect on the content but the information it encodes serves to inform the invertibility of the transformation. Building on the notion established by Definition 1, we can establish a family of transformations with common functionality parameterized by some key or index. Thus given the spaces K_1 and K_2, and a subset of their product $K \subseteq K_1 \times K_2$, a family of transformations $\mathcal{T}_K$ is defined as the set $\{(\mathsf{Map}_{k_1}, \mathsf{Invert}_{k_2}) : (k_1, k_2) \in K\}$.

Encryption. A useful family of transformations is one whose function is to encrypt the content of a document. This is, for example, readily instantiated by a symmetric cipher. In this case, we have $K_1 = K_2$. For confidentiality only (or in addition to revision integrity as described in Section 5), the Map algorithm may be 'stateless' and act independently of a history argument. A generalized IND-CPA game for the confidentiality-only setting is sketched as follows:

Fix a security parameter λ. The challenger $\mathcal{C}$ generates some $(k_1, k_2) \in K$ from λ. The attacker $\mathcal{A}$ is granted oracle access to $\mathsf{Map}_{k_1}(\cdot)$ to make queries. $\mathcal{A}$ produces two distinct revisions $\boldsymbol{r_0}$ and $\boldsymbol{r_1}$ with the property that $\mathsf{proj}_0 \circ \delta_{\boldsymbol{r}_0} = \mathsf{proj}_0 \circ \delta_{\boldsymbol{r}_1}$, and sends them to $\mathcal{C}$. Accordingly, $\mathcal{C}$ samples a bit b uniformly at random, and returns $\boldsymbol{r_b}' = \mathsf{Map}_{k_1}(\mathsf{Normalize}(\boldsymbol{r_b}))$ to $\mathcal{A}$. The selected revision is normalized prior to application of Map_{k_1} to prevent trivial distinguishability. Finally, $\mathcal{A}$ outputs a bit b', and as usual, it is said to win the game if $b = b'$. As is standard, if the advantage of any polynomially-bounded $\mathcal{A}$ is negligible in λ, the system is IND-CPA-secure. Of course it is easy to extend this game to capture the IND-CCA1 and IND-CCA2 security properties by exposing an oracle to $\mathsf{Invert}_{k_2}(\cdot)$ with appropriate restrictions on queries.

For the remainder of this paper, we omit the first argument (history) from Map and Invert since we only consider per-revision integrity in this paper due to space constraints. However, complete integrity is not difficult to achieve and is discussed in Section 5.

2.4 Replay and Reconstitution Algorithms

Suppose we have a family of transformations $\mathcal{T}_K$ for some key space $K \subseteq K_1 \times K_2$. Let $(k_1, k_2) \in K$. Suppose that all sites apply $\mathsf{Map}_{\mathsf{k}_1}$ to each revision before sending it to C. A key goal is to minimize the number of applications of $\mathsf{Invert}_{\mathsf{k}_2}$ that must be applied to 'undo' the transformation for each revision. An important observation is that the number of such applications need only depend on the length of the reconstituted document string $\delta_{\boldsymbol{H}}(\epsilon)$ where $\boldsymbol{H}$ is a history that has not been transformed. Thus, the time complexity is linear in $|\delta_{\boldsymbol{H}}(\epsilon)|$ as opposed to linear in $|\boldsymbol{H}|$. Clearly, this is of considerable practical value since a document may have a long revision history whose early revisions no longer contribute to the present document content.

Firstly, we need to build a list of operations that must be executed in sequence to reconstitute the document string - see Algorithm 1. It is necessary to keep a reference to the 'parent' revision of a particular operation since it is such revisions that the $\mathsf{Invert}_{\mathsf{k}_2}$ algorithm is applied to in the end. Now any string that is inserted into the document as a result of a single revision may be broken up into many contiguous portions due to subsequent deletions. We will refer to these contiguous portions as *pieces* and each one points to its parent revision. A piece may be described by an offset and length with respect to the inserted string - this also obviates the need to allocate memory to store multiple substrings.

A self-balancing search tree of pieces ordered by document position is constructed in the replay algorithm. Roughly speaking, an insertion is handled by finding the piece that precedes it (if it exists) in the tree and splitting that piece if its range overlaps with the position of the string to be inserted. We can avoid shifting the positions of the pieces that follow it in the tree by instead translating the positions of subsequent operations (subtracting the number of characters inserted in the case of insertion). This is achieved by maintaining a 'delta' integer to be added to all operations. Deletion is handled by removing all pieces whose range completely overlaps with the deleted range, and appropriately splitting any pieces whose range partially overlaps; the delta is increased by the number of characters removed.

The replay algorithm (Algorithm 2) builds up a search tree of pieces. Recall that a piece specifies a portion of a string inserted in a revision in addition to referencing that revision. Subsequently, in the 'reconstitute' algorithm (Algorithm 4), the document's content is assembled by performing an in-order traversal of the tree, and for each piece: (1) applying $\mathsf{Invert}_{\mathsf{k}_2}$ to its associated revision; (2) normalizing the result; and (3) projecting the string in its insertion mutation in accordance with the offset and length attributes of the piece. An outline of the these algorithms can be found in Figure 3.

The behavior of the replay algorithm for insertions is illustrated in Figure 2. In the figure, two simple revisions are considered that insert text into the document. In particular, this signifies how a piece is fragmented through successive replays while retaining a pointer to the original revision structure that generated it. In this example, no transformation has been applied to the text. Therefore the correct document content after revisions r1 and r2 have been applied may be reconstituted by performing an in-order traversal of the tree.

3 Concrete Collaborative Editing System

The STCE model presented in the previous section is compatible with the supported features of Google Docs. Some of these features are directly exposed via standard APIs published by Google, whereas the availability of others may not be guaranteed at present - the latter were discovered by reverse engineering and experimentation. In brief, in order for our approach to work, it is necessary that such systems support both structural invariance for revisions and arbitrary range queries on the history. It will be shown later however that the latter requirement

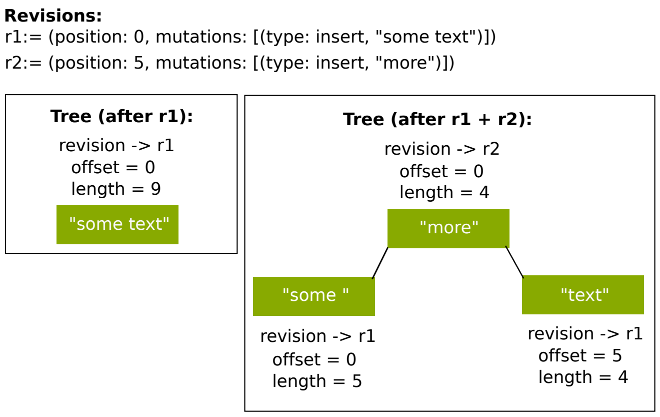


Fig. 2. Formation of the tree as revisions are replayed

Algorithm 1 MakeOpList

```
Input: $\boldsymbol{H} : \mathbb{H}$
 1: $m \leftarrow |\boldsymbol{H}|$
 2: $L \leftarrow$ new list
 3: $i \leftarrow 0$
 4: for $0 \leq i < m$ do
 5:     Parse $\boldsymbol{r} := (p, \mathsf{mutations}) \leftarrow \boldsymbol{H}[i]$
 6:     $\boldsymbol{r'} \leftarrow \mathsf{Normalize}(\boldsymbol{r})$
 7:     Parse $(\tau_{\mathsf{del}}, k), (\tau_{\mathsf{ins}}, v) \leftarrow \boldsymbol{r'}$
 8:     Append $(p, \boldsymbol{r}, (\tau_{\mathsf{del}}, k))$ to $L$
 9:     Append $(\min(p, p+k), \boldsymbol{r}, (\tau_{\mathsf{ins}}, v))$ to $L$
10: end for
11: return $L$
```

Algorithm 2 Replay

```
Input: $L$ ▷ List of operations outputted by MakeOpList.
 1: $T \leftarrow$ new search tree
 2: $\Delta_p \leftarrow 0$
 3: for all $(p, \boldsymbol{r}, \mu) \in L$ do
 4:     $p \leftarrow p + \Delta_p$
 5:     $\tau \leftarrow \mathsf{proj}_0(\mu)$
 6:     if $\tau = \tau_{\mathsf{ins}}$ then
 7:         $\Delta_p \leftarrow \Delta_p - \mathsf{ReplayInsertion}(T, p, \boldsymbol{r}, \mu)$
 8:     else if $\tau = \tau_{\mathsf{del}}$ then
 9:         $\Delta_p \leftarrow \Delta_p + \mathsf{ReplayDeletion}(T, p, \boldsymbol{r}, \mu)$
        ▷ ReplayDeletion entails more cases than
        ReplayInsertions but in many ways is similar. It
        is omitted in this version of the paper due to space
        constraints.
10:     end if
11: end for
12: return $T$
```

Algorithm 3 ReplayInsertion

```
Input: $T$
Input: $p$
Input: $\boldsymbol{r}$
Input: $\mu := (\tau_{\mathsf{ins}}, v)$
 1: $\mathsf{Pre} \leftarrow \{x \in T \mid \mathsf{x.position} < p\}$
 2: if $\mathsf{Pre} \neq \emptyset$ then
 3:     $\mathsf{prev} \leftarrow \max(\mathsf{Pre})$ (piece with max position)
 4:     if $\mathsf{prev.position} + \mathsf{prev.length} > p$ then
 5:         $\phi_1, \phi_2 \leftarrow \mathsf{Split}(\mathsf{prev}, \mathsf{p}, |v|)$ ▷
    Split divides a piece about a position into two sub-
    pieces. The position of the second piece is shifted by
    the third argument.
 6:         Remove $\mathsf{prev}$ from $T$
 7:         Add $\phi_1$ to $T$
 8:         Add $\phi_2$ to $T$
 9:     end if
10: end if
11: Add new piece to $T$ with position:=$p$, length:=$|v|$,
    offset := 0, revision:=$\boldsymbol{r}$
12: return $|v|$
```

Algorithm 4 Reconstitute

```
Input: $T$ ▷ Tree of pieces outputted by Replay
Input: $f : \mathcal{R} \rightarrow \mathcal{R} \cup \{\perp\}$
 1: $w \leftarrow \epsilon$
 2: $U \leftarrow$ list of pieces from in-order traversal of $T$
 3: for all $u \in U$ do
 4:     $\boldsymbol{r'} \leftarrow f(u.\mathsf{revision})$
 5:     if $\boldsymbol{r'} = \perp$ then
 6:         return $\perp$
 7:     end if
 8:     $v \leftarrow \delta_{\boldsymbol{r'}}(\epsilon)$
 9:     if $v = \perp$ then
10:         return $\perp$
11:     end if
12:     $w \leftarrow w \,\|\, v[u.\mathsf{offset}, u.\mathsf{offset} + u.\mathsf{length}]$
13: end for
14: return $w$
```

Fig. 3. Replay Algorithms, abstractly, as introduced in Section 2.4

can be somewhat relaxed. The former is attractive because it facilitates unrestricted 'undo' (a feature supported for example by the open source editor Etherpad [5]) along with granular logging.

3.1 Correspondence between the STCE Model and Google Docs

Google Docs fits directly into our STCE model. Its server acts as the coordinator C and each collaborator session can be viewed as a site. Google Docs is a web application with a Javascript-based client-side front-end, referred to in this paper as the *client*. Edits to the document or movements of the cursor generate events that result in *mutate* requests being periodically sent to the server in an asynchronous manner; no such requests are sent during spans of inactivity. However, when events are being generated, the period between network requests is low enough to give rise to small revisions on average. An empirical analysis, examined later, reveals that on the order of 1 or 2 inserted characters are embedded in a typical update. This alone has security implications.

The client also receives periodic updates on the edits applied to the document in concurrent sessions, which is captured (only synchronously) in STCE by periodic invocation of C.GetHistorySize followed by invocation of C.LoadRevisions. An outline of the correspondence between STCE and Google Docs[2]. is given in Table 1.

Table 1. Correspondence between the interface to the coordinator C in our STCE Model and Google Docs

STCE	Analog in Google Docs
C.EstablishSession	HTTPS GET *edit* request: Returns document content string, site/session id and most recent revision number.
C.Edit	HTTPS POST *mutate* request: Contains a sequence of JSON encoded mutations.
C.UpdateCursorPosition	HTTPS POST *mutate* request: Contains a JSON encoded structure containing the new starting and ending positions of the cursor (for selection).
C.LoadRevisions	HTTPS GET *load* request: Specifies starting revision and ending revision (inclusive) of the desired range. Returns a sequence of JSON encoded revisions.
C.GetHistorySize	HTTPS GET *bind* request: Returns updates made by other collaborators; includes most recent revision number at the time of request, which corresponds to $\|\boldsymbol{H}\|$.

In particular, if formatting information is ignored, the document content is represented in Google Docs by a string of UTF-8 characters. Edits submitted by collaborators comprise an array of mutations (insertions or deletions) bundled into an atomic unit called a revision. Each revision is assigned a unique positive integer. The property of structural invariance of revisions, discussed in Section 2, is adhered to - potentially for the reasons mentioned earlier. Recall that structural invariance requires that the server persist the exact revision representation it receives from the collaborator without any manipulation. Arbitrary range queries are also supported; this functionality is provided by the official API [16]. It is important to note that the Google Docs interface briefly

[2] This work targets the version of Google Docs available at the time of writing - April 2012.

described in Table 1 pertains to the interface employed by the Google Docs client and not that specified in the API.

3.2 Implementation

A proof-of-concept browser extension was developed for the Firefox web browser [17] (tested with Firefox version 11).

The extension consists of two fundamental components - a lightweight module written in Javascript, which is loaded by the browser, and a more complex core written in Java that performs almost all of the processing. Intercommunication between Javascript and Java is facilitated by LiveConnect [18]. This solution offers far better performance for cryptographic computations than similar implementations developed entirely in Javascript [7,8], albeit at the expense of portability among browsers. Like the other implementations, the extension intercepts and modifies requests and responses depending on whether the URI is determined to match for Google Docs. Such protocol messages are then processed by the Java component. In addition, our extension establishes a 'background' HTTPS connection with the Google Docs server. This is made possible because the cookie used for session authentication can be readily captured since we are listening to HTTP messages to/from Google Docs.

Recall the abstraction of a revision transformation defined in Section 2.3. In the prototype extension, the concrete transformation that is adopted performs authenticated encryption of revisions. Therefore, both confidentiality and integrity of the document content are protected. More precisely, assume a set of collaborators share a symmetric key for a document. Now let the key space K be $\{0,1\}^\lambda$.

Our overall approach is independent of the choice of cipher or MAC. For the prototype, AES in GCM mode was chosen, largely due to performance. In comparison to the mode (RPC) employed in [7], the throughput of CTR mode is higher due to the encryption of 'chaining' nonces in RPC. This becomes more important for revisions that insert sizable content strings. Furthermore, aside from content, revision metadata can be authenticated using the same GHASH operation (this also readily facilitates linear integrity of revisions), which requires only a single multiplication in $GF(2^{128})$ for each block of encrypted and authenticated-only data. Another motivation is the parallelizable nature of both GHASH and CTR mode. Although sufficient for our proof-of-concept implementation, replacing GHASH with a more secure MAC should probably be considered for a deployed implementation due to several vulnerabilities of polynomial MACs, especially those based on $GF(2^{128})$ [19].

The transformation is thus described by the pair of algorithms ($\mathsf{Map}_k := \mathsf{EncR}_k, \mathsf{Invert}_k := \mathsf{DecR}_k$). Informally, EncR_k is a randomized algorithm that is roughly described as follows[3]

[3] Support for complete integrity defined in Section 5 is omitted from this description. One way to achieve this is to let EncR_k take a history argument and include the tag from the last revision in the authenticated-only data in the MAC (GHASH in this case).

Algorithm 5. EncR_k

```
Input: $\boldsymbol{r} \in \mathcal{R}$
 1: Uniformly sample $\mathsf{IV} \xleftarrow{\$} \{0,1\}^\lambda$
 2: $s \leftarrow \epsilon$
 3: for all insertion mutations $\mu_i$ in $\boldsymbol{r}$ do
 4:     Parse $\mu_i$ as $(\tau_{\mathsf{ins}}, v_i)$
 5:     $s \leftarrow s \parallel v_i$
 6: end for
 7: $c \leftarrow \mathsf{Base64Encode}(\mathsf{Encrypt}_{\mathsf{AES-GCM}\,k}(s, \mathsf{IV}))$
 8: Replace each $v_i$ with characters from $c$ equal in length to the original $v_i$. Denote
    the updated $\boldsymbol{r}$ as $\boldsymbol{r'}$. Denote the remainder of $c$ as $c'$.
 9: Append the mutation sequence $((\tau_{\mathsf{ins}}, c'), (\tau_{\mathsf{del}}, -|c'|))$ to $\boldsymbol{r'}$.
10: return $\boldsymbol{r'}$.
```

In brief, Algorithm 5 encrypts and MACs, using AES in GCM mode, the concatenation of every content string that is featured in a revision. In Google Docs, a revision is a JSON object encoded in UTF-8. As such, the raw bytes constituting the ciphertext and tag (produced by the MAC) must be appropriately encoded - we use Base64. The resultant encoded string can be partially distributed over the original mutations, but the remainder must be stored somewhere within the revision to enable future MAC validation and decryption.

We omit a description of the deterministic algorithm DecR_k but as expected, it merely removes the auxiliary mutations and decrypts the content (after validating the MAC).

3.3 Replaying Revisions

When a document is initially opened, the browser extension uses the background HTTPS connection it has established in a new thread to asynchronously fetch the complete revision history for the document ($C.\mathsf{LoadRevisions}$). As revisions are received, the replay algorithm (Algorithm 2) described in Section 2.4 is executed. The following steps are executed:

1. Run Algorithm 1 followed by Algorithm 2 to build a tree of *pieces* (substrings of the content inserted in a revision) that contribute 'content' to the present document string. Each piece in the tree is linked to the revision that contains it.
2. Transform the 'content' of each piece in the tree back into plaintext by running the reconstitution algorithm (Algorithm 4) to assemble the decrypted content string. The reconstitution algorithm applies DecR_k to the revision linked to each piece. In summary, the following steps are executed for every piece in the tree.

(a) Verify the authentication tag of the revision. Exit if verification fails.
(b) Decrypt the 'content' of the piece. In practice, only the ciphertext blocks in the revision that correspond to the piece are decrypted. Thus, a stream encipherment mode such as CTR is advantageous.
(c) Append the decrypted string to the document string

Concurrently, the extension intercepts the HTTPS response containing the complete document content that is used to render the document in the Google Docs editor. It then substitutes the reconstituted (decrypted) string for the original string in the response (which is merely ciphertext), and the Google Docs client renders the decrypted document.

3.4 Snapshot Optimization

A snapshot is a recording of the complete textual state of the document at a position in the discrete timeline of revisions. Snapshots have the same purpose here as 'checkpoints' in [11]. A snapshot consists of a string of text constituting the document's global content at the given time. Such a feature would be employed to optimize processing of a timeline, as a single revision subsumes the cumulative effect of a sequence of revisions. Accordingly, this would reduce the bandwidth consumption incurred by loading the revision history while also curtailing the number of replay steps necessary. This mechanism also helps to minimize the expansion overhead of encryption in addition to minimizing cost of decryption / MAC validation.

Concretely, a snapshot is a revision consisting of a deletion mutation that deletes the entire document content followed by an insertion mutation that restores it. Naturally, the snapshot's insertion mutation would be encrypted. We leave to future work the problem of systematically creating and locating snapshots. It seems that synchronization may be the most significant obstacle, but this can be mitigated by waiting for periods of inactivity.

Remark 1. The inclusion of a snapshot mechanism allows us to relax some of the assumptions made of the cloud provider. In particular, we only require that a constant number of most recent revisions in the timeline are subject to range queries and/or persisted.

4 Evaluation

4.1 Statistical Analysis of Revisions

A small collection of 20 documents authored in Google Docs were submitted by distinct users for analysis. These documents, evolved organically, were shared among an average of 3 users. Analysis of these documents reveals some interesting findings, although a much larger and diverse sample size is needed to draw clear conclusions.

In Table 2, 'filtered' means that only insertion revisions with a total number of characters ≤ 10 were considered. This amounts to 99.3% of all revisions

Table 2. Statistics of Revision Histories for Sample Documents

	Mean
Bytes Transferred Per Revision	701.2
Number of Inserted Characters Per Insertion Revision (filtered)	1.96
Number of Inserted Characters Per Insertion Revision (Unfiltered)	3.55

processed, which eliminates a significant number of revisions, though not all, generated by pasting or other means. The standard deviation in this particular case was 0.93. The high mean for the unfiltered case is explained by the occurrence of considerably large revisions in the data set arising from pasting or other non-typing events. Of particular interest are those revisions arising from 'naturally' typed edits. Another interesting question was the observed distribution of the number of inserted characters in revisions incorporating insertions. A histogram based on all revisions across all sample documents is shown in Figure 4

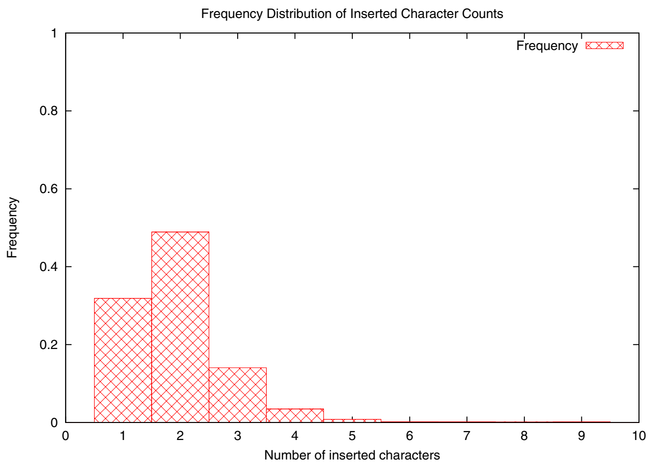


Fig. 4. Relative Frequency Distribution of the Number of Inserted Characters Per Insertion Revision

If the histogram is shifted to the left by 1, it bears some resemblance to the Poisson distribution with $\lambda \approx 1$. Although much work has shown a connection between the Pareto distribution and keystrokes sent on a network connection [20,21], this distribution does not appear to fit our empirical data. Clearly, more data is needed to find an appropriate model.

Remark 2. Considerable variability has been observed with autosaving in Google Docs and thus there can be fluctuating burst lengths for a variety of reasons ranging from network conditions to local workloads. Hence, these results were obtained to help extract some macro-scale behavioral patterns.

The histogram demonstrates that there will naturally be suboptimal ciphertext expansion in the proposed system. Furthermore, the figure reveals significant information leakage that can be exploited by an attacker. Together with timing information, there is scope to learn information about the plaintexts. Such attacks are discussed in Section 5.

4.2 Performance

Overhead Size. AES-GCM (128-bit block size) encryption is employed by the browser extension, and is configured to use a 16-byte MAC. Every revision is associated with a unique IV. Since a revision is an encoded JSON object, it is necessary to base64-encode the ciphertext + IV. Finally, two 'dummy' mutations are added - an insertion followed by a deletion. Consequently, the ratio of size of the processed mutations over the size of the single insertion mutation is approximately 4.82. To give concrete sizes, a mutation of 28 bytes grows to three mutations requiring an aggregate 135 bytes. Since revisions sent to the server typically contain additional metadata and formatting information, the percentage of bandwidth consumed by such overhead is less pronounced. This is highlighted in the results that were collected from analysis of sample documents, as shown in Table 3. It seems that on average approximately 4% of the bandwidth for insertion edits is consumed by the overhead incurred by authenticated encryption, although the overhead percentage is significant for the actual mutation sequence in a revision. Therefore, there is an appreciable impact on storage and bandwidth for initial loads, which further supports the future implementation of snapshots.

It must be emphasized that the bandwidth and storage overheads are markedly larger than the solution presented in [7], mainly due to constant-sized JSON formatting in the 'dummy' mutations. However, this is compensated by our preservation of full collaboration.

Table 3. Overhead Incurred by Authenticated Encryption (Unoptimized) For Insertion Edits

	Mean	**Standard Deviation**
Percentage Overhead Per Mutation Sequence	66.52%	13.18%
Percentage Overhead Per Revision	40.46%	5.8%
Percentage Overhead Per *Total* (headers + body) Mutate HTTP Message	4.37%	0.37%

Running Time. A microbenchmark was conducted on a machine with an Intel Core i3 processor running at 3.2 GHz and 4 GB of RAM. The mean was taken of 1,000 executions for each case. Each test run involved generating a 16-byte IV and performing an AES encryption in the selected mode of operation. Our implementation used the Bouncy Castle cryptography library [22].

Using the benchmark environment, the running times of the replay and reconstitution algorithms were evaluated. A small set of documents was employed, each containing approximately 10,000 revisions, half of which were encrypted

Table 4. Run-time performance of algorithms executed during a document's initial loading stage

	Mean	**Standard Deviation**
Running time of replay algorithm (documents without encryption)	$< 1\mu$s	$< 1\mu$s
Running time of replay algorithm (documents with encryption)	107 ms	11 ms
Running time of reconstitution algorithm (documents with encryption)	160 μs	54 μs

using our browser extension. Table 4 gives a rough guideline of the real-time performance of the initial load process where the fetched revisions are used to build a tree. Unsurprisingly, an appreciable cost is incurred for the encrypted documents, although it is small enough to remain hidden from users. Nevertheless, this is clearly not scalable. As the revision history increases, the bandwidth costs to transfer the history in addition to the processing thereof become impractical. Therefore, the snapshot algorithm described in Section 3.4 must be employed as an optimization. In practice, the point at which the history size becomes infeasible is determined by the timeouts dictated by browsers. Hence, it is necessary that the latency of the load revisions requests along with tree processing and reconstitution are less than this limit. Empirical results would suggest that this limit is well beyond 20,000 revisions for common usage patterns, and thus snapshots would not have to be taken frequently.

5 Security Analysis

5.1 Threat Model

A threat model that only focuses on passive attacks is considered. A malicious cloud provider could alter the client-side part of the web application to inject Javascript that surreptitiously relays to the server the unencrypted content rendered in the browser. Therefore, we assume the client-side code executing in the browser is trusted. This is the position assumed in other works, most notably [7]. Additionally, a host of other active threats such as mounting a man-in-the-browser attack are also not covered in this threat model.

We return briefly to the STCE model wherein we can express the desired security properties. Disregarding the client-side code, the interface to the coordinator C is a good reflection of the view of the real system. Let P be a set of parties with access to the document. Naturally, we have that $C \in P$. Let $S \subset C$ be a set of sites who possess a *pre*-shared key k for the document in question. Let $\mathcal{A}$ be an adversary in $P \backslash S$. We define some desirable security properties:

1. **Confidentiality**: $\mathcal{A}$ has a negligible advantage in learning any information about a ciphertext under an adaptive Chosen Ciphertext Attack (negligible advantage in the IND-CCA2 game discussed in Section 2.3).
2. **Revision Integrity**: $\mathcal{A}$ has a negligible advantage in forging a revision in $\boldsymbol{H}$ that passes the integrity check performed by parties in S.
3. **Complete Integrity**: A stronger property than (2) requires that an attacker is unable to efficiently forge an alternate history whose revisions all

satisfy (2) i.e. it has negligible advantage in the following game. The attacker is permitted to make adaptive queries to the oracles $\mathsf{MapH}(\mathsf{Map}_{k_1}, \cdot)$ and $\mathsf{InvertH}(\mathsf{Invert}_{k_2}, \cdot)$. Suppose it makes $q = poly(\lambda)$ queries to the former. It records $\boldsymbol{H'_i} \leftarrow \mathsf{MapH}(\mathsf{Map}_{k_1}, \cdot)$ where $\boldsymbol{H_i} \in \mathbb{H}$ for $0 \leq i < q$. It wins the game if it outputs a non-empty history $\boldsymbol{H'}$ such that $\boldsymbol{H'}$ is not a substring of any $\boldsymbol{H'_i}$ for $0 \leq i < q$ and $\mathsf{InvertH}(\mathsf{Invert}_{k_2}, \boldsymbol{H'}) \neq \perp$.

Properties (1) and (2) follow from our adoption of AES-GCM. A more thorough analysis of these properties is provided in the full version [4] of this paper. As stated earlier, a more secure alternative is to replace GHASH with a more secure MAC. Certain countermeasures are needed to limit the power of an adversary to choose arbitrary long messages; for example a revision may be broken up into independently encrypted/authenticated units whose size is limited.

Property (3) ensures that there exists some means to prevent C rearranging the revisions in a history to yield an alternative document content string. Thus, there needs to be a chain of dependence between successive revisions, for example by means of a linked authenticated structure. Consider a site with knowledge of a current history $\boldsymbol{H}$ (its local copy). Suppose it has to commit a new revision $\boldsymbol{r}$. A means to fulfill property (3) is to compute a linear hash chain of all revisions in $\boldsymbol{H}$. This is similar to the approach adopted in [11]. Naturally, only a single hash needs to be stored by a site, thus incrementality is not sacrificed. This hash along with a precise description of a revision is incorporated into the MAC that is included as part of the revision. Now it may be the case that other sites, who are synchronized with knowledge of $\boldsymbol{H}$, concurrently send revisions to C. These revisions can be arranged in an arbitrary order by C. However, the correct operational transformations must be performed on this ordering. Otherwise this can be detected (by virtue of including a site's position in the MAC). Hence, the only scope for re-ordering afforded to C is at such points of concurrency, which is inherent in the consistency notions of OT systems. As such, the game definition of property (3) deprives the adversary of this particular case. We defer to the full paper a more formal security assessment of this property together with other active attacks mountable by C such as dropping edits and forking documents as explored in [11]. Furthermore, this paper omits a detailed treatment of Property (3) from both the algorithm descriptions and the implementation, in particular verification of an entire history.

The issue of information leakage arising from timing patterns (including those due to typing patterns) is a fundamental concern. Timestamp information recorded in each revision facilitates the formation of a detailed profile of a user's editing characteristics. This prompts the questions: is it feasible for the browser extension to smoothen the distribution by inserting artificial delays? Perhaps a dedicated out-of-browser proxy would afford more control over this. Alternatively, perhaps another option is to implement a custom feature that allows a user to enter a particularly sensitive string of text (social security numbers etc...) that is then collectively sent as a single revision using the 'background' connection. This would serve to prevent information leakage sourcing from

[4] Full version available on the authors' webpage

typing traits. In fact, given the scope for information leakage, users may only desire privacy for certain sensitive fields in their documents, and thus disable encryption for the remaining content.

5.2 Conclusions and Future Work

We have presented a new approach for protecting confidentiality and integrity of documents in collaborative editing systems. A proof-of-concept implementation developed for the Firefox web browser demonstrates the practicality of this solution over alternative work in the literature. In particular, this approach is the first, to the best of our knowledge, to preserve full collaboration and not to rely on externalized, out-of-band resources to store metadata for deployed untrusted OT systems.

Many questions have been raised by this research, which will be followed up as part of future work. Some of these question relate to ensuring protection against malicious active attacks such as forking documents, lessening information leakage from timing information, and investigating and implementing optimizations such as snapshots.

Acknowledgments. The authors would like to thank the anonymous reviewers for many helpful comments and suggestions.

References

1. Mitzenmacher, M. (ed.): Fully homomorphic encryption using ideal lattices. ACM Press (September 2009)
2. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
3. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
4. Google docs (2012), https://docs.google.com/
5. Etherpad collaborative real-time editor (2012), http://www.etherpad.com
6. Apache wave (2012), http://incubator.apache.org/wave
7. Huang, Y., Evans, D.: Private editing using untrusted cloud services. In: 2011 31st International Conference on Distributed Computing Systems Workshops (ICDCSW), pp. 263–272 (2011)
8. Adkinson-Orellana, L., Rodríguez-Silva, D.A., Gil-Castiñeira, F., Burguillo-Rial, J.C.: Privacy for google docs: Implementing a transparent encryption layer. In: 2nd Cloud Computing International Conference - CloudViews (2010)
9. Bellare, M., Goldreich, O., Goldwasser, S.: Incremental cryptography: The case of hashing and signing, pp. 216–233 (1994)
10. Lipmaa, H.: Secure and Efficient Time-stamping Systems. Dissertationes mathematicae Universitatis Tartuensis. Tartu University Press (1999)
11. Feldman, A.J., Zeller, W.P., Freedman, M.J., Felten, E.W.: Sporc: group collaboration using untrusted cloud resources. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, OSDI 2010, p. 1. USENIX Association, Berkeley (2010)

12. Buonanno, E., Katz, J., Yung, M.: Incremental Unforgeable Encryption. In: Matsui, M. (ed.) FSE 2001. LNCS, vol. 2355, pp. 109–124. Springer, Heidelberg (2002)
13. Sun, C., Ellis, C.: Operational transformation in real-time group editors: issues, algorithms, and achievements. In: Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work, CSCW 1998, pp. 59–68. ACM, New York (1998)
14. Li, D., Li, R.: Preserving operation effects relation in group editors. In: Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work, CSCW 2004, pp. 457–466. ACM, New York (2004)
15. Li, R., Li, D.: Commutativity-based concurrency control in groupware. In: 2005 International Conference on Collaborative Computing: Networking, Applications and Worksharing, p. 10 (2005)
16. Google documents list api version 3.0 (2012), `https://developers.google.com/google-apps/documents-list/`
17. Mozilla firefox (2012), `http://www.mozilla.org/firefox`
18. Liveconnect technical documentation (2012), `https://developer.mozilla.org/en/LiveConnect`
19. Saarinen, M.J.O.: Cycling attacks on gcm, ghash and other polynomial macs and hashes. Cryptology ePrint Archive, Report 2011/202 (2011), `http://eprint.iacr.org/`
20. Donoho, D.L., Flesia, A.G., Shankar, U., Paxson, V., Coit, J., Staniford, S.: Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 17–35. Springer, Heidelberg (2002)
21. Paxson, V., Floyd, S.: Wide area traffic: the failure of poisson modeling. IEEE/ACM Trans. Netw. 3, 226–244 (1995)
22. Bouncy castle crypto api (2012), `http://www.bouncycastle.org`

Selective Document Retrieval from Encrypted Database

Christoph Bösch[1], Qiang Tang[2], Pieter Hartel[1], and Willem Jonker[1]

[1] University of Twente, The Netherlands
{c.boesch,pieter.hartel,willem.jonker}@utwente.nl
[2] APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi, L-1359 Luxembourg
qiang.tang@uni.lu

Abstract. We propose the concept of selective document retrieval (SDR) from an encrypted database which allows a client to store encrypted data on a third-party server and perform efficient search remotely. We propose a new SDR scheme based on the recent advances in fully homomorphic encryption schemes. The proposed scheme is secure in our security model and can be adapted to support many useful search features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product. To evaluate the performance, we implement the search algorithm of our scheme in C. The experiment results show that a search query takes only 47 seconds in an encrypted database with 1000 documents on a Linux server, and it demonstrates that our scheme is much more efficient, i.e., around 1250 times faster, than a solution based on the SSW scheme with similar security guarantees.

Keywords: Searchable Encryption, Homomorphic Encryption, Privacy.

1 Introduction

Outsourcing data to a third-party server is continuously gaining popularity because it can significantly reduce operational costs for a client. However, to store outsourced data securely on an untrusted server, the data should be encrypted to make it inaccessible to the server and other attackers. The issue is that, if the encryption is done with standard encryption schemes, the client will not be able to search anymore unless it retrieves the whole outsourced database from the server. To solve the problem, we need a special type of encryption primitive which allows the following things.

1. The client can encrypt his data and store the ciphertext on the server. More specifically, we assume the client stores a list of (document, index) pairs on the server, where the index is an encrypted version of the keywords which appear in the document. Note that the document should be encrypted independently. We skip the details of document encryption in the paper because it is not relevant for the search.

2. The client can ask the server to search in the indexes on his behalf, without leaking information about the keywords in the indexes and what has been searched for. Moreover, the client may even want to hide from the server the fact which documents have been matched by a search query.
3. The client can selectively retrieve the contents identified by a search, possibly in a private manner. The option for the client to selectively retrieve matched documents may be very useful in practice. For example, a search may indicate that 900 out of 1000 documents are matched, but the client may just want to retrieve 10 of them instead of all of them due to various reasons. The option for the client to retrieve matched documents in a private manner may also be useful in practice since which documents are retrieved can already leak some information about the documents.

The first two requirements are straightforward and are coming from security considerations, while the last one is coming from flexibility, efficiency, and security considerations. Note that, an alternative solution is for the client to store a plaintext copy of the indexes locally so that he can search by himself. Clearly, this is not a good solution because the client needs to maintain the index storage.

1.1 Problem Statement

In the direction of solving the above problem, searchable encryption (SE) schemes have been the most relevant cryptographic primitive. A SE scheme enables a third-party server to search on his behalf directly on encrypted data without knowing the plaintext data. In particular, SE in the symmetric setting (referred to as SSE throughout the paper) can serve as a more suitable solution, where the term *symmetric* means that only the client can generate searchable contents. It is worth noting that there also exist SE schemes in the asymmetric setting, such as PEKS [1], where the concept of a public key encryption scheme is employed and every entity can generate searchable data. Clearly, asymmetric SE is of less interest to our problem, compared to SSE.

The concept of SSE and its security model can be found in [5,27,28]. From the definition, we can see that it is meant to achieve the functionalities in the first two requirements mentioned before. By a straightforward extension as discussed in *Remark 1* in Section 2.1, it can achieve the functionality in the third requirement. However, with respect to the desired security guarantees, an SSE scheme leaks a lot of sensitive information to the server, and such information includes (at least) which documents match the client's search request and which documents the client has retrieved.

The research problem we will address in this paper is to design a new cryptographic primitive, which satisfies all three requirements from the perspectives of both functionalities and security guarantees.

1.2 Our Contribution

Firstly, we propose a new cryptographic primitive, namely selective document retrieval (SDR), and present a security model. Secondly, based on the recent advances in fully homomorphic encryption schemes and the index construction technique by Chang and Mitzenmacher [5], we propose a SDR scheme to support equality test predicates and prove its security in the proposed security model. The intuition behind the construction is rather straightforward, but interestingly it can serve as a framework to support more flexible search features. We show that the proposed SDR scheme can be easily adapted to support features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product.

Thirdly, we set appropriate parameters for the symmetric BV encryption scheme [3] and implement it in C. This is the first publicly-available implementation of the scheme in C with carefully chosen parameters, so that it may be of independent interest for other works. We use the BV scheme to instantiate the encryption component in the proposed SDR scheme, and evaluate the performances. The experiment results show that a search query takes only 47 seconds in an encrypted database with 1000 documents and 100 keywords, while a search query takes around 10 minutes in an encrypted database with 5000 documents and 250 keywords. In contrast, for the SSW scheme by Shen et al. [27], a search query takes around 16 hours in an encrypted database with 1000 documents and 100 keywords on the same server. We did not study the document retrieval performance in the paper, because it will be similar for all schemes if they are to achieve a similar level of security. We note that although the performance of the proposed SDR scheme does not say that it is an efficient solution in all application scenarios, it is the most efficient one we have now.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we describe SDR and formalize its security property. In Section 3, we propose a SDR scheme and prove its security. In Section 4, we describe various search features of the proposed SDR scheme. In Section 5, we implement the search algorithm of the proposed SDR scheme and analyse the experimental results. In Section 6 we review related work. Section 7 concludes the paper.

2 Definitions of SDR

Throughout the paper, we use the following notation. Given a document d consisting of keywords w, let $u(d)$ denote the distinct keywords contained in d. Let $\mathcal{D}$ be the possible document space and $\mathcal{W} = \{s_0, \ldots, s_b\}$ be a pre-built dictionary of keywords to search for, which satisfies that $u(d) \subseteq \mathcal{W}$ for any $d \in \mathcal{D}$.

2.1 Algorithmic Definition of SDR

An SDR scheme comprises five algorithms (Keygen, $\mathsf{BuildIndex}$, $\mathsf{Trapdoor}$, $\mathsf{SearchIndex}$, $\mathsf{Retrieve}$), defined as follows.

– $\mathsf{Keygen}(s)$: Run by a client, this algorithm takes a security parameter s as input, and outputs a secret key K. It may also generate some other public parameters such as a predicate set $\mathcal{F}$.
– $\mathsf{BuildIndex}(K, d)$: Run by the client, this algorithm takes the key K and a document $d \in \mathcal{D}$ as input, and outputs an index $\mathcal{I}_d$ which encodes $u(d)$ (i.e. all keywords w from the document d).
– $\mathsf{Trapdoor}(K, \mathsf{f})$: Run by the client, this algorithm takes the key K and a predicate $\mathsf{f} \in \mathcal{F}$ as input, and outputs a trapdoor T_f.
– $\mathsf{SearchIndex}(T_\mathsf{f}, \mathcal{I}_d)$: Run by the server, this algorithm takes a trapdoor T_f and an index $\mathcal{I}_d$ as input and returns an encrypted result $[\![R_d]\!]$ to the client, where R_d implies whether $u(d)$ satisfies the predicate f or not.
– $\mathsf{Retrieve}(K, \{[\![R_d]\!] \,|\, d \in \mathcal{DB}\}; \mathcal{DB})$: Run between the client and the server, the client takes the secret key K and the encrypted search results $\{[\![R_d]\!] | d \in \mathcal{DB}\}$ as input and the server takes the encrypted database $\mathcal{DB}$ as input. At the beginning of the protocol, the client first decrypts $\{[\![R_d]\!] \mid d \in \mathcal{DB}\}$ and decides which documents to retrieve, and at the end of the protocol the client retrieves the documents he wants.

A standard work flow of SDR usage is as follows. A client first runs the Keygen algorithm to generate the key K and parameters, then runs the BuildIndex algorithm to generate an index for every document he has, finally stores every (document, index) pair on the server. We assume that the documents are encrypted by the client with some standard symmetric encryption algorithm using a key different from K. Later on, when the client wants to retrieve some documents, it first runs the Trapdoor algorithm to generate a trapdoor, then sends the trapdoor to the server which can then run the SearchIndex algorithm to match the trapdoor with every index in the database and send the encrypted match results to the client. Finally, the client runs the Retrieve algorithm with the server to retrieve (some of) the matched documents. Note that the client can selectively retrieve the matched documents, not necessarily all of them.

Remark 1. Referring to the definition of SSE [7], any SSE scheme can be trivially extended to a SDR scheme: by letting the server send the search results (i.e. outputs of SearchIndex executions) back to the client, who then selectively determines which documents to retrieve. If we assume that the server returns all the documents matched by the SearchIndex in SSE, then it is equivalent to a SDR scheme in which the client always retrieves all the matched documents.

Similar to the case in other cryptographic primitives, an SDR scheme should always be sound, namely the following two conditions should always hold.

1. If $u(d)$ satisfies f, then $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{DB}\}; \mathcal{DB})$ will return all documents d chosen by the client.
2. If $u(d)$ does not satisfy f, then the probability that $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{DB}\}; \mathcal{DB})$ returns d is negligible.

2.2 Security Properties for SDR

Recall that the main objective of SDR schemes is to enable the server to search over the encrypted data and let the client selectively retrieve the matched contents. In this setting, information leakage can come from three sources, namely index, trapdoor, and query results. Correspondingly, there are three types of privacy concerns.

– *Index privacy*, similar to the plaintext privacy in [27], means that indexes should not leak any information about the encoded keywords.
– *Trapdoor privacy*, similar to the predicate privacy in [27], means that trapdoors should not leak any information about the encoded predicates.
– *Query result privacy* means that if the client retrieves x documents for any integer x in two executions of the Retrieve algorithm, then the server should not know whether the two executions return the same documents or not.

The concerns of index privacy and trapdoor privacy have been considered by existing SSE schemes. Notably, Shen et al. [27] propose a definition of *full security*, which tries to capture the above two privacy concepts. Note that, Shen et al. only give a fully secure SSE scheme which support inner product queries for vectors of even length, without being able to present a scheme which generally achieves *full security*. To our knowledge, no SSE scheme has been shown to be fully secure in general.

However, query result privacy has not been touched upon in the setting of outsourcing encrypted data, although it is a practical concern for many application scenarios. For example, suppose that Alice stores both her work-related documents and personal documents on a remote server protected by an SSE scheme. Moreover, she only queries her work-related documents in her office, and queries personal documents at home. One day, if the server notices at 10:00 pm that Alice is querying the same document as that she queried at 11:00 am, then the server can guess that Alice is working over the time in her office.

2.3 Game-Style Security Definition

Similar to the security definitions in SSE security models, we consider the attacker to be a semi-honest server (and any other outside attacker). By semi-honest we mean an honest-but-curious [12] database server that can be trusted to adhere to the protocol, but which tries to learn as much information as possible. Formally, the definition is as follows.

Definition 1. *An SDR scheme is secure if no probabilistic polynomial-time attacker has non-negligible advantage in the attack game defined in Fig. 1, where the advantage is defined to be* $|\Pr[b = b'] - \frac{1}{2}|$.

By granting index oracle queries to the attacker, we cover index privacy in the sense that the attacker cannot distinguish the indexes of different documents. By granting trapdoor oracle queries to the attacker, we cover trapdoor privacy

1. The challenger runs the Keygen algorithm and obtains the secret key K and the predicate set $\mathcal{F}$. The challenger publishes $\mathcal{F}$ and picks a random bit b.
2. The attacker $\mathcal{A}$ adaptively makes the following types of queries.
   - **Index oracle query.** On the j-th index query, $\mathcal{A}$ outputs two documents $d_{j,0}, d_{j,1} \in \mathcal{D}$. The challenger responds with $\mathsf{BuildIndex}(K, d_{j,b})$.
   - **Trapdoor oracle query.** On the i-th trapdoor query, $\mathcal{A}$ outputs two predicates $f_{i,0}, f_{i,1} \in \mathcal{F}$. The challenger responds with $\mathsf{Trapdoor}(K, f_{i,b})$.
   - **Retrieve oracle query.** Suppose that there have been j index queries and i trapdoor queries, the challenger (simulating the client) and the server run the Retrieve algorithm. The server's input is the database $\mathcal{DB}$, which contains j (index, document) pairs, and the challenger's input is the key K and a set of document identifiers $\mathcal{ID}_b$, where $\mathcal{ID}_0$ and $\mathcal{ID}_1$ are two identifier sets of identical size chosen by the attacker. Basically, $\mathcal{ID}_b$ tells which documents the challenger should retrieve.
3. $\mathcal{A}$ outputs a guess b' of the bit b.

Fig. 1. Attack Game of SDR

in the sense that the attacker cannot distinguish the trapdoors received from the client. Similarly, by granting retrieve oracle queries to the attacker, we cover query result privacy in the sense that the attacker cannot tell the retrieved documents by the client. Note that in granting the retrieve oracle queries, we restrict that the identity sets are of the same cardinality; otherwise the attacker may trivially win the game unless the client always retrieves all the documents. As a consequence, if an SDR scheme is secure under this definition, an attacker only learns how many documents the challenger has retrieved but nothing else.

2.4 Relaxation of the Security Definition

As discussed before, query result privacy may be an important concern in many application scenarios for SDR schemes, but it may not be so important in other scenarios. To be secure under Definition 1, the Retrieve algorithm of an SDR scheme will use private information retrieval [6, 22] technique in one way or another so that it will incur significant computational and communication complexities, hence it is very likely that this privacy property may be sacrificed for the efficiency reasons. As a result, it is useful to have a definition covering only index privacy and trapdoor privacy. Formally, we give the following definition.

Definition 2. *An SDR scheme achieves index privacy and trapdoor privacy, if no probabilistic polynomial-time attacker has non-negligible advantage in the attack game defined in Fig. 1 with the following exceptions.*

1. *Retrieve oracle challenge query is disallowed in the game.*
2. *For any index oracle challenge query $(d_{j,0}, d_{j,1})$ and any trapdoor oracle challenge query $(f_{i,0}, f_{i,1})$, the following is true: $u(d_{j,0})$ satisfies $f_{i,0}$ if and only if $u(d_{j,1})$ satisfies $f_{i,1}$.*

With the relaxation, the above definition provides the same level of security guarantees to the full security definition [27].

3 The Proposed SDR Scheme

In this section, we propose a new SDR scheme and prove its security in the security model described in Section 2. We describe the scheme for the case of equality test predicates, while the scheme does support other types of predicates which will be elaborated in Section 4.

3.1 Preliminary

An encryption function $E(\cdot)$ is called homomorphic if there exist two (possibly the same) operations ($\otimes$ and $\oplus$), such that $E(a) \otimes E(b) = E(a \oplus b)$. In this paper the homomorphic encryption of an element x is written as $[\![x]\!]$. Thus $[\![a]\!] \otimes [\![b]\!] = [\![a \oplus b]\!]$. In our construction, we use a semantically secure homomorphic encryption scheme that allows one multiplication followed by multiple additions on encrypted values. For example, the lattice-based schemes such as the Gentry-Halevi-Vaikuntanathan (GHV) scheme [10] and Brakerski-Vaikuntanathan (BV) scheme [3] and the pairing-based Boneh-Goh-Nissim (BGN) scheme [2] satisfy the required property.

3.2 The Proposed Scheme

The proposed SDR scheme makes use of a symmetric homomorphic encryption scheme satisfying the requirements stated in Section 3.1 and the index construction method by Chang and Mitzenmacher [5]. Next, we describe the algorithms of the proposed scheme, namely (Keygen, $\mathsf{BuildIndex}$, $\mathsf{Trapdoor}$, $\mathsf{SearchIndex}$, $\mathsf{Retrieve}$).

- $\mathsf{Keygen}(s)$. Given a security parameter s, generate a key K for a symmetric homomorphic encryption scheme, such as the symmetric version of the BV scheme described in Section 5.1, and equality test predicate set $\mathcal{F} = \{\mathsf{f}_w \,|\, w \in \mathcal{W}\}$. For any document d, $u(d)$ satisfies f_w if and only if $w \in u(d)$.
- $\mathsf{BuildIndex}(K, d)$. With the key K and a document d, the algorithm does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[b])$. Note that b is the size of the possible keyword set. The bit $I_d[i]$ is set to be 1 if $w \in u(d) = s_i \in \mathcal{W}$; otherwise, the $I_d[i]$ is set to be 0.
  3. Generate $[\![I_d]\!] = ([\![I_d[1]]\!], [\![I_d[2]]\!], \ldots, [\![I_d[b]]\!])$, which means the plaintext version index is encrypted bit by bit.
  4. Output the index $\mathcal{I}_d = [\![I_d]\!]$.
- $\mathsf{Trapdoor}(K, \mathsf{f}_w)$. With the key K and a predicate f_w, the algorithm does the following:
  1. Construct $t_{\mathsf{f}_w} = (t_{\mathsf{f}_w}[1], t_{\mathsf{f}_w}[2], \ldots, t_{\mathsf{f}_w}[b])$. For every $1 \le i \le b$, the value of $t_{\mathsf{f}_w}[i]$ is set to be 1 if $w = s_i$ and 0 otherwise.
  2. Output the trapdoor $T_{\mathsf{f}_w} = ([\![t_{\mathsf{f}_w}[1]]\!], [\![t_{\mathsf{f}_w}[2]]\!], \ldots, [\![t_{\mathsf{f}_w}[b]]\!])$.

- $\mathsf{SearchIndex}(T_{\mathsf{f}_w}, \mathcal{I}_d)$. With a trapdoor T_{f_w} and an index $\mathcal{I}_d$, the algorithm outputs $[\![R_d]\!] = [\![t_{\mathsf{f}_w} \odot I_d]\!]$, where the notation $\odot$ represents an inner product. Note that the computation is based on T_{f_w} and $\mathcal{I}_d$ using the homomorphic properties stated in Section 3.1. The server sends $[\![R_d]\!]$ to the client.
- $\mathsf{Retrieve}(K, \{[\![R_d]\!] \mid d \in \mathcal{DB}\}; \mathcal{DB})$. Here, $\mathcal{DB}$ is the database which contains all the (document, index) pairs the client has stored at the server. The client and the server interact as follows:
  1. The client first decrypts the encrypted search results $\{[\![R_d]\!] \mid d \in \mathcal{DB}\}$, and gets to know which are the matched documents.
  2. The client decides a subset of the matched documents, and runs a private information retrieval (PIR) protocol (e.g. [6,22,25]) with the server to retrieve the documents.

For efficiency reasons, in the Retrieve algorithm, the client can select the desired documents and directly tell the server which documents he wants.

3.3 Security Results

With respect to the proposed SDR scheme, it is clear that the SearchIndex algorithm always returns 1 if $u(d)$ satisfies f_w and 0 otherwise. Hence, the soundness property is achieved given that the PIR protocol used in the Retrieve algorithm is also sound. Next, we summarize the security of the SDR scheme and leave the proofs to the full paper due to space reasons.

Theorem 1. *The proposed SDR scheme in Section 3.2 is secure under Definition 1 given that the adopted symmetric homomorphic encryption scheme is IND-CPA secure [14] and the PIR protocol in the* Retrieve *algorithm is secure [6].*

In the proposed SDR scheme, if the client directly retrieves the matched documents without using a PIR protocol in the Retrieve algorithm, then the scheme achieves the relaxed security under Definition 2 given that the encryption scheme is IND-CPA secure. The intuition is very straightforward based on the fact that all operations in the search are carried out in the ciphertext domain using the homomorphic properties of the encryption scheme.

Theorem 2. *The proposed SDR scheme without using PIR protocol in the* Retrieve *algorithm achieves index privacy and trapdoor privacy under Definition 2 given that the adopted symmetric homomorphic encryption scheme is IND-CPA secure [14].*

4 Adaptations of the Proposed SDR Scheme

In the previous section, we described an SDR scheme and analysed its security. Besides supporting equality test predicates, the scheme can be adapted to support a number of useful search features, including aggregating search results, supporting conjunctive keyword search queries, advanced keyword search, search with keyword occurrence frequency, and search based on inner product. Moreover, based on the same analysis in Section 3.3, all variants in this section are still secure in our security model. We also show that it is straightforward to adapt the proposed SDR scheme to the asymmetric setting or multi-user setting.

4.1 Aggregating Search Results

In the proposed scheme, the server has to send back an $[\![R_d]\!]$ for each document. If the symmetric BV scheme [3] is used in the scheme, to reduce the communication complexity, we can transform (depending on the degree α of the polynomials) up to α ciphertexts that encode α bits separately, into a single ciphertext C_p [17]. For a detailed description of the BV-scheme and the used variables, we refer the reader to Section 5.1. The *packed* ciphertext is calculated by:

$$C_p = \left(\sum_i c_{0,i} x_i, \sum_i c_{1,i} x_i \right).$$

This means, for a collection of 1000 documents and using a 1024 degree polynomial, the server has to send back only one ciphertext instead of 1000.

4.2 Conjunctive Keyword Search

To support conjunctive keyword search queries for any number of keywords, we propose a variant of the proposed SDR scheme. The $\mathsf{Trapdoor}$ algorithm needs to be changed slightly, while other algorithms stay basically the same. For conjunctive keyword search, the predicate set can be denoted as $\mathcal{F} = \{\mathsf{f}_{\mathcal{W}'} \,|\, \mathcal{W}' \subseteq \mathcal{W}\}$. For any document d, $u(d)$ satisfies $\mathsf{f}_{\mathcal{W}'}$ if and only if $u(d) \subseteq \mathcal{W}'$.

- $\mathsf{Trapdoor}(K, \mathsf{f}_{\mathcal{W}'})$. With the key K and a predicate $\mathsf{f}_{\mathcal{W}'}$, it does the following:
  1. Construct $t_{\mathsf{f}_{\mathcal{W}'}} = (t_{\mathsf{f}_{\mathcal{W}'}}[1], t_{\mathsf{f}_{\mathcal{W}'}}[2], \ldots, t_{\mathsf{f}_{\mathcal{W}'}}[b])$. For every keyword w_i from $\mathcal{W}$, the value of $t_{\mathsf{f}_{\mathcal{W}'}}[i]$ is set to be 1 if $s_i \in \mathcal{W}'$ and 0 otherwise.
  2. Output the trapdoor $T_{\mathsf{f}_{\mathcal{W}'}} = ([\![t_{\mathsf{f}_{\mathcal{W}'}}[1]]\!], [\![t_{\mathsf{f}_{\mathcal{W}'}}[2]]\!], \ldots, [\![t_{\mathsf{f}_{\mathcal{W}'}}[b]]\!])$.

As a result of the modification, the output of a $\mathsf{SearchIndex}(T_{\mathsf{f}_{\mathcal{W}'}}, \mathcal{I}_d)$ query tells the client how many keywords in the trapdoor appear in the index $\mathcal{I}_d$.

4.3 Advanced Keyword Search

In some application scenarios, the client may care about some keywords more than others, which implies that it is desirable to allow the client to put a weight on each keyword in the trapdoor. To do so, we propose another variant of the proposed SDR scheme. The $\mathsf{Trapdoor}$ and $\mathsf{Retrieve}$ algorithms need to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set can be denoted as $\mathcal{F} = \{\mathsf{f}_{\mathcal{W}'} \,|\, \mathcal{W}' \subseteq \mathcal{W}\}$, as specified in Section 4.2.

- $\mathsf{Trapdoor}(K, \mathsf{f}_{\mathcal{W}'})$. With the key K and a predicate $\mathsf{f}_{\mathcal{W}'}$, it does the following:
  1. Construct $t_{\mathsf{f}_{\mathcal{W}'}} = (t_{\mathsf{f}_{\mathcal{W}'}}[1], t_{\mathsf{f}_{\mathcal{W}'}}[2], \ldots, t_{\mathsf{f}_{\mathcal{W}'}}[b])$. For every keyword w_i from $\mathcal{W}$, the value of $t_{\mathsf{f}_{\mathcal{W}'}}[i]$ is set to be 2^{i-1} if $s_i \in \mathcal{W}'$ and 0 otherwise.
  2. Output the trapdoor $T_{\mathsf{f}_{\mathcal{W}'}} = (\llbracket t_{\mathsf{f}_{\mathcal{W}'}}[1] \rrbracket, \llbracket t_{\mathsf{f}_{\mathcal{W}'}}[2] \rrbracket, \ldots, \llbracket t_{\mathsf{f}_{\mathcal{W}'}}[b] \rrbracket)$.
- $\mathsf{Retrieve}(K, \{\llbracket R_d \rrbracket \mid d \in \mathcal{DB}\}; \mathcal{DB})$. Here, $\mathcal{DB}$ is the database which contains all the (index, document) pairs the client has stored at the server. The client and the server interact as follows:
  1. The client first decrypts the encrypted search results $\{\llbracket R_d \rrbracket | d \in \mathcal{DB}\}$. For every document d, the client can recover which keywords are contained in the index (by writing R_d in a binary form, if the i-th bit is 1 then s_i is contained in the index). The client can then add weights on the keywords and decide which documents to retrieve.
  2. The client and the server run a PIR protocol for the client to retrieve the documents.

By letting the client know exactly, which of several keywords satisfy the search, the client is able to run multiple queries at once using only one trapdoor.

4.4 Search with Keyword Occurrence Frequency

In practice, a search query may rank the relevance of a document based on not only whether some keywords are contained but also the occurrence frequency of these keywords in the documents. The proposed scheme can be modified to support such a requirement. To do so, we proposed another variant of the proposed SDR scheme. The BuildIndex algorithm needs to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set is still the equality test one.

- $\mathsf{BuildIndex}(K, d)$. With the key K and a document d, it does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[b])$. The bit $I_d[i]$ is set to be the occurrence frequency of w if $w \in u(d) = s_i \in \mathcal{W}$; otherwise, the $I_d[i]$ is set to be 0.
  3. Generate $\llbracket I_d \rrbracket = (\llbracket I_d[1] \rrbracket, \llbracket I_d[2] \rrbracket, \ldots, \llbracket I_d[b] \rrbracket)$.
  4. Output the index $\mathcal{I}_d = \llbracket I_d \rrbracket$.

In this variant, the value of a $\mathsf{SearchIndex}(T_{\mathsf{f}_w}, \mathcal{I}_d)$ query tells the client the occurrence of the keyword w in the document d, and then the client can decide which documents to retrieve accordingly.

4.5 Search Based on Inner Product

To support search based on inner product, we propose another variant of the proposed SDR scheme. The BuildIndex and Trapdoor algorithms need to be changed slightly, while other algorithms stay basically the same. For this variant, the predicate set can be denoted as $\mathcal{F} = \{\mathsf{f} = (f[1], f[2], \cdots, f[b]) \,| f[i] (1 \le i \le b) \in \mathbb{N}\}$.

- $\mathsf{BuildIndex}(K, d)$. With the key K and a document d, it does the following:
  1. Generate the list of distinct keywords, namely $u(d)$.
  2. Construct a plaintext index for d, denoted as $I_d = (I_d[1], I_d[2], \ldots, I_d[b])$. The value $I_d[i]$ is set to be w if $w = s_i$; otherwise, the $I_d[i]$ is set to be 0.
  3. Generate $[\![I_d]\!] = ([\![I_d[1]]\!], [\![I_d[2]]\!], \ldots, [\![I_d[b]]\!])$.
  4. Output the index $\mathcal{I}_d = [\![I_d]\!]$.
- $\mathsf{Trapdoor}(K, \mathsf{f})$. With the key K and a predicate f, the algorithm outputs the trapdoor $T_\mathsf{f} = ([\![f[1]]\!], [\![f[2]]\!], \ldots, [\![f[b]]\!])$.

As a result of the modification, the output of a $\mathsf{SearchIndex}(T_\mathsf{f}, \mathcal{I}_d)$ query tells the client the inner product of f and the keyword vector in the index $\mathcal{I}_d$.

4.6 Multi-user Variant (adaption to asymmetric setting)

In some application scenarios, it may be desirable that multiple users are able to write new data to an existing database as in the case of PEKS [1]. The proposed SDR scheme can be extended straightforwardly to meet the requirement. In the Keygen algorithm, the client generates a public/private key pair for a homomorphic public key encryption scheme, such as the public key version of the BV scheme [3]. In the algorithms BuildIndex, Trapdoor, and SearchIndex, the encryptions are done with the client's public key. The Retrieve algorithm stays the same. In the extended scheme, everyone can generate searchable indexes based on the client's public key. However, only the client with the private key is able to decrypt the search results which are always encrypted under the client's public key. Thus, compared with other similar schemes in the asymmetric setting such as PEKS [1], the extended scheme does not suffer from the inherent offline keyword recovery attacks [4]. Without the client's secret key, the server cannot get the output of a search.

5 Search Performances of the Proposed SDR Scheme

In this section, we adapt the recent lattice-based symmetric BV scheme [3] to our proposed solution and explain our choice of parameters. We then show our implementation results and discuss some optimizations for the implementation. Note that our implementation focuses on the SearchIndex algorithm, in an attempt to demonstrate the efficiency differences between the proposed SDR scheme and existing SSE schemes.

5.1 Adaption of the Symmetric BV Scheme

In this subsection we denote scalars in plain and vectors in bold. We write $x \overset{R}{\leftarrow} X$ when we mean that x is chosen at random from the distribution X. The scheme uses the following parameters:

- the dimension α, which is a power of 2,
- the modulus q, which is a prime such that $q \equiv 1 \pmod{2\alpha}$

– the cyclotomic polynomial $f(x) = x_\alpha + 1$,
– the error distribution χ over the ring $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$
– ciphertext degree D (supports $D-1$ multiplications),
– number of supported additions A,
– message space $t < q$, which is prime,
– error parameter σ (standard deviation of the discrete Gaussian error distribution).

All parameters are chosen in such a way to guarantee correctness and security of the scheme. For correctness the BV scheme requires:

$$q \geq 4 \cdot (2t\sigma^2\sqrt{\alpha})^D \cdot (2\alpha)^{(D-1)/2} \cdot \sqrt{A}.$$

Note that D is the ciphertext degree and not the number of supported multiplications [17]. The encryption scheme consists of the following algorithms. We simplified the Mul and Add algorithms to support one multiplication followed by several additions:

– $\mathsf{SH.Keygen}(1^\kappa)$: Sample a ring element $s \overset{R}{\longleftarrow} \chi$ and set the secret key $sk := s$. (If we only care about homomorphism, sampling $s \overset{R}{\longleftarrow} R_q$ is sufficient.)
– $\mathsf{SH.Enc}(sk, m)$: Recall that the message space is R_t. We encode our message as a degree α polynomial with coefficients in $\mathbb{Z}_t$. To encrypt, sample $a \overset{R}{\longleftarrow} R_q$ and $e \overset{R}{\longleftarrow} \chi$ and output the ciphertext $\mathbf{c} = (c_0, c_1) \in R_q^2$ where $c_1 = -a$ and $c_0 = as + te + m$.
– $\mathsf{SH.Mul}(\mathbf{c}, \mathbf{c}')$: Given the two ciphertexts $\mathbf{c} = (c_0, c_1)$ and $\mathbf{c}' = (c_0', c_1')$ output the ciphertext vector $\mathbf{c}_{\text{mul}} = \mathbf{c} \cdot \mathbf{c}' = (c_0c_0', c_0c_1' + c_0'c_1, c_1c_1')$ using polynomial multiplication.
– $\mathsf{SH.Add}(\mathbf{c}, \mathbf{c}')$: Given the two ciphertexts $\mathbf{c} = (c_0, c_1, c_2)$ and $\mathbf{c}' = (c_0', c_1', c_2')$ output the ciphertext vector $\mathbf{c}_{\text{add}} = \mathbf{c} + \mathbf{c}' = (c_0 + c_0', c_1 + c_1', c_2 + c_2') \in R_q^3$ which is calculated by coordinate-wise vector addition of the ciphertext vectors.
– $\mathsf{SH.Dec}(sk, \mathbf{c})$: To decrypt, first define the secret key vector $\mathbf{s} = (1, s, s^2, \ldots, s^D) \in R_q^{D+1}$, compute $\langle \mathbf{c}, \mathbf{s}\rangle = \sum_{i=0}^{D} c_i s^i \in R_q$, and output the message $m = \langle \mathbf{c}, \mathbf{s}\rangle \pmod t$.

5.2 Choice of BV Parameters and Implementation

We choose our parameters for the symmetric BV scheme based on our needs, and also take into account the work of Lauter et al. [17] which assessed the security against the decoding attack [18] and the distinguishing attack [21]. We use the following parameters: $D = 2, A = 100, t = 2, \sigma = 8$. With these fixed parameters, we calculate the flexible parameters as seen in Table 1. We made experiments with smaller q and larger A (up to 1000) and still ended up with correct results.

Table 1. Implementation results for the parameters mentioned in Section 5.2. The degree of the polynomials is denoted by α, $\lceil \lg(q) \rceil$ is the bit size of q, and $\lg(T)$ is the logarithm of the runtime of the distinguishing attack from [18]. WC $|c|$ is the worst case ciphertext size and the last two columns describe the time in seconds, that is required for a single multiplication or addition, respectively.

α	$\lceil \lg(q) \rceil$	$\lg(T)$	WC $\|c\|$	MUL	ADD
256	14	64	896 B	410 E-06	11 E-06
512	20	107	2.5 kB	454 E-06	21 E-06
1024	33	134	8.25 kB	2.8 E-03	72 E-06

We implemented the scheme in C/C++ using FLINT, namely Fast Library for Number Theory [16]. We tested the code on an Intel Xeon CPU $X5677@3.47$ GHz running linux 2.6.37-sabayon x86_64. In this situation, our results for degree 512 polynomials show that an addition (after a multiplication) takes $21 \cdot 10^{-6}$ seconds and a multiplication takes $454 \cdot 10^{-6}$ seconds.

At this moment, we only have a single threaded implementation of our scheme. The homomorphic multiplication operation has to calculate four independent polynomial multiplications, which can be done in parallel. This will decrease the computation time significantly. The same is applicable for the addition operation, which uses three independent polynomial additions. These additions can also be easily done in parallel. Another optimization, which is mentioned by Lauter et al. [17] is to use the Fast Fourier Transformation (FFT) to speed up computations. This has already been considered in SWIFFT [20]. Due to the choice of parameters ($\mathbb{Z}_q \mod x^\alpha + 1$, where α is a power of 2 and $q = 1 \pmod{2\alpha}$) the FFT can be computed more efficiently.

To compare our scheme with others, we also implemented a type A symmetric prime order pairing, using the PBC [19] library. On the same machine, a single pairing operation takes $5.8 \cdot 10^{-3}$ seconds.

5.3 Performance of the Proposed SDR Scheme

We now consider the efficiency of the proposed SDR scheme, where the efficiency is measured in terms of the computation, communication and space complexities.

In Table 2, the first column shows the number of supported search keywords. The second column shows the number of documents stored on the server. The third and fourth columns show the number of required additions and multiplications for a search over the database. The last two columns show the worst case trapdoor size, which has to be transmitted, depending on the degree of the polynomial. Based on the performances of the symmetric BV scheme, for a document set of size 1000 with a keyword set of size 100, a search takes 47 seconds. For a document set of size 5000 with a keyword set of size 250, a search takes around 10 minutes. The result of a query is of size $\lceil \frac{\text{Docs}}{\alpha} \rceil \cdot |c|$, where α is the degree of the polynomial and $|c|$ the size of a single ciphertext according to Table 1. Note that the worst case trapdoor size is also the worst case index size,

Table 2. Number example of the computational complexity. The last two columns describe the worst case trapdoor size considering the use of 256 or 512 degree polynomials. This is also the size of an encrypted index for a single document.

Keywords	Docs	Additions	Multiplications	WC $\lvert T^{256}_{\mathsf{f}_w}\rvert$	WC $\lvert T^{512}_{\mathsf{f}_w}\rvert$
100	1000	99,000	100,000	87 kB	250 kB
100	5000	495,000	500,000		
250	1000	249,000	250,000	218.75 kB	625 kB
250	5000	1,245,000	1,250,000		

Table 3. Number example of the computational complexity of the SSW scheme. GE(CT) and GE(T) shows the number of group elements per ciphertext and trapdoor, respectively.

Keywords	Docs	Pairings	GE(CT)	GE(T)
100	1000	202,000	202	202
100	5000	1,010,000		
250	1000	502,000	502	502
250	5000	2,510,000		

that has to be stored on the server for a single document. Table 3 shows the computational complexity of the SSW [27] scheme in terms of pairings that have to be computed per search. The last two columns show the number of group elements per ciphertext and trapdoor, respectively.

In Table 4, we compare our scheme to other schemes. The first three rows describe the asymptotic comparison from the perspective of computational complexity of the algorithms. Our Trapdoor algorithm is a constant time operation, since it requires only a table lookup which can be done using a trivial hash function as index. Our BuildIndex algorithm has to process each distinct keyword per document. Thus the complexity is $O(n|\Delta|)$. To search, the server has to perform a constant number (namely, b) of operations for all n documents. Thus the server load is $O(n)$. The server has to store one index per document, so the index size is $O(n)$. The fourth and fifth rows of the table compare the expressiveness of search queries, and the last row compares the security of the schemes.

It is worth noting that the above computational complexity comparison is asymptotic. In practice, different operations make a great difference for the real speed number. In our case, the operations are polynomial additions and multiplications, which are much more efficient than other operations such as pairings. For example, for the SSW scheme [27], a search query in the database needs $n(2v+2)$ composite order pairings. As shown in Table 5, for the proposed scheme, given a database set of size 1000 with a keyword set of size 100, a search takes 47 seconds. However, for the same setting, a search takes 58580 (i.e. $1000 \times 202 \times 0.0058 \times 50$) seconds ($\approx$ 16.3 hours) for the SSW scheme on the same machine, which is 1247$\times$ slower than our proposed scheme. These numbers are based on the performance

Table 4. Computational performance of different search schemes, where n is the number of documents in the database, v the number of words per document, and a is the number of keywords in the trapdoor. The number of distinct words per document is denoted by $|\Delta|$ and $|\mathcal{R}(w)|$ denotes the number of documents containing the keyword w. The asterisk * refers to the use of a so-called *FKS dictionary* introduced by Fredman et al. [8], which reduces the lookup time to $O(1)$.

Properties	SWP [28]	Goh [11]	SSE [7]	SSW [27]	Ours
Compute Trapdoor	$O(1)$	$O(1)$	$O(1)$	$O(v)$	$O(a)$
Compute Indexes	$O(nv)$	$O(n\|\Delta\|)$	$O(n\|\Delta\|)$	$O(nv)$	$O(n\|\Delta\|)$
Search Indexes	$O(nv)$	$O(n)$	$O(\|\mathcal{R}(w)\|)^*$	$O(nv)$	$O(n)$
Conjunctive Search	No	No	No	Yes	Yes
Advanced Search Features	No	No	No	Yes	Yes
Full Security	No	No	No	No	Yes

Table 5. Comparison of the search times of our scheme and Shen et al. scheme. The SSW (prime) column shows the SSW scheme under the assumption that it uses prime order pairing. The SSW (composite) shows a calculated value.

Scheme	small (100/1000)	large (250/5000)
Our ($\alpha = 512$)	47 s	9.9 m
Our ($\alpha = 1024$)	4.8 m	59.8 m
SSW (prime)	19.5 m	4.0 h
SSW (composite)	16.3 h	8.4 d

of a type A symmetric prime order pairing using the PBC [19] library and the fact that a pairing on a 1024-bit composite order elliptic curve can be 50 times slower than in a prime order group [9]. For our comparison this is a conservative estimate since the SSW scheme uses composite order groups, where the order is the product of four primes. Our scheme is more efficient than the SSW scheme.

6 Related Work

Research into the area of search over encrypted data has been a topic of both database community and cryptography community. Starting from the work by Hacigümüş et al. [15], many proposals have been proposed and most of them are centered around the concept of bucketization. However, these proposals have not been analysed in a security model and are usually vulnerable to some attacks. Next, we briefly review some representative SSE schemes which have tried to achieve provable security in some security model.

Searchable encryption can be achieved by *oblivious RAMs* [13,23,24], which hide all information, including the query result, from the server. An oblivious RAM based scheme is not efficient in the sense that it needs a logarithmic number of rounds of interaction for each read and write. SSE schemes improve the efficiency at the cost of revealing *some* information. The first practical scheme for searching in encrypted data in the symmetric setting was proposed by Song, Wagner and Perrig [28] (SWP). SWP uses a special two-layered encryption construct, which is known as a sequential scan. SWP is not secure against statistical analysis across multiple queries and leaks the positions of the queried keywords in a document, thus revealing index information.

Some of the above problems are addressed by Goh [11] by adding a Bloom filter index to each document. The index makes the scheme independent of the document encryption. Goh also introduces the formal indistinguishability against chosen keyword attack (IND-CKA) and a slightly stronger IND-CKA2 security model. Goh notes that the security models for secure indexes do not require the trapdoors to be secure, which is an important requirement for any SSE scheme. Chang and Mitzenmacher [5] propose two index schemes, similar to Goh [11], using pre-built dictionaries. They propose a simulation-based definition of security to guarantee privacy for the indexes and the trapdoors. They achieve a notion of security that is similar to IND-CKA2. As discussed by Curtmola et al. [7], the definition is not only non-adaptive, but can also be trivially satisfied by any SSE scheme, even one that is insecure. Curtmola et al. [7] propose new adversarial models for searchable encryption: a non-adaptive and an adaptive one, and proposed two schemes. The first scheme (SSE-I) is only secure against non-adaptive adversaries but more efficient than the second scheme (SSE-II), which is also secure against adaptive adversaries. Curtmola et al. review existing security definitions for searchable encryption and propose new indistinguishability and simulation-based definitions that address the shortcomings of the existing definitions. At the same time they loosen the character of SSE by allowing the leakage of a user's search pattern. Shen, Shi and Waters [27] (SSW) give formal definitions for *predicate encryption* (PE) and its security. SSW defines *full security* for secret key PE (SKPE). Full security for SKPE is security in the strongest sense possible since the server should only learn the access pattern. In addition, SSW introduces a symmetric key predicate encryption scheme that is proven to be selectively single challenge secure (SCS) which is a relaxed versions of the full security definition. However, they show that only for a special type of query (namely, inner products), SCS security is as good as full security. SSW uses bilinear groups of composite order and is therefore not efficient.

Other works, such as that in [26], have tried to implement SSE in practice. Unfortunately, they usually sacrifice privacy to a certain extent in order to achieve satisfactory performances.

7 Conclusion and Future Work

In this paper, we have proposed the concept of selective document retrieval (SDR) as a cryptographic primitive for outsourcing encrypted data. Compared with symmetric searchable encryption (SSE), an SDR scheme can potentially provide more flexible services and better security guarantees. We described a security model to cover three types of privacy properties, including index privacy, trapdoor privacy, and query result privacy. Note that a secure SSE scheme cannot be trivially extended to provide query result privacy. We have proposed a construction for SDR based on homomorphic encryption and the index construction method by Chang and Mitzenmacher [5]. The construction offers a very flexible framework, and can be adapted very easily to support many useful search features. To evaluate the performance, we have implemented the search algorithm in C based on the symmetric Brakerski-Vaikuntanathan (BV) scheme [3], and the results show that it can be much more efficient than a solution based on existing SSE schemes. In Section 5, we have evaluated the search algorithm of the proposed SDR scheme, but a comprehensive performance study is still needed, in particular for the Retrieve algorithm. The performance of PIR protocols is currently an ongoing research topic for the community, and recently researchers have shown that such protocols can actually be practical [22]. We leave a full discussion of the issue as a future work.

Acknowledgements. We like to thank Michael Naehrig for helpful discussions on the BV scheme and the anonymous reviewers for their valuable comments. This work was done when Qiang Tang worked at the University of Twente.

References

1. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
2. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
3. Brakerski, Z., Vaikuntanathan, V.: Fully Homomorphic Encryption from Ring-LWE and Security for Key Dependent Messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011)
4. Byun, J.W., Rhee, H.S., Park, H.-A., Lee, D.-H.: Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In: Jonker, W., Petković, M. (eds.) SDM 2006. LNCS, vol. 4165, pp. 75–83. Springer, Heidelberg (2006)
5. Chang, Y.-C., Mitzenmacher, M.: Privacy Preserving Keyword Searches on Remote Encrypted Data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
6. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: FOCS 1995: Proceedings of the 36th Annu. IEEE Symposium on Foundations of Computer Science, pp. 41–50 (1995)
7. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions. In: CCS 2006: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM (2006)
8. Fredman, M.L., Komlós, J., Szemerédi, E.: Storing a Sparse Table with 0(1) Worst Case Access Time. J. ACM 31(3), 538–544 (1984)
9. Freeman, D.M.: Converting Pairing-Based Cryptosystems from Composite-Order Groups to Prime-Order Groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010)
10. Gentry, C., Halevi, S., Vaikuntanathan, V.: A Simple BGN-Type Cryptosystem from LWE. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 506–522. Springer, Heidelberg (2010)

11. Goh, E.-J.: Secure Indexes. Cryptology ePrint Archive, Report 2003/216 (2003)
12. Goldreich, O.: Secure Multi-Party Computation. Working draft (October 2002)
13. Goldreich, O., Ostrovsky, R.: Software Protection and Simulation on Oblivious RAMs. J. ACM 43(3), 431–473 (1996)
14. Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Comput. Syst. Sci. 28(2), 270–299 (1984)
15. Hacigümüş, H., Iyer, B., Li, C., Mehrotra, S.: Executing SQL over Encrypted Data in the Database-Service-Provider Model. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, pp. 216–227. ACM (2002)
16. Hart, W.: FLINT: Fast Library for Number Theory, http://www.flintlib.org
17. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can Homomorphic Encryption be Practical? In: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, CCSW 2011, pp. 113–124 (2011)
18. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 319–339. Springer, Heidelberg (2011)
19. Lynn, B.: The Pairing-Based Cryptography library, http://crypto.stanford.edu/pbc
20. Lyubashevsky, V., Micciancio, D., Peikert, C., Rosen, A.: SWIFFT: A Modest Proposal for FFT Hashing. In: Nyberg, K. (ed.) FSE 2008. LNCS, vol. 5086, pp. 54–72. Springer, Heidelberg (2008)
21. Micciancio, D., Regev, O.: Lattice-Based Cryptography, pp. 147–191. Springer (2009)
22. Olumofin, F., Goldberg, I.: Revisiting the Computational Practicality of Private Information Retrieval. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 158–172. Springer, Heidelberg (2012)
23. Ostrovsky, R.: Efficient Computation on Oblivious RAMs. In: Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing, pp. 514–523. ACM (1990)
24. Ostrovsky, R.: Software Protection and Simulations on Oblivious RAMs. PhD thesis. MIT (1992)
25. Ostrovsky, R., Skeith III, W.E.: A Survey of Single-Database Private Information Retrieval: Techniques and Applications. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 393–411. Springer, Heidelberg (2007)
26. Pappas, V., Raykova, M., Vo, B., Bellovin, S.M., Malkin, T.: Private Search in the Real World. In: Zakon, R.H., McDermott, J.P., Locasto, M.E. (eds.) Twenty-Seventh Annual Computer Security Applications Conference, ACSAC 2011, pp. 83–92. ACM (2011)
27. Shen, E., Shi, E., Waters, B.: Predicate Privacy in Encryption Systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
28. Song, D.X., Wagner, D., Perrig, A.: Practical Techniques for Searches on Encrypted Data. In: Proceedings of the 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society (2000)

Additively Homomorphic Encryption with a Double Decryption Mechanism, Revisited

Andreas Peter[1], Max Kronberg[2], Wilke Trei[2], and Stefan Katzenbeisser[1]

[1] Security Engineering Group
Technische Universität Darmstadt and CASED, Germany
{peter,katzenbeisser}@seceng.informatik.tu-darmstadt.de

[2] Arbeitsgruppe Algebra/Geometrie
Universität Oldenburg, Germany
{m.kronberg,wilke.trei}@uni-oldenburg.de

Abstract. We revisit the notion of additively homomorphic *encryption with a double decryption mechanism* (DD-PKE), which allows for additions in the encrypted domain while having a master decryption procedure that can decrypt all properly formed ciphertexts by using a special master secret. This type of encryption is generally considered as a practical way to enforce access control in hierachical organisations where some form of malleability properties are required. Up to now, only two additively homomorphic DD-PKE schemes have been proposed: CS-Lite by Cramer and Shoup (Eurocrypt 2002), and a variant called *BCP* by Bresson, Catalano and Pointcheval (Asiacrypt 2003).

In this work, we argue that the two existing schemes only provide partial solutions for hierarchical organisations. Essentially, this is due to the fact that the master authority, being in possession of the master secret, has no control on the validity of given ciphertexts. We say that the master is unable to "detect invalid ciphertexts", which limits the employment of such schemes in practice. Therefore, we propose the first additively homomorphic DD-PKE scheme which allows the master to detect invalid ciphertexts. In fact, our scheme has the additional property that the master decryption is independent of the users' public keys. Our solution is based on elliptic curves over rings and we prove it to be semantically secure under a DDH-related assumption. Moreover, we give experimental results on the choice of elliptic curves and their effect on the efficiency of our scheme's setup.

Keywords: Public-Key Cryptography, Homomorphic Encryption, Double Decryption Mechanisms, Elliptic Curves, Factoring.

1 Introduction

We consider a concrete example taken from practice that involves a company having many employees (e.g., an insurance company) with a certain hierarchy among them, and in particular with some master authority (e.g., the head of the company) that sits at the top of this hierarchy. Most of the company's data is stored on some central servers where hierachical access control is enforced by using encryption. But it happens occasionally that some employees leave the company or new people are being employed, and so every employee should get her own public and corresponding private keys. In this scenario, the company should be concerned with the following challenges:

– To avoid expensive key management, employees should be able to generate their own key pairs without getting in touch with the master authority.
– If an employee leaves the company or loses her keys (this concerns both the public and the private key), the master authority still wants to be able to recover all data. Hence, the master authority needs some master secret (independent of the employees' individual private keys) that allows to decrypt *any* data stored on the company's servers. Moreover, the master authority should be able to check whether a ciphertext has been encrypted under a given employee's public key. This is relevant, for instance, in the following case: Assume an unavailable employee (for whatever reason, maybe due to quitting) left some important data on the server, e.g., an encryption of an important decision (1 for 'yes' and 0 for 'no'). The master authority needs to know this decision, but at the same time needs to verify whether it was encrypted by the respective employee, i.e., under her public key. In fact, an encryption under the wrong employee's public key might lead the master to a wrong decision.
– Additionally, in practice there is often the requirement that the used cryptosystem has a certain malleability property or is even homomorphic.

The just described scenario is a typical application (cf. [16]) of so-called additively homomorphic *encryption schemes with a double decryption mechanism* (DD-PKE) which combine all the above properties in just one cryptosystem. Roughly speaking, such schemes have two independent, additively homomorphic decryption procedures. Now, because solutions to the described scenario are most wanted in practice, one would expect the existence of many cryptosystems of this type. But in fact, there exist only two such schemes, namely CS-Lite by Cramer and Shoup [8] and a variant called *BCP* by Bresson, Catalano and Pointcheval [7]. Looking at these two schemes in detail, one notices two major weaknesses:

1. In the BCP cryptosystem, in order for the master authority to decrypt a given ciphertext, it has to know the employee's public key under which it was created. This fact contradicts to the requirement that the company does not want to do any complex key management (and in fact simply does not see the public keys in general).
2. Furthermore, both cryptosystems have the drawback that the master authority is unable to check whether a given ciphertext was encrypted under a given public key. This also contradicts the requirements of the above scenario. We note here that the authors of [7] left such "ciphertext validity checks" of the master authority as an open question.

In this work, we propose the *first* additively homomorphic DD-PKE scheme that avoids both just mentioned drawbacks: It is *User-Independent* (i.e., the master decryption procedure is independent of the public keys of the employees/users) and it allows the master to *detect invalid ciphertexts* (i.e., given a ciphertext and a user's public key the master can check whether the ciphertext was encrypted under the given public key).

Our solution is based on elliptic curves over rings $\mathbb{Z}_{N^2}$ where $N = pq$ is some RSA-modulus, and we prove its semantic security under a Decisional Diffie-Hellman (DDH) related assumption on such curves. Finally, we discuss different possible choices of elliptic curves in the setup of our cryptosystem. Since these choices might have an effect on the security of our scheme, we also consider randomly chosen curves (which we require to have an order with at least two large prime factors). For this, one has to rely on a conjecture by Galbraith and McKee [14] about the likelyhood of hitting on such curves. Therefore, we made a substantial number of experiments to get an idea on the efficiency of our setup algorithm for randomly chosen curves. Since there are only a few experimental results on this matter in the literatue, our results might be of independent interest.

Related Work. Since the first efficient, additively homomorphic encryption scheme was proposed by Paillier [24] a lot of follow-up papers appeared in this area (see [10] for a survey). In particular, there were many approaches to construct such schemes by using elliptic curves (see Galbraith's elliptic-curve-based Paillier scheme [13] and the references therein). Another important paper in this context is by Armknecht, Katzenbeisser and Peter [2] who give an easy to use abstract framework and security characterization of such schemes. While we are only interested in additively homomorphic encryption (i.e., it is possible to evaluate the addition of plaintexts over their encryptions without knowledge of the private key), much attention is recently being devoted to the topic of fully homomorphic encryption [17,6], which allows for the evaluation of any circuit over encrypted data without being able to decrypt.

Besides the great many of works on homomorphic encryption, there are several constructions of (non-homomorphic) DD-PKE schemes [16,27]. In this regard, we note that although identity-based encryption [5,25] is related to DD-PKE, therein the master secret is essential in order to generate the users' private keys (in DD-PKE only some publicly known master information is needed, so there is no interaction between the users and the master).

Finally, we mention the only two existing schemes [8] and [7] which are *both* additively homomorphic *and* have a double decryption mechanism.

2 Preliminaries

Notation. We write $x \longleftarrow X$ if X is a random variable or distribution and x is to be chosen randomly from X according to its distribution. In the case where X is solely a set, $x \overset{U}{\longleftarrow} X$ denotes that x is chosen uniformly at random from X. For an algorithm $\mathcal{A}$ we write $x \longleftarrow \mathcal{A}(y)$ if $\mathcal{A}$ outputs x on fixed input y according to

$\mathcal{A}$'s distribution. Sometimes, we need to specify the randomness of a probabilistic algorithm $\mathcal{A}$ explicitly. To this end, we interpret $\mathcal{A}$ as a deterministic algorithm $\mathcal{A}(y, r)$, which has access to random values r.

By a *description* of a finite set X we mean an efficient sampling algorithm (according to some distribution) for the set X. If X is a group, a *description* of X additionally includes the neutral element and a set of efficient algorithms that allow us to perform the usual group operation on X and the inversion of group elements. We abuse notation and write X both for the description and for the set itself. If a description of X is given, we denote sampling from X according to the distribution given by the sampling algorithm of the description by $x \longleftarrow X$.

If $f : X \to Y$ is a mapping between two sets X and Y, we write $\mathrm{dom}(f) = X$ for the *domain* of f and $\mathrm{im}(f)$ for its *image*. In addition, we write $f|_S$ for the *restriction* of f to a subset $S \subseteq X$, i.e. $f|_S : S \to Y$ with $f|_S(s) := f(s)$ for all $s \in S$. If X and Y are groups (additively written), and f is a group homomorphism, we write $\ker(f) := \{x \in X \mid f(x) = 0\}$ for the *kernel* of f. If f is surjective, we write $f^{-1}(y) := \{x \in X \mid f(x) = y\}$ for the *preimage* of y under f for $y \in Y$. Surjective group homomorphisms are also called *group epimorphisms*.

We recall that a public-key encryption scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of a probabilistic polynomial time (PPT) key generation algorithm KeyGen which generates a pair $(\mathsf{pk}, \mathsf{sk})$ of corresponding public and private keys for a given security parameter κ, a PPT encryption algorithm Enc and a deterministic PT decryption algorithm Dec with the usual correctness condition. We denote the set of plaintexts by $\mathcal{P}$, the set of ciphertexts by $\widehat{\mathcal{C}}$, and the *set of all encryptions* (i.e., outputs of the encryption algorithm) by $\mathcal{C}$.

Elliptic Curves over Rings. In this section, we want to recall some facts about elliptic curves over rings. To this end, let R be a commutative unital ring with R^* denoting its group of units. We say that for $a, b \in R$ the equation

$$E : y^2 z = x^3 + axz^2 + bz^3 \tag{1}$$

defines an *elliptic curve E over R* if the *discriminant* $\Delta := 16(4a^3 + 27b^2)$ is a unit in R, i.e., $\Delta \in R^*$. For all triples $(x, y, z) \in R^3$ that satisfy (1), we say that (x, y, z) is *equivalent to* (x', y', z') if there exists $\nu \in R^*$ such that $\nu x = x', \nu y = y'$ and $\nu z = z'$. Indeed this defines an equivalence relation (denoted by $\sim$) on all such triples and we denote equivalence classes by $(x : y : z)$. This relation allows us to define the *set of R-valued points* of E (denoted by $E(R)$) as the set of all equivalence classes $(x : y : z)$ with $x, y, z \in R$ satisfying (1) such that the *ideal I generated by x, y, z* is R, i.e., $I := \{rx + sy + tz \mid r, s, t \in R\} = R$.

It can be shown (see [21, Section 3]) that the usual chord and tangent process on elliptic curves over fields (cf. [26, Chapter III]) yields a group law on $E(R)$ with identity element $\mathcal{O} := (0 : 1 : 0)$ *if* R has the property that every projective R-module of rank one is free. For our work, it suffices to consider this case, since we will only work over finite rings which have this property. Therefore, we will from now on restrict our attention to finite rings R.

It should be noted that there are explicit and efficient formulae to perform the group law on $E(R)$ which we do not want to recall here due to space limitations (instead we refer to [21,13]). Furthermore, we recall that the Chinese Remainder Theorem on $\mathbb{Z}_N$ (where $N = pq$ is some RSA-modulus) implies natural reduction maps from $E(\mathbb{Z}_N)$ to $E(\mathbb{Z}_p)$ and $E(\mathbb{Z}_q)$. It follows that $E(\mathbb{Z}_N) \cong E(\mathbb{Z}_p) \times E(\mathbb{Z}_q)$ (see [13]).

There are a few other facts in the case where $R = \mathbb{Z}_{N^2}$ for some RSA-modulus $N = pq$, that are of particular interest to us, which follow from the p-adic theory of elliptic curves, and we refer the reader to [13] and [26] for details:

1. $\#E(\mathbb{Z}_{N^2}) = N\#E(\mathbb{Z}_N) = N\#E(\mathbb{Z}_p)\#E(\mathbb{Z}_q)$.
2. $P_i := (Ni : 1 : 0) \in E(\mathbb{Z}_{N^2})$ with $mP_i = P_{mi}$ for all $m \in \mathbb{Z}_N$.
3. $NP_1 = \mathcal{O}$.

Finally, we state that if the factorization of N is not known, the *Decisional Diffie-Hellman Problem* is believed to be hard for elliptic curves over $\mathbb{Z}_{N^2}$. It is defined as follows: Given a random point Q of large order k (meaning that k has about the same size as N), points rQ, sQ and tQ $(r, s, t \in \mathbb{Z}_k)$, it is computationally infeasible to decide whether $t = rs \bmod k$ or not. We denote this problem by $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$. We stress that even if the factorization of N is known, $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$ is still believed to be hard for *randomly* chosen elliptic curves over $\mathbb{Z}_{N^2}$. We will see later that, as in the case where N is prime, if the factorization of N is known, then $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$ can be solved efficiently for *pairing-friendly* curves (here, we mean curves where the reduced Weil or Tate pairing over $\mathbb{Z}_p$ and $\mathbb{Z}_q$ can be efficiently computed). For a detailed discussion on pairing-friendly elliptic curves over fields, we refer to [11], and to [15] when working over rings.

3 (User-Independent) Double Decryption

We start by recalling what it means for an encryption scheme to have a double decryption mechanism. We do this along the lines of Galindo and Herranz's work [16].

Definition 1. *A public key encryption scheme* with a double decryption mechanism (DD-PKE) *is a tuple* (Setup, KeyGen, Enc, Dec, mDec) *of PPT algorithms such that*

Setup: $\mathsf{Setup}(\kappa)$ *takes a security parameter κ as input and outputs a tuple* $(\mathsf{PP}, \mathsf{MK})$ *where* PP *contains the public system parameters (particularly includes descriptions of the plaintext space $\mathcal{P}$ and the ciphertext space $\widehat{\mathcal{C}}$), and* MK *is the master secret key which is only known to the master entity.*

Key Generation: $\mathsf{KeyGen}(\mathsf{PP})$ *takes the system's public parameters* PP *as input and outputs a pair of public/private keys* $(\mathsf{pk}, \mathsf{sk})$ *to a user.*

Encryption: $\mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m)$ *takes the public parameters* PP*, a user's public key* pk *and a message $m \in \mathcal{P}$ as input and outputs a ciphertext $c \in \mathcal{C}$.*

User Decryption: $\mathsf{Dec}_{(\mathsf{PP},\mathsf{sk})}(c)$ *takes the public parameters* PP*, a user's secret key* sk *and a ciphertext $c \in \widehat{\mathcal{C}}$ as input and outputs either a plaintext $m \in \mathcal{P}$ or the special symbol $\perp$.*

Master Decryption: $\mathsf{mDec}_{(\mathsf{PP},\mathsf{MK},\mathsf{pk})}(c)$ *takes the public parameters* PP*, the master secret key* MK*, a user's public key* pk *and a ciphertext* $c \in \widehat{\mathcal{C}}$ *as input and outputs either a plaintext* $m \in \mathcal{P}$ *or the symbol* $\perp$*.*

For such schemes, we require the usual correctness condition in public key encryption schemes both for the user decryption and the master decryption. It should be noted that by combining the system's public parameters in the user's public keys, we can think of a DD-PKE scheme as being a usual encryption scheme that additionally has a master decryption procedure (that uses the master secret key). Also, we stress that the notion of semantic security is exactly the same as that for usual public-key encryption schemes. Furthermore, it is noteworthy that the key generation algorithm KeyGen does *not* get the master secret MK as input.

Next, we introduce the notion of *User Independence* in the context of such DD-PKE schemes, which basically means that the master entity can decrypt any given ciphertext even without knowing the corresponding receiver (i.e., the user's public key under which it has been encrypted). In other words this means that the master decryption is independent of the users.

Definition 2. *A DD-PKE scheme is* user-independent *(UI-DD-PKE) if the master decryption does* not *get the user's public key as an input, i.e., it only gets the system's public parameters, the master secret and a ciphertext as input.*

4 An Additively Homomorphic UI-DD-PKE Scheme

We introduce a new public key cryptosystem with a simple structure that combines a couple of unique properties in a single scheme. Due to its many properties, we will restrict our attention to the scheme's formal definition and proof of correctness in this section, and deal with its properties in the next section. The semantic security of the scheme will be proven in Section 6. In order to formally define our cryptosystem, we need the following two facts:

Proposition 1. *If* $N = pq$ *is some RSA-modulus, i.e.,* p *and* q *are primes of about the same bit length* κ*, then there is an efficient construction of elliptic curves* $E : y^2z = x^3 + axz^2 + bz^3$ *over* $\mathbb{Z}_{N^2}$ *such that* $M := lcm(\#E(\mathbb{Z}_p), \#E(\mathbb{Z}_q))$ *has at least two large (of about the same size as* p *and* q*) prime factors.*

Proof. There are three different methods to construct such elliptic curves, which have direct influence on the system's efficiency and applicability. We therefore put the proof of this proposition in a section on its own (see Section 7). □

Lemma 1. *As in Proposition 1, let* $M \in \mathbb{N}$ *have at least two large prime factors (of about* κ *bits). If* $\pi(M)$ *denotes the product of all small prime factors (including multiples) of* M*, then*

$$\Pr_{s \xleftarrow{U} \Pi(M)} [\gcd(s, M) \neq 1] \text{ is negligible in } \kappa,$$

where $\Pi(M) := \{s \in \mathbb{Z}_{N^2} \setminus \{0\} \mid \gcd(s, \pi(M)) = 1\}$*.*

Proof. Let $L(M) = \prod_{i=1}^{r} p_i^{\nu_i}$ be the product ($r \geq 2$) of all large (of about κ bits) prime factors in M, i.e., $M = \pi(M) \cdot L(M)$. By definition, we have that $\Pr[\gcd(s, M) \neq 1 \text{ for } s \in \Pi(M)] = \Pr[s \in \mathbb{Z}_{L(M)} \setminus \mathbb{Z}^*_{L(M)}]$. But if φ denotes Euler's totient function, we have

$$\#\mathbb{Z}^*_{L(M)} = \varphi(L(M)) = \prod_{i=1}^{r}(p_i - 1)p_i^{\nu_i - 1}, \text{ and hence } \frac{\#\mathbb{Z}^*_{L(M)}}{\#\mathbb{Z}_{L(M)}} = \prod_{i=1}^{r}\left(1 - \frac{1}{p_i}\right).$$

However, the fractions $\frac{1}{p_i}$ are negligible in κ, and so the product of all these is negligibly close to 1. Therefore, we have

$$\Pr[s \in \mathbb{Z}_{L(M)} \setminus \mathbb{Z}^*_{L(M)}] = 1 - \frac{\#\mathbb{Z}^*_{L(M)}}{\#\mathbb{Z}_{L(M)}} = 1 - (1 - \mathtt{negl}(\kappa)) = \mathtt{negl}(\kappa),$$

where $\mathtt{negl}(\kappa)$ denotes a negligible function in κ. □

Definition 3 (The Cryptosystem).

Setup: $\mathsf{Setup}(\kappa)$ *computes an RSA-modulus* $N = pq$ *where* p *and* q *are primes of about the same bit length* κ *and constructs an elliptic curve* $E : y^2z = x^3 + axz^2 + bz^3$ *over* $\mathbb{Z}_{N^2}$ *such that* E *has the properties as described in Proposition 1. Furthermore, it chooses a point* $Q = (x : y : z) \in E(\mathbb{Z}_{N^2})$ *whose order divides* $M = lcm(\#E(\mathbb{Z}_p), \#E(\mathbb{Z}_q))$.[1]
It outputs the public parameters $\mathsf{PP} := (N, \pi(M), a, b, Q)$ *and the master secret key* $\mathsf{MK} := M$*. The plaintext space is* $\mathcal{P} = \mathbb{Z}_N$ *and the ciphertext space is* $\widehat{\mathcal{C}} = \langle Q \rangle \times \langle Q, P_1 \rangle$.

Key Generation: $\mathsf{KeyGen}(\mathsf{PP})$ *chooses* $s \in \mathbb{Z}^*_M$ *at random and computes* $R := sQ$*. This can be done by sampling* $s \in \Pi(M)$ *(which is possible as* $\pi(M)$ *is included in* PP*), since then* $s \in \mathbb{Z}^*_M$ *holds with overwhelming probability by Lemma 1.*[2] *It outputs the user's public key* $\mathsf{pk} := R$ *and secret key* $\mathsf{sk} := s$.

Encryption: $\mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m)$ *chooses a random value* $r \in \mathbb{Z}_{N^2}$ *and computes the ciphertext* (A, B) *as*

$$A := rQ \text{ and } B := rR + P_m.[3]$$

User Decryption: $\mathsf{Dec}_{(\mathsf{PP},\mathsf{sk})}(A, B)$ *outputs*

$$m = \frac{x(B - sA)}{N}.$$

Master Decryption: $\mathsf{mDec}_{(\mathsf{PP},\mathsf{MK})}(A, B)$ *outputs*

$$m = \frac{x(MB)}{N} M^{-1} \bmod N.$$

[1] This can be done by taking a random point $Q' = (x' : y' : z') \in E(\mathbb{Z}_{N^2})$ and setting $Q := NQ'$. See also Section 7.

[2] We note that by using Hasse's bound on $\#E(\mathbb{Z}_p)$ and $\#E(\mathbb{Z}_q)$, we have $M \leq \#E(\mathbb{Z}_p)\#E(\mathbb{Z}_q) \leq N^2$.

[3] We note that if we forget about the first component A of our ciphertexts, then the encryption looks very similar to Galbraith's elliptic-curve-based Paillier scheme [13].

Concerning the *correctness* of both decryption procedures, we see that

$$\mathsf{Dec}_{(\mathsf{PP},\mathsf{sk})}(\mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m)) = \frac{x(rR + P_m - srQ)}{N} = m$$

and

$$\mathsf{mDec}_{(\mathsf{PP},\mathsf{MK})}(\mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m)) = \frac{x(M(rR + P_m))}{N} M^{-1} \bmod N = m$$

by using the fact that $\operatorname{ord}(Q)$ divides M, so $MR = sMQ = \mathcal{O}$.

Remark 1. 1. We stress that the knowledge of M is polynomial-time equivalent to the knowledge of the factorization of N (cf. [23, Theorem 10]). Therefore, it is computationally infeasible to compute the master secret key MK from the public parameters PP.
2. It is also computationally infeasible to compute the user's secret key from its public key under the assumption that the *Discrete Logarithm Problem* (DLP) is hard in $E(\mathbb{Z}_{N^2})$.
3. Without knowledge of the factorization of N it is computationally infeasible to find a point Q' on the curve (that differs from linear combinations of the publicly known points Q, R and P_1), because one would need to solve polynomial equations in $\mathbb{Z}_{N^2}$.
4. We notice that the users' public keys are not needed in the master decryption algorithm, and so we have successfully defined a UI-DD-PKE scheme.
5. Finally, we note that the master decryption never fails on a given ciphertext $c \in \widehat{\mathcal{C}}$, and so it always outputs a message $m \in \mathcal{P}$. This is different for the user decryption. It will output $\perp$ if $x(B - sA)$ is not divisible by N. We will show in the next section (Property 2) that this happens if and only if the given ciphertext is invalid, which users can efficiently detect.

5 Properties of the Cryptosystem

We start with two properties of the cryptosystem that are independent of the choice of elliptic curves in the setup algorithm as long as these curves satisfy the properties of Proposition 1.

Property 1. The cryptosystem is additively homomorphic, i.e.,

$$\mathsf{Dec}_{(\mathsf{PP},sk)}(\mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m_1) + \mathsf{Enc}_{(\mathsf{PP},\mathsf{pk})}(m_2)) = m_1 + m_2.$$

Together with item 4 of Remark 1 this means that the scheme is an additively homomorphic UI-DD-PKE scheme.

Proof. Let $m_1, m_2 \in \mathbb{Z}_N$ be two plaintexts encrypted as (A_1, B_1) and (A_2, B_2), respectively. Then $(A, B) := (A_1, B_1) + (A_2, B_2)$ is a ciphertext of $m := m_1 + m_2$ since

$$\begin{aligned}\frac{x(B - sA)}{N} &= \frac{x(r_1R + P_{m_1} + r_2R + P_{m_2} - sr_1Q - sr_2Q)}{N} \\ &= \frac{x(P_{m_1+m_2})}{N} = \frac{(m_1 + m_2)N}{N} = m_1 + m_2.\end{aligned}$$

□

Property 2. Users can detect invalid ciphertexts.

Proof. By definition (see also item 3 of Remark 1), a ciphertext c is of the form $(A, B) = (rQ, tQ + P_m) \in \langle Q \rangle \times \langle Q, P_1 \rangle$ (recall that $mP_1 = P_m$ and $\mathrm{ord}(P_1) \mid N$; cf. Section 2). If s denotes a user's private key, we know that a ciphertext c is valid if and only if $t = rs \bmod \mathrm{ord}(Q)$, which in turn is equivalent to saying that $B - sA = P_m$ (recall that $P_m \notin \langle Q \rangle$ for all $0 \neq m \in \mathbb{Z}_N$). □

There are a couple of interesting properties of the cryptosystem that depend on the actual choice of the elliptic curve in the setup algorithm. We start with the detection of invalid ciphertexts for the master entity.

Property 3. If $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$ is hard in $E(\mathbb{Z}_{N^2})$ (even when the factorization of N is known), then the master entity, when given a user's public key, *cannot* decide whether a given ciphertext is a valid encryption under this public key or not.

Proof. Assume that the master can detect invalid ciphertexts which are, by definition, of the form $(A, B) = (rQ, tQ + P_m) \in \langle Q \rangle \times \langle Q, P_1 \rangle$. Then we can use this detection algorithm to solve $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$ as follows: Given a $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$-tuple (Q, rQ, sQ, tQ), we just check the ciphertext $(A, B) = (rQ, tQ)$ for validity under the public key sQ. Clearly, we have:

$$(A, B) \text{ is valid} \iff (Q, rQ, sQ, tQ) \text{ is a valid } \mathsf{DDH}_{\mathbb{Z}_{N^2}}\text{-tuple.}$$

□

As explained in the Introduction, there are applications where the master entity should be able to check ciphertexts for validity as well. This is where pairings come into play. We will see that our cryptosystem is actually a nice application of *hidden pairings* – a notion introduced by Dent and Galbraith [9]. Therein, they present an identification scheme as a cryptographic application which was the only interesting application known until now. Unfortunately, since our scheme uses elliptic curves with certain properties in a non-black-box way, we cannot use the construction of a "hidden pairing"-group of [9] directly, but need to construct our own. Our construction is given in the following result that we prove in Section 7:

Lemma 2. *There is an efficient construction of an elliptic curve E over $\mathbb{Z}_{N^2}$ with properties as in Proposition 1 together with a point $Q \in E(\mathbb{Z}_{N^2})$ of large order dividing M such that*

1. *if Q_1 and Q_2 denote the natural reductions of Q to $E(\mathbb{Z}_p)$ and $E(\mathbb{Z}_q)$, respectively, we have that $ord(Q) = lcm(ord(Q_1), ord(Q_2))$*
2. *we can efficiently compute the 'reduced' Tate pairings τ_p and τ_q on E over $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively.*

Since the Tate pairings τ_p and τ_q can only be computed if the factorization of N is known, they are called *hidden pairings*. Concerning the security of elliptic curves with properties as in the Lemma, we refer the reader to [15] and [9]. This Lemma has an interesting consequence on our cryptosystem:

Property 4. Let Q be a point on an elliptic curve E over $\mathbb{Z}_{N^2}$ as in Lemma [2]. If our cryptosystem uses E and Q in its public parameters, then the master entity can detect invalid ciphertexts under a given user's public key.

Proof. Let $R = sQ$ be a user's public key and let $(A, B) = (rQ, tQ + P_m) \in \langle Q \rangle \times \langle Q, P_1 \rangle$ be a ciphertext. In order to check the validity of (A, B) under R, the master entity first uses the master secret M to compute the plaintext m (by using mDec). Since the master knows the factorization of N, it can now compute the reductions modulo p of Q, R, A and $T := B - P_m = tQ$ in $E(\mathbb{Z}_p)$ which we denote by Q_1, R_1, A_1 and T_1, respectively. Additionally, let $Q_2, R_2, A_2, T_2 \in E(\mathbb{Z}_q)$ be the respective reductions modulo q. Since the master can efficiently compute the 'reduced' Tate pairings τ_p and τ_q, respectively, it can check whether (Q_1, R_1, A_1, T_1) and (Q_2, R_2, A_2, T_2) are valid DDH-tuples in $E(\mathbb{Z}_p)$ and $E(\mathbb{Z}_q)$, respectively, in the usual way (see [12] and [22]). We have the relation that (Q_1, R_1, A_1, T_1) is a valid DDH-tuple in $E(\mathbb{Z}_p)$ if and only if $t = sr \bmod \mathrm{ord}(Q_1)$. An analogous relation holds for the prime q. Together, the Chinese Remainder Theorem over $\mathbb{Z}_{\mathrm{ord}(Q)}$ yields that (Q_1, R_1, A_1, T_1) and (Q_2, R_2, A_2, T_2) are valid DDH-tuples over their respective prime fields if and only if $t = rs \bmod \mathrm{ord}(Q)$ which in turn holds if and only if (A, B) is a valid ciphertext under R. □

6 Semantic Security

Considering the fact that our cryptosystem is an additive variant of the ElGamal cryptosystem, it is rather obvious that it is semantically secure under the $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$-assumption. Therefore, and due to space limitations, we only give a proof sketch of this fact here: Proving semantic security of additively homomorphic cryptosystems boils down to proving that a random encryption is computationally indistinguishable from an encryption of 0 (e.g., Armknecht et al. [2]). In our cryptosystem, a random encryption has the form $(rQ, rR + P_m)$ with randomness $r \in \mathbb{Z}_{N^2}$ and random message $m \in \mathbb{Z}_N$. An encryption of 0, on the other hand, has the form (rQ, rR) for randomness $r \in \mathbb{Z}_{N^2}$. Now, if we write $X = rQ$ and $S = rR + P_m$ for randomness $r \in \mathbb{Z}_{N^2}$ and random message $m \in \mathbb{Z}_N$, we see that semantic security states: Given points $X, R \in \langle Q \rangle$ and given a random point S, decide whether $\log_Q(S) = \log_Q(X) \log_Q(R)$. This problem is the $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$-problem, except that S is chosen from a larger group (and not only from $\langle Q \rangle$). However, $\mathsf{DDH}_{\mathbb{Z}_{N^2}}$ reduces to this more general problem.

In practice, from an adversary's point of view the situation is even worse, since without knowledge of the factorization of N it is extremely hard to find a point Q' on the curve at all (that differs from linear combinations of the publicly known points Q, R and P_1), because one would need to solve polynomial equations in $\mathbb{Z}_{N^2}$. It should be mentioned though that the security highly depends on the order of the point Q. Therefore, one should always take great care in the setup of the cryptosystem that the point Q really has large order (of about the same size as the prime factors of N).

Finally, we note that concerning the size of the security parameter of our scheme, we need to ensure that the bit length of the primes p and q is roughly

512 (at least). This yields a 1024 bit RSA-modulus and so we can assume that factoring such a large number is indeed hard in practice. Since solving discrete logarithms on elliptic curves over prime fields is assumed hard if the bit length of the order of the underlying prime field is about 180, having 512 bits here makes it reasonable to assume that the DLP is indeed hard on our chosen curves. Such a parameter setting is similar to the settings of [13] and [9], where it is argued that one can assume a high level of security while having efficient group operations on the curve at the same time.

7 Concrete Setup of the System's Parameters

The basic goal of this section is to prove Proposition 1 and Lemma 2, i.e., to give efficient constructions of elliptic curves with the properties as described in the respective claim. Since curves satisfying Lemma 2 will also satisfy Proposition 1, we start with the latter (Method 1) and then look at which of these curve additionally satisfy the Lemma (Methods 2 and 3).

Method 1: Random Curves. Given a security parameter κ (which in practice will be of size 512), the fundamental idea is to choose two distinct, random primes p and q of about κ bits (so $N = pq$ is our RSA-modulus) together with two random elliptic curves E_1 and E_2 over $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively. We require that both $E_1(\mathbb{Z}_p)$ and $E_2(\mathbb{Z}_q)$ have at least one large prime factor (of about κ bits) – so we discard all curves not having this property and repeat choosing random curves until we find two suitable elliptic curves. Then, by using standard techniques (i.e., considering E_1 and E_2 over $\mathbb{Z}_{p^2}$ and $\mathbb{Z}_{q^2}$, respectively (cf. Lemma 3), and then using the Chinese Remainder Theorem), we construct an elliptic curve E over $\mathbb{Z}_{N^2}$ such that $M := \operatorname{lcm}(\#E(\mathbb{Z}_p), \#E(\mathbb{Z}_q))$ has at least two large prime factors.

We remark that concerning the security of our cryptosystem, this way of constructing the elliptic curves prevents an attacker to exploit any particular structure of the used elliptic curve.

Likelyhood of hitting on such curves. One problem with this approach concerns the likelyhood of hitting on such curves by random sampling given a prime p. Since there is no final answer to this question in theory, we have to rely on a conjecture by Galbraith and McKee [14]: First, let us only consider elliptic curves E with prime order. It is conjectured that

$$\Pr[\#E(\mathbb{Z}_p) \text{ is prime}] \text{ is asymptotic to } c_p \frac{1}{\log p} \text{ as } p \to \infty,$$

where

$$c_p = \frac{2}{3} \prod_{l>2} \left(1 - \frac{1}{(l-1)^2}\right) \prod_{2<l|p-1} \left(1 + \frac{1}{(l+1)(l-2)}\right)$$

and the probability is over all random primes p and $(a, b) \xleftarrow{U} \mathbb{Z}_p^2 \setminus \{(a, b) \in \mathbb{Z}_p^2 \mid 4a^3 + 27b^2 = 0\}$. We ran some numerical tests ourselves (see Table 1) which confirm the conjecture in practice.

Table 1. Numerical probability of hitting on a curve with prime order

Bit length of p	64	128	192	256
$\Pr[\#E(\mathbb{Z}_p)$ is prime$]$	1.17 %	0.58 %	0.38 %	0.27 %

As we have discussed before, we actually do not need the curve to have a large prime order, but only a nearly prime order. Therefore, we can optimize our search for elliptic curves by using a result by Lenstra [20] that small prime factors appear with a high probability. The idea is to fix a set S of small primes and allow $\#E(\mathbb{Z}_p)$ to be divisible by powers of $s \in S$. This increases the probability to hit on a curve with a large prime dividing the order by a huge factor (e.g., for orders of the form 2^k·prime this factor is about 3, while for orders of the form $2^k \cdot 3^l$·prime the factor is about 5.5 in our numerical results). This was also conjectured by Galbraith and McKee in [14] and our numerical tests give evidence for this conjecture (cf. Table 2).

Table 2. Numerical probability of hitting on a curve with nearly prime order

Bit length of p	64	128	192	256
$\Pr[\#E(\mathbb{Z}_p) = 2^k \cdot \text{prime}]$	3.61 %	1.78 %	1.28 %	0.90 %
$\Pr[\#E(\mathbb{Z}_p) = 2^k \cdot 3^l \cdot \text{prime}]$	6.58 %	3.06 %	2.23 %	1.45 %

Concerning the efficiency of constructing curves as in Proposition 1, our experiments show that for an RSA-modulus of 512 bits (i.e., two primes of about 256 bits) it takes roughly 15 minutes using MAGMA on a single core of an Intel Xeon running at 2.5 GHz. For an 1024 bit RSA-modulus, it takes approximately 13 hours per curve. Allowing primes of up to three dividing the group order, we were able to generate five pairs of elliptic curves in approximately two days, while allowing prime factors of up to 13, this time halves (cf. Table 3). Since the Setup algorithm of our cryptosystem needs to be run only once, such an efficiency is reasonable in practice.

Table 3. Numerical results for the runtime of the Setup algorithm for $\log p = 512$ and 5 keys generated, where $S := \{\text{prime } \mathfrak{p} \mid \mathfrak{p} \text{ is allowed to divide } \#E(\mathbb{Z}_p)\}$

	$S = \{2, 3\}$	$S = \{2, 3, 5\}$	$S = \{2, 3, 5, 7, 11\}$	$S = \{2, 3, 5, 7, 11, 13\}$
Time	2d 6h 4m 20s	23h 15m 10s	2d 5h 49m 41s	1d 4h 43m 17s
Tested Curves	1298	552	1266	687

Performing our cryptosystem's setup. Recall that for a high level of security it is not enough to find suitable elliptic curves, we should also choose the point $Q \in E(\mathbb{Z}_{N^2})$ in the Setup algorithm to be of large order dividing $M = \mathrm{lcm}(\#E(\mathbb{Z}_p), \#E(\mathbb{Z}_q))$ (cf. Section 6). The following two lemmata can be used in order to do this:

Lemma 3. *For a prime $p > 3$ and an elliptic curve E over $\mathbb{Z}_p$, we can efficiently construct an elliptic curve E' over $\mathbb{Z}_{p^2}$ such that $E'(\mathbb{Z}_{p^2})$ has order $\#E(\mathbb{Z}_p) \cdot p$ and the reduction from $\mathbb{Z}_{p^2}$ to $\mathbb{Z}_p$ induces a group homomorphism from $E'(\mathbb{Z}_{p^2})$ to $E(\mathbb{Z}_p)$.*

Proof. Let E be given by the short Weierstrass equation $y^2z = x^3 + axz^2 + bz^3$ over $\mathbb{Z}_p$. The existence of E' such that it reduces to E is simple, because it is sufficient to define E' by the same Weierstrass equation. Since the discriminant of E is invertible modulo p it also is modulo p^2, thus E' is an elliptic curve. Due to the geometric definition of the elliptic curve group law, the existence of the induced group homomorphism is obvious. It is left to be proven that this homomorphism is surjective.

Fix any finite point $P = (x_0 : y_0 : 1) \in E(\mathbb{Z}_p)$, then y_0 is a solution to the polynomial equation $0 = y^2 - (x_0^3 + ax_0 + b)$. In the case $y_0 \not\equiv 0 \pmod p$ there is a unique integer $0 \le k < p$ such that $(y_0 + kp)^2 - (x_0^3 + ax_0 + b) \equiv 0 \pmod{p^2}$ by Hensel's lifting lemma. The new point $(x_0 : y_0 + kp : 1)$ obviously reduces to the initial point. In the case $y_0 \equiv 0 \pmod p$ we know that x_0 has been a solution for $0 = x^3 + ax + b \pmod p$. Since this polynomial cannot have any double roots we can find a solution $x_0 + kp$ for the same equation modulo p^2. This proves surjectivity.

To compute the order of $E'(\mathbb{Z}_{p^2})$ it is sufficient to compute the kernel of the reduction to $E(\mathbb{Z}_p)$. Obviously there are exactly p points $(kp : 1 : 0)$ for $0 \le k < p$ on $E'(\mathbb{Z}_{p^2})$ that reduce to the point at infinity on $E(\mathbb{Z}_p)$. Thus $\#E'(\mathbb{Z}_{p^2}) = \#E(\mathbb{Z}_p) \cdot p$ holds due to the homomorphism theorem. □

Lemma 4. *Let $p > 3$ be a prime, E be an elliptic curve defined over $\mathbb{Z}_p$ and $P \in E(\mathbb{Z}_p)$ a point with $\gcd(ord(P), p) = 1$. Then the curve $E'(\mathbb{Z}_{p^2})$ constructed as in Lemma 3 contains a point P' of order $ord(P)$.*

Proof. Let Q' be any preimage of P under the reduction map (Q' can be constructed following the proof of Lemma 3). By the homomorphism theorem, we have $\mathrm{ord}(P) \mid \mathrm{ord}(Q')$. Multiplying both points with p permutes the subgroup generated by P on $E(\mathbb{Z}_p)$ and $P' := pQ'$ has order $\mathrm{ord}(P') = \mathrm{ord}(pP) = \mathrm{ord}(P)$ since the order of P is coprime to p. □

Now, the construction of a point $Q \in E(\mathbb{Z}_{N^2})$ with large order dividing M, where E is a random curve such that M has at least two large prime factors (as in Proposition 1), works as follows:

1. Choose a random RSA-modulus $N = pq$ and random elliptic curves $E_1(\mathbb{Z}_p)$, $E_2(\mathbb{Z}_p)$ with nearly prime order as described before.
2. Pick points $P_1 \in E_1(\mathbb{Z}_p)$ and $P_2 \in E_2(\mathbb{Z}_q)$ of high order coprime to p and q.

3. Apply Lemma 3 and 4 to construct elliptic curves E_1' and E_2' with points $P_1' \in E_1'(\mathbb{Z}_{p^2})$ and $P_2' \in E_2'(\mathbb{Z}_{q^2})$ of high order.
4. Use the Chinese Remainder Theorem to merge E_1' and E_2' to a single curve E defined over the ring $\mathbb{Z}_{N^2}$. The lift Q of P_1' and P_2' will have order $\mathrm{lcm}(\mathrm{ord}(P_1), \mathrm{ord}(P_2))$ and is the point used for the public parameters PP.

Method 2: Supersingular Curves. A more efficient way to construct elliptic curves that satisfy Proposition 1 is by using supersingular curves E and particular RSA-moduli $N = pq$. For such curves it is known that over a prime field $\mathbb{Z}_p$ we have $\#E(\mathbb{Z}_p) = p + 1$. We note that the following discussion can be done for arbitrary supersingular elliptic curves, however, we restrict our attention to the following family of curves:

Lemma 5 (see [19]). *Let p be an odd prime with $p \equiv 2 \pmod 3$ and let $0 \neq b \in \mathbb{Z}_p$. Consider the elliptic curve $E : y^2 = x^3 + b$. Then, $E(\mathbb{Z}_p)$ is cyclic and $\#E(\mathbb{Z}_p) = p + 1$.*

So if we start with a *strong* prime p (i.e., $p + 1$ is not smooth) with $p \equiv 2 \pmod 3$ and setting E to be the curve given by the equation $y^2 = x^3 + b$ for some $0 \neq b \in \mathbb{Z}_p$, we ensure a large factor in $\#E(\mathbb{Z}_p) = p + 1$. To construct a strong prime p fulfilling the congruence condition it is possible to take a prime p' of the desired bit length κ such that $p := 6p' - 1$ is also prime.

Now, by using Lemmas 3 and 4, we can construct an elliptic curve together with a point Q of high order suitable for our cryptosystem in exactly the same way as we did in Method 1 (items $2 - 4$ in the construction therein). We remark that constructing the elliptic curves in this way gives a very fast and easy setup of our system.

Additional property: Hidden pairing. Since supersingular elliptic curves have an embedding degree of at most $k = 6$, they allow for an efficient evaluation of the 'reduced' Tate pairing, which can then be used to solve DDH-challenges [12,22]. Therefore, our just constructed elliptic curve E over $\mathbb{Z}_{N^2}$ has a hidden pairing [9], and we can efficiently solve DDH if the factorization of N is known. This proves Lemma 2.

Method 3: Complex Multiplication. The CM method [3] allows us to construct elliptic curves E together with primes p and q such that E satisfies Proposition 1. Even more, by using extended algorithms [11], it is possible to construct E over $\mathbb{Z}_{N^2}$ such that it has a small embedding degree over the prime fields $\mathbb{Z}_p$ and $\mathbb{Z}_q$. This yields another construction satisfying Lemma 2.

8 Conclusions

We presented a new additively homomorphic UI-DD-PKE scheme that combines many interesting properties in just one scheme. Most importantly, by choosing the system's parameters (i.e., the elliptic curves) appropriately, our scheme is the first that allows the master entity to check for invalid ciphertexts, additionally to being a UI-DD-PKE scheme. Such a cryptosystem has practical relevance in hierarchical organisations, e.g., in order to reduce key management, or to deal with the problem of key loss. Additionally, the ability to check for invalid ciphertexts might be useful in electronic voting systems where some form of "after the fact" validity checks of votes are required [1]. Finally, we note that due to its ElGamal-like structure, our cryptosystem is likely to be *anonymous* [4] and hence it would be interesting to investigate the effect of the double decryption mechanism on known constructions such as *group encryption* [18]. Further potential future work includes an analysis of the MOV-attack [22] in hidden pairing scenarios, and the possibilities of extracting the randomness used to encrypt a message in our scheme, which would probably yield a practical trapdoor discrete logarithm group.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX Security Symposium, pp. 335–348. USENIX Association (2008)
2. Armknecht, F., Katzenbeisser, S., Peter, A.: Group homomorphic encryption: characterizations, impossibility results, and applications. Designs, Codes and Cryptography, 1–24, doi:10.1007/s10623-011-9601-2
3. Atkin, A.O.L., Morain, F.: Elliptic curves and primality proving. Math. Comp. 61, 29–68 (1993)
4. 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)
5. Boneh, D., Franklin, M.K.: Identity-based encryption from the weil pairing. SIAM J. Comput. 32(3), 586–615 (2003)
6. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: FOCS, pp. 97–106. IEEE (2011)
7. Bresson, E., Catalano, D., Pointcheval, D.: A Simple Public-Key Cryptosystem with a Double Trapdoor Decryption Mechanism and Its Applications. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 37–54. Springer, Heidelberg (2003)
8. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
9. Dent, A.W., Galbraith, S.D.: Hidden Pairings and Trapdoor DDH Groups. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 436–451. Springer, Heidelberg (2006)
10. Fontaine, C., Galand, F.: A survey of homomorphic encryption for nonspecialists. EURASIP J. Inf. Secur. 2007, 15:1–15:15 (2007)
11. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. J. Cryptology 23(2), 224–280 (2010)
12. Frey, G., Rück, H.G.: A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. Math. Comput. 62, 865–874 (1994)
13. Galbraith, S.D.: Elliptic curve paillier schemes. J. Cryptology 15(2), 129–138 (2002)

14. Galbraith, S.D., McKee, J.F.: The probability that the number of points on an elliptic curve over a finite field is prime. Journal of the LMS 62(03), 671–684 (2000)
15. Galbraith, S.D., McKee, J.F.: Pairings on Elliptic Curves over Finite Commutative Rings. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 392–409. Springer, Heidelberg (2005)
16. Galindo, D., Herranz, J.: On the security of public key cryptosystems with a double decryption mechanism. Inf. Process. Lett. 108(5), 279–283 (2008)
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178. ACM (2009)
18. Kiayias, A., Tsiounis, Y., Yung, M.: Group Encryption. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 181–199. Springer, Heidelberg (2007)
19. Koyama, K., Maurer, U.M., Okamoto, T., Vanstone, S.A.: New Public-Key Schemes Based on Elliptic Curves over the Ring Z_n. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 252–266. Springer, Heidelberg (1992)
20. Lenstra, H.W.: Factoring integers with elliptic curves. Annals of Mathematics, 649–673 (1987)
21. Lenstra, H.W.: Elliptic curves and number theoretic algorithms. In: Proceedings of the International Congress of Mathematicians, pp. 99–120 (1988)
22. Menezes, A., Okamoto, T., Vanstone, S.A.: Reducing elliptic curve logarithms to logarithms in a finite field. IEEE Trans. Inf. Theory 39(5), 1639–1646 (1993)
23. Okamoto, T., Uchiyama, S.: Security of an Identity-Based Cryptosystem and the Related Reductions. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 546–560. Springer, Heidelberg (1998)
24. 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)
25. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
26. Silverman, J.H.: The Arithmetic of Elliptic Curves. GTM, vol. 106. Springer (1986)
27. Youn, T.-Y., Park, Y.-H., Kim, C.-H., Lim, J.: An Efficient Public Key Cryptosystem with a Privacy Enhanced Double Decryption Mechanism. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 144–158. Springer, Heidelberg (2006)

Secure Hierarchical Identity-Based Identification without Random Oracles

Atsushi Fujioka[1], Taiichi Saito[2], and Keita Xagawa[1]

[1] NTT Secure Platform Laboratories,
3-9-11 Midori-cho, Musashino-shi, Tokyo 180-8585, Japan
{fujioka.atsushi,xagawa.keita}@lab.ntt.co.jp
[2] Tokyo Denki University,
5 Senju Asahi-cho, Adachi-ku, Tokyo 120-8551, Japan
taiichi@c.dendai.ac.jp

Abstract. This paper proposes a generic construction of hierarchical identity-based identification (HIBI) protocols secure against impersonation under active and concurrent attacks in the standard model. The proposed construction converts a digital signature scheme existentially unforgeable against chosen message attacks, where the scheme has a protocol for showing possession of signing key. Our construction is based on the so-called certificate-based construction of hierarchical identity-based cryptosystems, and utilizes a variant of the well-known OR-proof technique to ensure the security against impersonation under active and concurrent attacks.

We also present several concrete examples of our construction employing the Waters signature (EUROCRYPT 2005), and other signatures. As results, its concurrent security of each instantiation is proved under the computational Diffie-Hellman (CDH) assumption, the RSA assumption, or their variants in the standard model.

Chin, Heng, and Goi proposed an HIBI protocol passively and concurrently secure under the CDH and one-more CDH assumption, respectively (FGIT-SecTech 2009). However, its security is proved in the random oracle model.

Keywords: hierarchical identity-based identification, impersonation under active and concurrent attacks, computational Diffie-Hellman assumption, RSA assumption.

1 Introduction

Identification is one of the important research topics in cryptography. An identification protocol is a protocol between a *prover* and a *verifier*, in which the prover tries to convince the verifier of his/her identity. In the so-called *standard identification* (SI), the verifier needs to know the identity and a public key of the prover. The security of SI protocols is defined by an experiment in which an adversary is allowed to act as verifiers in the *learning phase*, and then tries to impersonate some entity in the *challenge phase*. We say that the SI protocol is secure when, for any polynomial-time adversary, the probability that it succeeds in the impersonation is negligible.

In the experiment, when an adversary is only allowed to eavesdrop communications in identification in the learning phase, the attack model is called *impersonation under passive attacks* ($\mathsf{imp\text{-}pa}$ model). When an adversary is allowed to sequentially access entities who prove their identities (namely, only one access is allowed at a same time), the attack model is called *impersonation under active attacks* ($\mathsf{imp\text{-}aa}$ model), and when an adversary is allowed to concurrently access entities who prove their identities, even in the challenge phase, the attack model is called *impersonation under concurrent attacks* ($\mathsf{imp\text{-}ca}$ model) [2]. When we use an identification protocol in real world, an adversary may behave actively, and possibly establishes many connections. Thus, it is desirable that the protocol ensures at least the $\mathsf{imp\text{-}aa}$ security, and is expected to be $\mathsf{imp\text{-}ca}$ secure.

After the proposal of identity-based cryptography [18], identification in the identity-based setting, called *identity-based identification* (IBI), has been investigated. In an IBI protocol, a trusted third party called *private key generator* (PKG) is assumed to exist as in other identity-based cryptographic primitives. It publishes a master public key, generates a secret key corresponding to the identity of an entity, and gives the secret key to the entity. When an identification protocol starts, the verifier in the IBI protocol needs to know only the master public key and the identity of the prover, while an SI protocol requires the verifier to check the authenticity of the public key of the prover in some way (such as *public key infrastructure*).

Hierarchical identity-based identification (HIBI) is an extension of IBI. In an HIBI protocol, the single PKG functionality of generating secret keys is divided into partial ones and they are delegated to multiple PKGs. If a PKG is assigned an identity vector, $\mathsf{ID}^{(k-1)} = (I_1, \ldots, I_{k-1})$, and given a secret key, $sk_{\mathsf{ID}^{(k-1)}}$, corresponding to the identity vector, then it can generate a secret key, $sk_{\mathsf{ID}^{(k)}}$, corresponding to an identity vector, $\mathsf{ID}^{(k)} = (I_1, \ldots, I_{k-1}, I_k)$. We may denote an identity by ID if we need not to specify its hierarchy depth.

Similarly as IBI has three phase, HIBI also has Setup, Extract, and Identification: In phase Setup, the root PKG chooses a security parameter κ, generates a pair of master public and secret keys, and publishes the master public key. In phase Extract, the (intermediate) PKG of $\mathsf{ID}^{(k-1)}$ has the secret key, $sk_{\mathsf{ID}^{(k-1)}}$, and extracts the secret key, $sk_{\mathsf{ID}^{(k)}}$, for $\mathsf{ID}^{(k)}$, where $\mathsf{ID}^{(k-1)}$ is a prefix of $\mathsf{ID}^{(k)}$. In phase Identification, the prover of ID who has sk_{ID} proves the identity to a verifier.

The first security formulation for IBI was given by Kurosawa and Heng [14], and Bellare, Namprempre, and Neven provided formal security definitions of IBI protocols, the $\mathsf{imp\text{-}pa}$, $\mathsf{imp\text{-}aa}$, and $\mathsf{imp\text{-}ca}$ security [1]. Kurosawa and Heng also proposed the first generic construction of IBI protocols from digital signature (DS) schemes. The construction generates an $\mathsf{imp\text{-}pa}$ secure IBI protocol from a DS scheme *existentially unforgeable against chosen message attacks* ($\mathsf{euf\text{-}cma}$ secure) and its Σ-protocol [7] for signature possession.

The first security formulation for HIBI was given by Chin, Heng, and Goi [6]. They also proposed the first HIBI protocol based on the Kurosawa-Heng IBI protocol [14] and the *hierarchical identity-based signature* (HIBS) scheme by Gentry and Silverberg [10]. Their HIBI protocol is imp-pa secure under the CDH assumption and imp-ca secure under the one-more CDH assumption. However, since the Gentry-Silverberg scheme is based on the Boneh-Franklin signature [5], the security of their HIBI protocol is inherently proved in the random oracle model [3]. To the best of our knowledge, there exists no HIBI protocol proved to be secure in the standard model.

Our Contributions. This paper proposes a generic construction of HIBI protocols imp-ca secure in the standard model.

The generic construction converts an euf-cma secure DS scheme that has a Σ-protocol proving knowledge of *signature generation key.* We show that if the DS scheme has such protocol, the *certificate-based construction* [1] is applicable to the Σ-protocol in order to construct an HIBI protocol. Roughly speaking, in the HIBI protocol, each user has a certificate chain that certifies a signature verification key. In phase IDENTIFICATION, the prover sends the certificate chain and proves the knowledge of the signature generation key corresponding to the signature verification key certified by the certificate chain. Next, we modify the HIBI protocol by applying a variant of the well-known OR-proof technique, and prove that the resulting HIBI protocol is imp-ca secure.

We also present several concrete examples of our construction employing the Waters signature [19], and other signatures. Then, we obtain HIBI protocols proved to be imp-ca secure under the computational Diffie-Hellman (CDH) assumption, the RSA assumption, or their variants in the standard model.

Organization. We give a definition of hierarchical identity-based identification and related notions in **Section 2**. A generic construction of HIBI protocols and its security discussion are provided in **Section 3**, and **Section 4** gives HIBI protocols secure under several number-theoretic assumption in the standard model.

2 Definitions

In this section, we present definitions of hierarchical identity-based identification (HIBI) protocols and digital signature (DS) schemes, and introduce a property similar to Σ-protocols [7] in DS schemes.

Hierarchical Identity-Based Identification. Two types of formal definitions for key generation in hierarchical identity-based cryptography have been proposed in the literature. One consists of three algorithms, *Root Setup, Lower-level Setup*, and *Extraction*, as in the hierarchical identity-based encryption (HIBE) in [10], and the other consists of two algorithms, *root-key-generation algorithm* and *node-key-generation algorithm*, as in the HIBE in [13]. The two types of definitions are essentially the same. Here, we adopt the formal definition of HIBI protocols in [6]. Note that key generation in [6] is the former type, and we describe it in the latter type.

We define an HIBI protocol as follows: Let $\mathsf{HIBI} = (\mathsf{SetUp}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ be an HIBI protocol, where SetUp is the root-key-generation algorithm that on input 1^κ outputs mpk and msk, KG is the node-key-generation algorithm that on input $(sk_{\mathsf{ID}^{(k-1)}}, \mathsf{ID}^{(k)})$ outputs $sk_{\mathsf{ID}^{(k)}}$, P is the prover algorithm that, taking inputs mpk, ID and sk_{ID}, interacts with V, and V is the verifier algorithm that, taking inputs mpk and ID, interacts with P and finally outputs $dec \in \{accept, reject\}$. Here, $\mathsf{ID}^{(k-1)} = (I_1, \ldots, I_{k-1})$, $\mathsf{ID}^{(k)} = (I_1, \ldots, I_k)$, i.e., $\mathsf{ID}^{(k-1)}$ is a prefix of $\mathsf{ID}^{(k)}$, and $sk_{\mathsf{ID}^{(0)}} = msk$. Thus, SetUp is used in SETUP, KG is used in EXTRACT, P and V are used in IDENTIFICATION. Hereafter, for $\mathsf{ID} = (I_1, \ldots, I_k)$, we let $\mathsf{pref}(\mathsf{ID})$ denote the set of all prefixes of ID including itself, that is, $\{(I_1), (I_1, I_2), \ldots, (I_1, \ldots, I_k)\}$.

We describe the formal definitions of the security of HIBI based on the following $\mathsf{imp\text{-}atk}$ experiments between a challenger and an impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$, where atk denotes a type of attack such that $\mathsf{atk} \in \{\mathsf{pa}, \mathsf{aa}, \mathsf{ca}\}$.

Setup Phase: The challenger obtains $(mpk, msk) \leftarrow \mathsf{SetUp}(1^\kappa)$. The impersonator, CV, is given the master public key mpk.

Learning Phase: CV can ask queries to the oracles INIT, CORR, and CONV when $\mathsf{atk} = \mathsf{pa}$, and also to PROV when $\mathsf{atk} = \mathsf{aa}$ or ca. In each oracle access, the oracle is initialized by independent random coin, but maintains its own state during identification.

- The oracle INIT receives a user initialization query ID. It initializes the identities in $\mathsf{pref}(\mathsf{ID})$. It then returns ID to CV.
- The oracle CORR receives a key extraction query ID. It obtains $sk_{\mathsf{ID}} \leftarrow \mathsf{KG}(msk, \mathsf{ID})$, and returns sk_{ID} to CV.
- The oracle CONV receives a transcript query ID. It returns a transcript of a transaction between the prover with identity ID and a verifier to CV.
- (only when $\mathsf{atk} = \mathsf{aa}$ or ca) The oracle PROV receives an identification query ID. It obtains $sk_{\mathsf{ID}} \leftarrow \mathsf{KG}(msk, \mathsf{ID})$, runs $\mathsf{P}(mpk, \mathsf{ID}, sk_{\mathsf{ID}})$, and interacts with CV as the prover of identity ID. When $\mathsf{atk} = \mathsf{aa}$, CV is allowed to access PROV once at the same time, and when $\mathsf{atk} = \mathsf{ca}$, it is allowed to access PROV with many different identities, many times at the same time.

Challenge Phase: CV outputs a target identity ID^* and state information st_{CP}. If ID^* or any prefix of ID^* is queried to the key-extraction oracle CORR in the learning phase, then the challenger outputs 0 and halts. In addition, if ID^* is not initialized in the learning phase, then the challenger also outputs 0 and halts. Otherwise, the challenger gives st_{CP} to CP.
CP plays the role of a prover of the target identity ID^* and interacts with the challenger running $\mathsf{V}(mpk, \mathsf{ID}^*)$. During the interaction, CP is allowed to ask queries to the oracles INIT, CORR, and CONV, (and PROV when $\mathsf{atk} = \mathsf{aa}$ or ca), as the same as in the learning phase except that identification queries should not be the target identity ID^* and key extraction queries should not be ID^* or its prefixes. Finally the challenger outputs $dec \leftarrow \mathsf{V}(mpk, \mathsf{ID}^*)$.

We say an HIBI protocol is *secure against impersonation under passive*, *active* or *concurrent attacks* (named imp-pa, imp-aa or imp-ca secure, respectively) if for any polynomial-time passive, active or concurrent impersonator $\mathcal{I}$, $\Pr[\ dec = accept\ ]$ is negligible.

We note that the maximal depth ℓ is determined independently of the security parameter.

Digital Signature. Let $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a DS scheme, where Gen is the key-generation algorithm that on input 1^κ outputs a pair of signature verification and generation keys (vk, gk), Sign is the signing algorithm that takes as input a message m and gk and outputs a signature σ, and Vrfy is the verification algorithm that takes as input a message m, a signature σ and vk, and outputs *accept* or *reject*. For any $(vk, gk) \leftarrow \mathsf{Gen}(1^\kappa)$ and any m, $accept = \mathsf{Vrfy}(vk, m, \mathsf{Sign}(gk, m))$ holds.

Some DS schemes use not only public keys of users but also common parameters. In such scheme, Gen is divided into two algorithms: PGen and KGen. PGen on input 1^κ generates the common parameters *params*, and KGen takes *params* as input and generates a pair of signature verification and generation keys (vk, gk) for each user. We call this type of DS schemes *set-up type.*

Key Possession Protocol for Digital Signature. In this paper, we require that a DS scheme has a variant of Σ-protocols [7] proving knowledge of signature generation key.

Let $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a DS scheme. Let vk be a signature verification key and gk be a signature generation key such that $(vk, gk) \leftarrow \mathsf{Gen}(1^\kappa)$. The common input to (P, V) is vk, and the secret input to P is gk.

Suppose that a three-move protocol between P and V are described by four probabilistic polynomial time algorithms (Σ_{com}, Σ_{ch}, Σ_{res}, Σ_{vrfy}), as follows.

$\mathsf{P} \rightarrow \mathsf{V}$: P computes $(a, st) \leftarrow \Sigma_{\mathsf{com}}(vk, gk)$ and sends a to V.
$\mathsf{V} \rightarrow \mathsf{P}$: V computes $c \leftarrow \Sigma_{\mathsf{ch}}(vk)$ and sends c to P.
$\mathsf{P} \rightarrow \mathsf{V}$: P computes $z \leftarrow \Sigma_{\mathsf{res}}(vk, gk, a, c, st)$ and sends z to V.
V: V computes $dec \leftarrow \Sigma_{\mathsf{vrfy}}(vk, a, c, z)$ and outputs $dec \in \{accept, reject\}$.

We call this type of three-move protocols *canonical* [2], and moreover, we call a canonical protocol *Σ^+-type* if it satisfies the following three properties: *special zero-knowledge*, *special soundness*, and *special challenge* [9]:

Special zero-knowledge: We can obtain an accepting transcript from a challenge, c, and vk. That is, there is a probabilistic polynomial time algorithm Σ_{sim} that takes on input vk and c such that $c \leftarrow \Sigma_{\mathsf{ch}}(vk)$, and outputs (a, z) such that $accept = \Sigma_{\mathsf{vrfy}}(vk, a, c, z)$. The distribution of transcripts generated by Σ_{ch} and Σ_{sim} is indistinguishable from that of real transcripts.

Special soundness: We can compute a signature generation key gk' from the signature verification key vk and two accepting transcripts (a, c, z) and (a, c', z') such that $c \neq c'$. That is, there is a probabilistic polynomial time algorithm

Σ_{ext} that takes as input vk and two transcripts (a, c, z) and (a, c', z') satisfying $accept = \Sigma_{\mathsf{vrfy}}(vk, a, c, z) = \Sigma_{\mathsf{vrfy}}(vk, a', c', z')$ and $c \neq c'$, and outputs gk', where $accept = \mathsf{Vrfy}(vk, m, \mathsf{Sign}(gk', m))$ holds for any m.

Special challenge: Σ_{ch} depends only on *params*, not on vk, and the output c is uniformly distributed over a commutative group $\mathbb{G}$. In addition, the group operation $+$ is computable in polynomial time. $\mathbb{G}$ is determined only by *params*, not by vk. That it, there is a probabilistic polynomial time algorithm Σ^{+}_{ch} such that it takes as input *params* (not vk) and outputs c, and c is uniformly distributed over $\mathbb{G}$.

We call an protocol Σ^**-type* if the special challenge property is replaced with the following property [9]:

Strongly special challenge: Σ_{ch} depends only on 1^κ, not on vk, and the output challenge c is uniformly distributed over a commutative group $\mathbb{G}$. In addition, the group operation $+$ is computable in polynomial time. $\mathbb{G}$ is determined only by 1^κ, not by vk. That it, there is a probabilistic polynomial time algorithm Σ^*_{ch} such that it takes as input 1^κ (not vk) and outputs c, and c is uniformly distributed over $\mathbb{G}$.

Note that while the special challenge property requires that the underlying DS scheme is set-up type, the strongly special challenge property does not.

3 Construction of Strongly Secure HIBI Protocols

Let $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a DS scheme which has a Σ^*-type protocol for key possession. Suppose that the protocol for key possession uses four probabilistic polynomial time algorithms (Σ_{com}, Σ^*_{ch}, Σ_{res}, Σ_{vrfy}), and has the special zero-knowledge property with a probabilistic polynomial time algorithm Σ_{sim} and the special soundness with a probabilistic polynomial time algorithm Σ_{ext}.

Before showing the proposed construction, we provide a construction of imp-pa secure HIBI protocols from DS schemes. Then, we modify the construction to the proposed one which generates imp-ca secure HIBI protocols by using the well-known OR-proof technique.

3.1 Weakly Secure Construction

In this section, we give a construction of HIBI from DS by describing an HIBI protocol, HIBI', produced from a DS scheme, DS. When DS is euf-cma secure, then HIBI', is weakly secure, i.e., imp-pa secure.

In phase Setup of HIBI', the root PKG chooses a security parameter κ, generates a pair of signature verification and generation keys, (vk, gk), with Gen, and publishes vk as the master public key.

In phase Extract of HIBI', the (intermediate) PKG of $\mathsf{ID}^{(k-1)}$ has the secret key, $sk_{\mathsf{ID}^{(k-1)}}$, and extracts the secret key, $sk_{\mathsf{ID}^{(k)}}$ for $\mathsf{ID}^{(k)}$, where $\mathsf{ID}^{(k-1)}$ is a prefix of $\mathsf{ID}^{(k)}$, i.e., $\mathsf{ID}^{(k-1)} = (I_1, \ldots, I_{k-1})$ and $\mathsf{ID}^{(k)} = (I_1, \ldots, I_{k-1}, I_k)$. $sk_{\mathsf{ID}^{(k-1)}}$ consists of a chain of signature verification keys $(vk_1, \ldots, vk_{k-1})$, their signatures $(\sigma_1, \ldots, \sigma_{k-1})$, and the signature generation key gk_{k-1}, where $vk_0 = mpk$, $gk_0 = msk$, $\sigma_i \leftarrow \mathsf{Sign}(gk_{i-1}, (\mathsf{ID}^{(i)}, vk_i))$, and $((\mathsf{ID}^{(i)}, vk_i), \sigma_i)$ is a valid certificate under vk_{i-1} $(1 \le i \le k)$. Then, the (intermediate) PKG generates a pair of signature verification and generation keys, (vk_k, gk_k), with Gen, signs $(\mathsf{ID}^{(k)}, vk_k)$ with Sign to compute its signature, σ_k, and forms $sk_{\mathsf{ID}^{(k)}}$ as a chain of signature verification keys $vk_1, \ldots, vk_k$, their signatures $\sigma_1, \ldots, \sigma_k$, and the signature generation key gk_k.

In phase Identification of HIBI', the prover of $\mathsf{ID}^{(n)}$ who has $sk_{\mathsf{ID}^{(n)}}$ proves the identity to a verifier by showing possession of gk_n corresponding to vk_n. It generates a commitment value a with Σ_{com}, and sends $(vk_1, \ldots, vk_n)$, $(\sigma_1, \ldots, \sigma_n)$, and a to the verifier. The verifier checks validity of all certificates, generates a challenge value c with Σ^*_{ch}, and sends c to the prover. The prover generates a return value z with Σ_{res}, and sends z to the verifier. Finally, the verifier checks validity of (a, c, z) with Σ_{vrfy}, and outputs the result.

Next, we show the above construction generates weakly secure IBI protocols.

Theorem 3.1. *If the underlying DS scheme is* euf-cma *secure, then the constructed HIBI protocol is* imp-pa *secure.*

We describe an outline of the proof. In order to prove the theorem by contradiction, let us consider an impersonator $\mathcal{I}' = (\mathsf{CV}', \mathsf{CP}')$ who breaks the imp-pa security of HIBI'.

In the security game, the challenger generates a chain of signature verification keys on the query ID to the Init oracle. When $\mathcal{I}'$ impersonates the target identity, $\mathcal{I}'$ uses either a new chain or the chain of signature verification keys which the challenger prepared on the query from $\mathcal{I}'$. Thus, we first classify the impersonator into two types; A Type-I impersonator sends the signature verification keys in the first message that are the same as them prepared by the challenger, and a Type-II impersonator does not.

We then construct a forger, $\mathcal{F}'$, who breaks the euf-cma security of DS from the impersonator $\mathcal{I}'$.

Type-I: Receiving vk, $\mathcal{F}'$ guesses when $\mathcal{I}'$ initializes an identity of which the target identity is a prefix, and guesses the depth of the target identity. $\mathcal{F}'$ generates the keys in the HIBI protocol and assigns vk the leaf key of the target identity, i.e., the verification key of the guessed depth. Since $\mathcal{F}'$ has the signing oracle and all signature generation keys but the one corresponding to vk, and the protocol is Σ^*-type, $\mathcal{F}'$ perfectly simulates the oracles until $\mathcal{I}'$ corrupts the identity that $\mathcal{F}'$ guessed as the target identity. Eventually, CV' declares the target identity ID^*. If $\mathcal{F}'$'s guess is wrong, then $\mathcal{F}'$ aborts. Otherwise, it runs CP' twice with different challenges and extracts the corresponding signature generation key gk to vk. So that, $\mathcal{F}'$ can forge any messages.

Type-II: From the hypothesis that $\mathcal{I}'$ does not send the signature verification keys in the first message that are the same as them prepared by the challenger, $\mathcal{I}'$ has forged at least one of certificates $(\sigma_1, \ldots, \sigma_n)$. As in the case of the Type-I impersonator, $\mathcal{F}'$ guesses when $\mathcal{I}'$ initializes an identity of which the target identity is a prefix, and guesses the depth of the signature verification key in the chain on which $\mathcal{I}'$ will forge a certificate. $\mathcal{F}'$ generates the keys in the HIBI protocol and assigns vk the key of the target identity in the chosen depth. Since $\mathcal{F}'$ has the signing oracle and all signature generation keys but corresponding one to vk, and the protocol is Σ^*-type, $\mathcal{F}'$ perfectly simulates the oracles. Eventually, CV' declares the target identity ID^*. If $\mathcal{F}'$'s guess is wrong, then $\mathcal{F}'$ aborts. Otherwise, it runs CP' and obtains a forgery.

A full proof will be given in the final version of this paper.

3.2 Proposed Construction

In this section, we give the proposed construction of HIBI from DS.

We describe an HIBI protocol $\mathsf{HIBI} = (\mathsf{SetUp}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ produced from a DS scheme, DS.

$\mathsf{SetUp}(1^\kappa)$**:** It takes as input 1^κ, runs $(vk, gk) \leftarrow \mathsf{Gen}(1^\kappa)$, $(vk_{master}, gk_{master}) \leftarrow \mathsf{Gen}(1^\kappa)$, and outputs $((1^\kappa, vk, vk_{master}), (1^\kappa, gk))$ as (mpk, msk).

$\mathsf{KG}(sk_{\mathsf{ID}^{(k-1)}}, \mathsf{ID}^{(k)})$**:** It takes as input $sk_{\mathsf{ID}^{(k-1)}}$ and $\mathsf{ID}^{(k)}$. Then, it parses $sk_{\mathsf{ID}^{(k-1)}}$ as $sk_{\mathsf{ID}^{(k-1)}} = (1^\kappa, vk_1, \ldots, vk_{k-1}, \sigma_1, \ldots, \sigma_{k-1}, gk_{k-1})$, runs $(vk_k, gk_k) \leftarrow \mathsf{Gen}(1^\kappa)$, $\sigma_k \leftarrow \mathsf{Sign}(gk_{k-1}, (\mathsf{ID}^{(k)}, vk_k))$, and outputs $(1^\kappa, vk_1, \ldots, vk_k, \sigma_1, \ldots, \sigma_k, gk_k)$ as $sk_{\mathsf{ID}^{(k)}}$. Here, $sk_{\mathsf{ID}^{(0)}} = (1^\kappa, gk)$, $gk_0 = gk$, and $\mathsf{ID}^{(k-1)}$ of $sk_{\mathsf{ID}^{(k-1)}}$ is a prefix of $\mathsf{ID}^{(k)}$, i.e., $\mathsf{ID}^{(k-1)} = (I_1, \ldots, I_{k-1})$ and $\mathsf{ID}^{(k)} = (I_1, \ldots, I_{k-1}, I_k)$.

$\mathsf{P} \to \mathsf{V}$**:** P takes as input $(mpk, \mathsf{ID}^{(n)}, sk_{\mathsf{ID}^{(n)}})$, parses mpk and $sk_{\mathsf{ID}^{(n)}}$ as $(1^\kappa, vk, vk_{master})$ and $(1^\kappa, vk_1, \ldots, vk_n, \sigma_1, \ldots, \sigma_n, gk_n)$, respectively, runs $(a_0, st) \leftarrow \Sigma_{\mathsf{com}}(vk_n, gk_n)$, $c_1 \leftarrow \Sigma^*_{\mathsf{ch}}(1^\kappa)$, and $(a_1, z_1) \leftarrow \Sigma_{\mathsf{sim}}(vk_{master}, c_1)$, and sends $(vk_1, \ldots, vk_n)$, $(\sigma_1, \ldots, \sigma_n)$ and (a_0, a_1) to V.

$\mathsf{V} \to \mathsf{P}$**:** V takes as input $(mpk, \mathsf{ID}^{(n)})$, parses mpk as $(1^\kappa, vk_0, vk_{master})$, runs $dec_i \leftarrow \mathsf{Vrfy}(vk_{i-1}, (\mathsf{ID}^{(i)}, vk_i), \sigma_i)$ $(1 \le i \le n)$, and outputs *reject* if $dec_i =$ *reject* for some i. Otherwise, V runs $c \leftarrow \Sigma^*_{\mathsf{ch}}(1^\kappa)$, and sends c to P.

$\mathsf{P} \to \mathsf{V}$**:** P computes $c_0 = c - c_1$, runs $z_0 \leftarrow \Sigma_{\mathsf{res}}(vk_n, gk_n, a_0, c_0, st)$, and sends (c_0, z_0, z_1) to V.

V**:** V computes $c_1 = c - c_0$, runs $dec'_1 \leftarrow \Sigma_{\mathsf{vrfy}}(vk_{master}, a_1, c_1, z_1)$, $dec'_0 \leftarrow \Sigma_{\mathsf{vrfy}}(vk_n, a_0, c_0, z_0)$, and outputs *accept* if both dec'_0 and dec'_1 are *accept*s. Otherwise, it outputs *reject*.

Note that the above construction requires the strongly special challenge property, i.e., challenges are generated by Σ^*_{ch}. However, when we starts from a set-up type DS scheme, it is clear that we need only the special challenge property since it is enough to generate the challenges from *params* not 1^κ.

3.3 Security against Active and Concurrent Attacks

In this section, we prove that the constructed HIBI protocol is strongly secure when the DS scheme is $\mathsf{euf\text{-}cma}$ secure. Actually, we show that the HIBI protocols is $\mathsf{imp\text{-}ca}$ secure, and it is clear that the $\mathsf{imp\text{-}ca}$ security implies the $\mathsf{imp\text{-}aa}$ security. Thus, the protocol is strongly secure, i.e., secure against active and concurrent attacks.

Setup

$\mathsf{SetUp}(1^\kappa)$

$(vk, gk) \leftarrow \mathsf{Gen}(1^\kappa)$
$(vk_{master}, gk_{master}) \leftarrow \mathsf{Gen}(1^\kappa)$
output $(mpk, msk) = ((1^\kappa, vk, vk_{master}), (1^\kappa, gk))$

Extract

$\mathsf{KG}(sk_{\mathsf{ID}^{(k-1)}}, \mathsf{ID}^{(k)})$

$sk_{\mathsf{ID}^{(k-1)}} = (1^\kappa, vk_1, \ldots, vk_{k-1}, \sigma_1, \ldots, \sigma_{k-1}, gk_{k-1})$
$(vk_k, gk_k) \leftarrow \mathsf{Gen}(1^\kappa)$
$\sigma_k \leftarrow \mathsf{Sign}(gk_{k-1}, (\mathsf{ID}^{(k)}, vk_k))$
output $sk_{\mathsf{ID}^{(k)}} = (1^\kappa, vk_1, \ldots, vk_k, \sigma_1, \ldots, \sigma_k, gk_k)$

Identification

$\mathsf{P}(mpk, \mathsf{ID}^{(n)}, sk_{\mathsf{ID}^{(n)}})$		$\mathsf{V}(mpk, \mathsf{ID}^{(n)})$
$mpk = (1^\kappa, vk, vk_{master})$		$mpk = (1^\kappa, vk, vk_{master})$
$sk_{\mathsf{ID}^{(n)}} = (1^\kappa, vk_1, \ldots, vk_n, \sigma_1, \ldots, \sigma_n, gk_n)$		
$(a_0, st) \leftarrow \Sigma_{\mathsf{com}}(vk_n, gk_n)$	$((vk_1, \ldots, vk_n),$	
$c_1 \leftarrow \Sigma^*_{\mathsf{ch}}(1^\kappa)$	$(\sigma_1, \ldots, \sigma_n),$	
$(a_1, z_1) \leftarrow \Sigma_{\mathsf{sim}}(vk_{master}, c_1)$	$(a_0, a_1))$ $\longrightarrow$	$dec_i \leftarrow \mathsf{Vrfy}(vk_{i-1}, (\mathsf{ID}^{(i)}, vk_i), \sigma_i)$ $(1 \le i \le n)$
		output *reject* if $dec_i = reject$ for some i
$c_0 = c - c_1$	c $\longleftarrow$	$c \leftarrow \Sigma^*_{\mathsf{ch}}(1^\kappa)$
$z_0 \leftarrow \Sigma_{\mathsf{res}}(vk_n, sk_n, a_0, c_0, st)$	(c_0, z_0, z_1) $\longrightarrow$	$c_1 = c - c_0$
		$dec'_0 \leftarrow \Sigma_{\mathsf{vrfy}}(vk_n, a_0, c_0, z_0)$
		$dec'_1 \leftarrow \Sigma_{\mathsf{vrfy}}(vk_{master}, a_1, c_1, z_1)$
		output *accept* if $dec'_0 = dec'_1 = accept$, and otherwise, output *reject*

Fig. 1. Proposed Construction

Theorem 3.2. *If the underlying DS scheme is* euf-cma *secure, then the constructed HIBI protocol is* imp-ca *secure.*

We describe an outline of the proof.

In order to prove the theorem by contradiction, we construct a forger, $\mathcal{F}$, who breaks the euf-cma security of DS from an impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$ who breaks the imp-ca security of HIBI.

To impersonate an identity, $\mathcal{I}$ needs to show possession of an signature generations key for either one of signature verification keys of a new certificate chain, the last of the certificate chain which $\mathcal{F}$ prepared, or vk_{master}.

The first two types are the same as Type-I and Type-II in the passive security. We classify the third type, $\mathcal{I}$ exploiting vk_{master}, as Type-III.

In the active and concurrent security, $\mathcal{F}$ needs to simulate Prov oracle for all types of impersonators. It is possible for the Type-I and Type-II impersonators when $\mathcal{F}$ has the generation key, gk_{master}, for vk_{master}, and possible for the Type-III impersonator when $\mathcal{F}$ has the all generation keys, gks, for all users.

Note that for the imp-ca security, the key possession protocol must satisfy the (strongly) special challenge property as the challenge should be split and is used for Σ-protocols to show possession of both signature generation keys gk and gk_{master}.

A full proof will be given in the final version of this paper.

3.4 Discussion

It is known that an HIBS scheme can be constructed from a DS scheme by using chains of certificates, which is called *certificate-based* HIBS in [1].

If we have a DS scheme with a Σ-protocol for signature possession, we obtain an HIBI-like protocol by combining the certificate-based HIBS construction and the Kurosawa-Heng IBI construction [14]. However the HIBI-like protocol is different from HIBI protocols in some points. The HIBI-like protocol needs two different key-generation algorithms; one generates secret keys for PKGs (internal nodes in hierarchical tree), and another secret keys for users (leaf nodes). Since the secret key for user is a signature of the underlying DS scheme and not a signature generation key, it cannot be used for generating secret keys for descendant nodes.

It is also known that an HIBI protocol can be constructed from an SI protocol and a DS scheme by generating a certificate of a public key of the SI protocol with the certificate-based HIBS scheme [1]. Our generic construction of HIBI protocols can be seen as a variant of this construction. If the SI protocol is a Σ-protocol for proving knowledge of signature generation key in the DS scheme, this protocol coincides with our generic construction of HIBI protocols from DS schemes. However our construction does not need an additional cryptographic primitive such as the SI protocol. In addition, the prover does not send a chain of certificates but a chain of the signature verification keys and that of signatures, since the verifier has the prover's identity vector. This reduces communication cost of the identities stored in the certificates.

4 Instantiations

4.1 Secure HIBI Protocol from Prime-Order Bilinear Group

Through our generic construction, we can instantiate an HIBI protocol from the Waters signature scheme $\mathsf{Wat05}$ [19], which is secure under the CDH assumption in a prime-order bilinear group, with a Σ-protocol described later. Unfortunately, the signature verification key in $\mathsf{Wat05}$ consists of $n+3$ group elements, where n is the length of a message. Hence, we modify $\mathsf{Wat05}$ to a set-up type one such that its signature verification key consists of a group element, whereas the public parameter contains $n+2$ group elements. We call it $\mathsf{Wat05}'$, and see that it is also $\mathsf{euf\text{-}cma}$ secure under the CDH assumption.

[A variant $\mathsf{Wat05}'$ of $\mathsf{Wat05}$] We start to describe our variant. Let $\mathbb{G}$ be a group of prime order, p, for which there exists an efficiently computable bilinear map into $\mathbb{G}_T$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ denote the bilinear map and let g be a generator of $\mathbb{G}$. The size of the group, p, is determined by the security parameter κ. We will sign messages of n bits, and we can use a collision-resistant hash function, $H : \{0,1\}^* \to \{0,1\}^n$, to sign messages of arbitrary length.

Let b be an n-bit binary string $b = (b_1, \ldots, b_n)$ $(b_i \in \{0,1\})$, and $\langle u \rangle$ be $\langle u \rangle = (u_0, u_1, \ldots, u_n)$ $(u_i \in \mathbb{G})$. Then, U_b denotes $U_b = u_0 \cdot \prod_{i=1}^{n} u_i^{b_i}$.

The variant $\mathsf{Wat05}'$ is defined as follows:

PGen: It takes as input 1^κ and generates a bilinear group $(\mathbb{G}, \mathbb{G}_T, e, p, g)$. It randomly generates $u_0, u_1, \ldots, u_n \leftarrow \mathbb{G}$ and sets $\langle u \rangle = (u_0, u_1, \ldots, u_n)$. It outputs $params = (\mathbb{G}, \mathbb{G}_T, e, p, g, \langle u \rangle)$.

KGen: It takes as input 1^κ and $params$ and randomly selects a secret $x \in \mathbb{Z}_p$. The signature verification key is $vk = e(g, g)^x$. The signature generation key is $gk = g^x$.

Sign: It takes as input $params$, gk, and $m \in \{0,1\}^n$, randomly selects $r \leftarrow \mathbb{Z}_p$, computes

$$(\sigma_1, \sigma_2) = (gk \cdot (U_m)^r, g^r),$$

and outputs $\sigma = (\sigma_1, \sigma_2)$.

Vrfy: It takes as input $params$, vk, m, and σ, parses σ as $\sigma = (\sigma_1, \sigma_2)$, checks whether

$$vk = e(\sigma_1, g)/e(\sigma_2, U_m)$$

holds, and outputs *accept* if it holds. Otherwise, it outputs *reject*.

[Σ^+-type Protocol for $\mathsf{Wat05'}$]. To construct an HIBI protocol, we require a Σ-protocol for the key relation $\{(vk, gk) \in \mathbb{G}_T \times \mathbb{G} : vk = e(gk, g)\}$. The following Σ-protocol is obtained by applying the Fiat-Shamir transform in reverse order to the identity-based signature scheme 3 in Hess [11]:

Σ_{com}: It takes as input $params$, vk and gk, and parses $params = (\mathbb{G}, \mathbb{G}_T, e, p, g, \langle u \rangle)$, $vk = e(g, g)^x$, and $gk = g^x$. It randomly selects $r \in \mathbb{Z}_p$, computes $A = e(g, g)^r$, and outputs $(a, st) = (A, r)$.

Σ^+_{ch}: It takes as input $params = (\mathbb{G}, \mathbb{G}_T, e, p, g, \langle u \rangle)$. It randomly selects $c \in \mathbb{Z}_p$ and outputs c.

Σ_{res}: It takes as input $params$, vk, gk, a, c, and st, and parses $st = r$. It then computes $Z = g^r \cdot gk^c$ and outputs $z = Z$.

Σ_{vrfy}: It takes as input $params$, vk, a, c, and z, and parses them as $params = (\mathbb{G}, \mathbb{G}_T, e, p, g, \langle u \rangle)$, $a = A$, $z = Z$. It checks whether $A \in \mathbb{G}_T$, $Z \in \mathbb{G}$, and $e(Z, g) = A \cdot vk^c$ hold, and outputs *accept* if they hold; otherwise, it outputs *reject*.

Σ_{sim}: It takes as input $params$, vk, and c. It randomly selects $Z \in \mathbb{G}$ and computes $A = e(Z, g) \cdot vk^{-c}$. It outputs (A, Z) as (a, z).

Σ_{ext}: It takes as input $params$, vk, (a, c, z), and (a, c', z'), and parses them as $a = A$, $z = Z$, and $z' = Z'$. It computes $t = (Z/Z')^{1/(c-c')} \in \mathbb{G}$ and outputs t as gk.

Theorem 4.1. *The above protocol is Σ^+-type.*

Proof. We only show the special soundness since it is easy to verify special zero-knowledge and special challenge properties. Suppose that (A, c, Z) and (A, c', Z') are two accepting transcripts such that $c \neq c'$. The algorithm Σ_{ext} computes $t = (Z/Z')^{1/(c-c')}$ as gk. From the hypothesis, it holds that

$$e(Z, g) = A \cdot vk^c \text{ and } e(Z', g) = A \cdot vk^{c'}.$$

Hence, we have that

$$e(t,g) = \left(\frac{e(Z,g)}{e(Z',g)}\right)^{1/(c-c')} = \left(\frac{A \cdot vk^c}{A \cdot vk^{c'}}\right)^{1/(c-c')} = \left(vk^{c-c'}\right)^{1/(c-c')} = vk.$$

This completes the proof. □

4.2 Secure HIBI Protocol from Composite-Order Bilinear Group

We next instantiate an HIBI protocol from $\mathsf{LW10}'$, which is a variant of the Lewko-Waters signature scheme, $\mathsf{LW10}$, [15], through our generic construction.

Roughly speaking, their signature scheme is the Boneh-Boyen identity-based encryption scheme [4] with a tweak of composite order pairing. Their scheme employs a pairing group $\mathbb{G}$ of composite order $N = p_1p_2p_3$, which has subgroups $\mathbb{G}_{p_i}$ of prime order p_i with generator g_i (for $i \in \{1, 2, 3\}$).

In the original scheme, the signature verification key is $(g_1, u, h, e(g_1, g_1)^x) \in \mathbb{G}_{p_1}^3 \times \mathbb{G}_T$ and the signature generation key is $(g_1^x, g_3) \in \mathbb{G}_{p_1} \times \mathbb{G}_{p_3}$. We observe that appearance of a generator of $\mathbb{G}_3$, g_3, causes nothing in the security proof of the signature. In addition, we observe that users can share the composite-order group if they trust the parameter generation.

Let $N = p_1p_2p_3$ be the product of three distinct primes. Let $\mathbb{G}$ be a group of order N, for which there exists an efficiently computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. For $i = 1, 2, 3$, let $\mathbb{G}_{p_i}$ denote a subgroup of $\mathbb{G}$ of order p_i. We will sign messages in $\mathbb{Z}_N$, and we can use a collision-resistant hash function, $H : \{0, 1\}^* \to \mathbb{Z}_N$, to sign messages of arbitrary length.

The scheme $\mathsf{LW10}'$ is defined as follows:

PGen: It takes as input 1^κ, randomly selects three primes p_1, p_2, p_3, and constructs $\mathbb{G}$ of order $N = p_1p_2p_3$ and $\mathbb{G}_T, e$, and generators g and R of $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_3}$, respectively. It randomly selects $u, h \in \mathbb{G}_{p_1}$. The public parameter is $params = (\mathbb{G}, \mathbb{G}_T, e, N, g, u, h, R)$.

KGen: It takes as input $params$. It randomly selects a secret $x \in \mathbb{Z}_N$. The signature verification key is $vk = e(g, g)^x$. The signature generation key is $gk = g^x$.

Sign: It takes as input $params$, gk and $m \in \mathbb{Z}_N$, randomly selects $r, \gamma, \gamma' \in \mathbb{Z}_N$, computes

$$(\sigma_1, \sigma_2) = (gk \cdot (u^m \cdot h)^r \cdot R^\gamma, g^r \cdot R^{\gamma'})$$

and outputs $\sigma = (\sigma_1, \sigma_2)$.

Vrfy: It takes as input $params$, vk, m, and $\sigma = (\sigma_1, \sigma_2)$, randomly selects $s \leftarrow \mathbb{Z}_N$, checks whether

$$vk^s = e(\sigma_1, g^s)/e(\sigma_2, (u^m \cdot h)^s)$$

holds, and outputs *accept* if it holds. Otherwise, it outputs *reject*.

[Σ^+-type Protocol for $\mathsf{LW10'}$]. Again, the following protocol is obtained by applying the Fiat-Shamir transform in reverse order to the identity-based signature scheme 3 in Hess [11].

Σ_{com}: It takes as input *params*, *vk* and *gk*. It randomly selects $r \in \mathbb{Z}_N$ and outputs $A = e(g,g)^r$ as a and r as st.
Σ^+_{ch}: It takes as input *params*. It randomly selects $c \in \mathbb{Z}_N$ and outputs c.
Σ_{res}: It takes as input *params*, *vk*, *gk*, a, c, and st, and parses $a = A$, $st = r$. It then computes $Z = g^r \cdot gk^c$, and outputs Z as z.
Σ_{vrfy}: It takes as input *params*, *vk*, a, c, and z, and parses them as $a = A$, $z = Z$. It checks whether $A \in \mathbb{G}_T$, $Z \in \mathbb{G}$, and $e(Z,g) = A \cdot vk^c$ hold, and outputs *accept* if they hold, and otherwise *reject*.
Σ_{sim}: It takes as input *params*, *vk*, and c. It randomly selects $\gamma \in \mathbb{Z}_N$, and compute $Z = g^\gamma$ and $A = e(Z,g) \cdot vk^{-c}$. It outputs (A, Z) as (a, z).
Σ_{ext}: It takes as input *params*, *vk*, (a,c,z), and (a,c',z'), and parses them as $a = A$, $z = Z$, and $z' = Z'$. It computes $t = (Z/Z')^{1/(c-c')} \in \mathbb{G}$ and outputs t as gk.

Remark 4.1. We note that the protocol is for the relation $\{(vk, gk) \in \mathbb{G}_T \times \mathbb{G} : e(gk,g) = vk\}$, where gk is in $\mathbb{G}$ rather than $\mathbb{G}_{p_1}$. Hence, gk' extracted from Σ_{ext} may contain the part of $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ but this (invalid) gk' has ability to forge any message. We can generate a signature from gk' as $(\sigma'_1, \sigma'_2) = (gk' \cdot (u^m h)^r R^\gamma, g^r R^{\gamma'})$ which passes the verification.

Theorem 4.2. *The above protocol is Σ^+-type.*

Proof. We only show the special soundness since it holds special zero-knowledge and special challenge properties obviously. As in the previous proof, suppose that (A,c,Z) and (A,c',Z') are two accepting transcripts such that $c \neq c'$. Then, the algorithm Σ_{ext} computes $t = (Z/Z')^{1/(c-c')}$ as gk. By the argument in the previous proof, we have that $e(t,g) = vk$. □

4.3 Secure HIBI Protocol under the (Strong) RSA Assumption

To instantiate the secure HIBI protocol under the (strong) RSA assumption, we employ the strong-RSA-based signature schemes $\mathsf{CS99}$ proposed by Cramer and Shoup [8] or the RSA-based signature scheme $\mathsf{HW09}$ proposed by Hohenberger and Waters [12] as examples.

In both schemes, the signature verification key involves the RSA modulus $N = pq$, which is the product of two safe primes. The signature generation key is the factorization p and q.

[Σ^+-type Protocol for Factorization]. The user can prove the knowledge of such factorization by the zero-knowledge proof by Poupard and Stern [17]. We give an optimized version of the protocol.

Let N be the product of two strong primes $p = 2p' + 1$ and $q = 2q' + 1$. Let λ be the bit length of N. The protocol involves two thresholds T_a and T_c, and a randomly chosen element $G \in [\,0, 2^{\lambda+\omega(\lg \kappa)})$.

Σ_{com}**:** It takes as input $params = (T_a, T_c, G)$, vk including $N = pq$ and $gk = (p, q)$. It randomly selects $r \in [0, T_a)$ and outputs $A = G^r \bmod N$ as a and r as st.

Σ^{+}_{ch}**:** It takes as input $params$. It randomly selects $c \in [0, T_c)$ and outputs c.

Σ_{res}**:** It takes as input $params$, vk, gk, a, c, and st, and parses $a = A$, $st = r$. It then computes $z = r + c \cdot (N - \phi(N))$, where $\phi(\cdot)$ is Euler's totient function, and outputs z.

Σ_{vrfy}**:** It takes as input $params$, vk, a, c, and z, and parses them as $params = (T_a, T_c, G)$ and $a = A$. It checks whether $z \in [0, T_a)$ and $A \equiv G^{z-Nc} \pmod N$ hold, and outputs *accept* if they hold, and otherwise *reject*.

Σ_{sim}**:** It takes as input $params$, $vk = N$, and $c \in [0, T_c)$. It randomly selects $z \in [0, T_a)$ and computes $A = G^{z-Nc} \bmod N$. It outputs (A, z) as (a, z).

Σ_{ext}**:** It takes as input $params$, vk, (A, c, z), and (A, c', z'). It computes $t = N(c - c') - (z - z')$ which may be a non-zero multiple of $p'q'$. If so, it obtains factoring of N by using Miller's factoring algorithm [16,17].

Remark 4.2. In the original protocol, $params = (T_a, T_c, G_1, G_2, \ldots, G_K)$ and it runs the above protocol in parallel with sharing state r and challenge c. Setting $K = 1$ suffices for our purpose as in the following.

Theorem 4.3 (Adapted version of [17, Theorems 4, 5 and 7]). *Let κ be the security parameter. Suppose that the common input is the RSA modulus $N = pq$ of length λ, where p and q are strong primes. If $\lg T_c = \omega(\lg \kappa)$ and $\lg T_a < \lambda$, then the above protocol has special soundness. If $(N - \phi(N))T_c/T_a$ is negligible in κ, then the above protocol is complete with negligible error and has special zero-knowledge property.*

Proof. Special challenge property and special zero-knowledge property follows from the original proof. Hence, we omit them.

We discuss the special soundness property. Let (A, c, z) and (A, c', z') be two acceptance transcripts. Since they are accepted, we have that

$$A \equiv G^{z-Nc} \equiv G^{z'-Nc'} \pmod N.$$

Hence, $t = N(c-c') - (z-z')$ is the multiple of the unknown order of G. We note that t is non-zero, since $c \neq c'$, $T_a < N$, and $z - z' \in (-T_a, T_a)$. We additionally note that the order of G in $\mathbb{Z}_N^*$ is in $\{1, 2, 4, p', q', 2'p, 2q', p'q', 2p'q' = \lambda(N)\}$. For randomly chosen G, the probability that the order of G is less than $p'q'$ is negligible in κ. Therefore, the algorithm Σ_{ext} can factorize N with the Miller factoring algorithm by using $2t$, which is the non-zero multiple of $\lambda(N)$ with overwhelming probability.

Remark 4.3. For example, we can set $T_a = 2^{\lambda/2+2\kappa}$ and $T_c = 2^{\kappa}$.

4.4 Comparison

In this section, we compare the $\mathsf{imp\text{-}ca}$-secure HIBI protocols produced by our generic construction with the CHG09 protocol [6].

Due to page limitation, we can not describe the detailed analysis, however, our HIBI protocols are less efficient than the CHG09 protocol in key size and communication cost. It is because that we adopt the certificate-based construction, and the certificate-based construction increases both the key size and communication cost. We will discuss this issue minutely in the final version of this paper.

Regarding to the assumption, our HIBI protocol based on a variant of the Waters signature [19] has an advantage than the CHG09 protocol. Though security of the CHG09 protocol depends on the one-more CDH assumption, our protocol is based on the CDH assumption.

Needless to say, all our HIBI protocols are superior to the CHG09 protocol so far as the security model is concerned. Our security proofs are accomplished in the standard model, while the security proof of the CHG09 protocol needs the random oracle model.

5 Conclusion

This paper proposed a generic construction of HIBI protocols secure against impersonation under active and concurrent attacks in the standard model. The proposed construction converts a DS scheme existentially unforgeable against chosen message attacks, where the scheme has a protocol for showing possession of the signature generation key. Our construction is based on certificate-based construction of hierarchical identity-based cryptosystems, and utilizes a variant of the well-known OR-proof technique to ensure the security against impersonation under active and concurrent attacks.

We also presented several concrete examples of our construction employing the Waters signature, and other signatures. As results, its imp-ca security of each instantiation is proved under the CDH assumption, the RSA assumption, or their variants in the standard model.

References

1. Bellare, M., Namprempre, C., Neven, G.: Security proofs for identity-based identification and signature schemes. Journal of Cryptology 22(1), 1–61 (2009)
2. Bellare, M., Palacio, A.: GQ and Schnorr Identification Schemes: Proofs of Security against Impersonation under Active and Concurrent Attacks. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 162–177. Springer, Heidelberg (2002)
3. Bellare, M., Rogaway, P.: Random oracle are practical: A paradigm for designing efficient protocols. In: CCS 1993, pp. 62–73. ACM (1993)
4. Boneh, D., Boyen, X.: Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. SIAM Journal on Computing 32(3), 584–615 (2003); A preliminary version appeared In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–615. Springer, Heidelberg (2001)

6. Chin, J.-J., Heng, S.-H., Goi, B.-M.: Hierarchical Identity-Based Identification Schemes. In: Ślęzak, D., Kim, T.-H., Fang, W.-C., Arnett, K.P. (eds.) SecTech 2009. CCIS, vol. 58, pp. 93–99. Springer, Heidelberg (2009)
7. Cramer, R.: Modular Design of Secure, yet Practical Cryptographic Protocols. PhD thesis, University of Amsterdam (1996)
8. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. ACM Transactions on Information and System Security 3(3), 161–185 (2000); A preliminary version appeared in 6th ACM CCS (1999)
9. Fujioka, A., Saito, T., Xagawa, K.: Security Enhancements by OR-Proof in Identity-Based Identification. In: Bao, F., Samarati, P., Zhou, J. (eds.) ACNS 2012. LNCS, vol. 7341, pp. 135–152. Springer, Heidelberg (2012)
10. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
11. Hess, F.: Exponent group signature schemes and efficient identity based signature schemes based on pairings. Cryptology ePrint Archive, Report 2002/012 (2002), http://eprint.iacr.org/2002/012
12. Hohenberger, S., Waters, B.: Short and Stateless Signatures from the RSA Assumption. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 654–670. Springer, Heidelberg (2009), The full version is available at http://eprint.iacr.org/2009/283
13. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
14. Kurosawa, K., Heng, S.-H.: From Digital Signature to ID-based Identification/Signature. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 248–261. Springer, Heidelberg (2004)
15. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010), The full version is available at http://eprint.iacr.org/2009/482
16. Miller, G.L.: Riemann's hypothesis and tests for primality. Journal of Computer and System Sciences 13(3), 300–317 (1976)
17. Poupard, G., Stern, J.: Short Proofs of Knowledge for Factoring. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 147–166. Springer, Heidelberg (2000)
18. 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)
19. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005), The full version is available at http://eprint.iacr.org/2004/180

Efficient Two-Move Blind Signatures in the Common Reference String Model

E. Ghadafi and N.P. Smart

Dept. Computer Science, University of Bristol, Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, United Kingdom
{ghadafi,nigel}@cs.bris.ac.uk

Abstract. Blind signatures provide a mechanism for achieving privacy and anonymity whereby a user gets the signer to sign a message of his choice without the signer learning the message, or being able to link a signature to the protocol run via which it was obtained. In this paper, we construct a blind signature scheme that requires minimal interaction (two moves) between the user and the signer, and which yields standard signatures. The signature request protocol is akin to the classic, blind-unblind methodology used for RSA blind signatures in the random oracle model; whilst the output signature is a standard Camenisch-Lysyanskaya signature in bilinear groups. The scheme is secure in the common reference string model, assuming a discrete logarithm related assumption in bilinear groups; namely a new variant of the LRSW assumption. We provide evidence for the hardness of our new variant of the LRSW assumption by showing it is intractable in the generic group model.

1 Introduction

Background. Since their introduction by Chaum [11], blind signatures have been used in a number of cryptographic applications that require one party (a signer) to authenticate a message for another party (the user), whilst maintaining privacy of the user's message. The classic example of their use is in e-cash protocols [11] where a bank acts as the signer, and the message is a representation of digital cash; the privacy requirement comes from the non-traceability requirement of cash.

A blind signature must satisfy two security requirements [20,28], blindness and unforgeability. Blindness requires that the signer does not learn the message he is signing and in addition, when he later sees the final signature, he cannot link it to the protocol run via which it was obtained. Unforgeability on the other hand requires that the user cannot output any new signatures that he has not asked the signer to sign for him, or in other words, the number of signatures the user can compute is equal to the number of completed interactions he was involved in with the signer.

Since their introduction, a number of authors have presented blind signature algorithms based on different assumptions and in different models. For example, schemes based on factoring related assumptions have been given in the Random Oracle Model (ROM) [3], and in the Common Reference String (CRS) model [10]; schemes based on discrete logarithm related assumptions have been given in the ROM [1,5,28] and in the CRS model [2,14,26]; schemes based on a combination of discrete logarithm and factoring based assumptions have been given in the CRS model [21]; finally, in [13,20] schemes in the CRS model are given under general assumptions.

A blind signature consists of two distinct phases. In the first phase, which we shall call the signature request phase, the user obtains from the signer the signature on the message he requires. In the second phase, the signature and message are made public and anyone can apply the public verification algorithm to verify the message/signature pair. The signature request phase is the most complex of all phases. One could consider such a phase as a general secure two-party computation, where the user has the message as his private input, whereas the signer has a secret key as his private input. After such a secure computation, the user outputs a valid blind signature on his secret message.

The "classic" blind signature schemes are in the ROM and are essentially Full-Domain-Hash (FDH) style signature schemes. In these schemes, the hash function is applied to the message, the result is then blinded and sent to the signer. The signer signs the blinded message as he would sign a normal message. On receiving the output from the signer the user then unblinds the signature using a homomorphic property of the underlying signature scheme. Such a mechanism is the basis of the original RSA based scheme of Chaum [11], which was proved secure in [3]; Chaum's scheme outputs standard RSA-FDH signatures [4]. It also forms a basis of the BLS signature [7] based blind signature scheme of Boldyreva [5]. The advantage of such schemes is that the output signature corresponds to a standard signature scheme; in these two cases RSA-FDH and BLS respectively. Our construction has a similar flavour in that the signature generation protocol is of the blind/unblind variant and that the output signature is a "standard" signature, namely a Camensich–Lysyanskaya (CL) signature [9]. However, we dispense with the need for random oracles and instead work in the CRS model.

Prior Work in the CRS Model. Due to the round-optimal nature of a two-move signature request phase, and the desire to avoid the use of the random oracle, much recent work has focused on developing round-optimal blind signatures in the CRS model.

In [13], Fischlin presented a scheme in the CRS model which has a two-move signature request protocol. The scheme is a generic construction from basic primitives, namely schemes for commitment, encryption and signatures as well as generic non-interactive zero knowledge (NIZK) proofs for NP-languages. The signature request protocol consists of the user sending a commitment to the message to the signer, who responds with a signature on the commitment. The user then uses this signature on the commitment to construct the blind signature, by first encrypting the commitment and the signature, and then adding NIZK proofs to prove that the encrypted signature is a valid signature on the encrypted commitment, and that the encrypted commitment is a commitment to the specific message.

Using the notion of automorphic signatures, Fuchsbauer [14] (see also [2]) presented a variant of the construction of Fischlin, using specific efficient components. In particular, he made use of the efficient NIZK proofs of Groth and Sahai [18,19] which hold for only specific NP-statements in bilinear groups. In Fuchsbauer's scheme, the blind signature is constructed by providing a Groth–Sahai proof of knowledge of a signature on a message (as opposed to a signature on a commitment as in Fischlin's generic construction). This makes the underlying NIZK proofs simpler, but makes use of a different signature request phase. The resulting blind signature consists of around 30 group elements, and is the most efficient round-optimal blind signature scheme in the CRS model known to date.

Fuchsbauer's scheme is based on a new intractibility assumption called the ADH-SDH assumption, which he shows holds in the Generic Group Model (GGM) [30,23]. This is a falsifiable assumption, in the sense of Naor [25], which is closely related to the q-SDH problem lying behind the Boneh–Boyen signature scheme [6]. However, the resulting blind signature is not a standard signature, e.g. it is not a true Boneh–Boyen signature.

Concurrently to our work, [24] proposed a scheme over composite-order bilinear groups (which are less efficient than their prime-order counterparts due to the necessity of requiring a large group order that is hard to factor). Although the final signatures of their scheme consist of only two group elements, their scheme requires a large communication overhead that linearly grows with the bit-length of the message to be signed. In addition, unlike ours, it requires large public parameters. The more recent work [29] (which was published after our work had been completed) instantiated the scheme of [24] in symmetric prime-order bilinear groups. Similar to [24], their scheme requires both larger communication overhead and much larger public parameters than our scheme.

In this paper, we present a round-optimal blind signature scheme in the CRS model which is significantly more efficient than Fuchsbauer's scheme; a signature only consists of three group elements. Indeed the resulting signature is a standard CL signature on the message m. Furthermore, our scheme requires a low communication overhead and the reference string is minimal. We note that our required hardness assumption, being interactive, is not falsifiable. However, this property is inherited from the underlying CL signature where the underlying hardness assumption is the LRSW assumption, which is itself interactive.

In [16], the authors present a generic round-optimal blind signature scheme in the standard model, which results in blind signatures from *any* standard signature scheme. However, their construction is not as efficient as our construction in the CRS model for the CL signature scheme. In particular, the generic construction of [16] requires the use of ZAPs and two-party secure function evaluation protocols.

Our Scheme. The scheme we present has a number of similarities to previous work, yet a number of distinct advantages. As already remarked it results in standard CL signatures, is optimal in the number of moves in the signing protocol, and dispenses with the need for using the ROM.

Recall that security of the RSA-FDH based blind signature is not based on the same assumption as the original RSA-FDH signature, indeed it is based on a stronger one, the same holds true for the BLS-based blind signature. In our construction the same property w.r.t. assumptions holds; whilst security of the standard CL signature scheme is based on the LRSW assumption [22], the security of our scheme relies on a stronger assumption, which we call the Extended-LRSW (E-LRSW) assumption, which is itself related to the Blind-LRSW (B-LRSW) assumption previously proposed in [12]. We justify the hardness of this new assumption by presenting a proof in the GGM. We note, our proof can be modified to also justify the B-LRSW assumption in the GGM.

We note that the CRS-based scheme in [21] also outputs standard CL signatures, however, the signature request protocol requires factoring-based assumptions to provide security. Our signature request protocol is significantly simpler.

We now state some disadvantages of our scheme. Firstly, to obtain a highly efficient protocol, we work in the honestly-generated keys model; security can easily be obtained in the adversarially-generated keys model with either the addition of an extra round of communication, the addition of NIZK proofs of knowledge, or a more complex setup phase. Secondly, our scheme reduces to an interactive assumption rather than a standard cryptographic assumption. However, this is relatively standard in the construction of efficient blind signature schemes, e.g. [11,5]. It remains an open problem to derive a (truly) efficient round-optimal blind signature scheme in the CRS model which allows adversarially-generated keys, and which reduces to a non-interactive assumption while requiring low communication overhead and a small reference string.

Paper Organization. The rest of the paper is organized as follows; In Section 2 we recap on the syntax and security definitions for blind signatures. In Section 3 we recap on the basic properties of bilinear groups that we shall need, and we will present the underlying hard problems on which the security of our scheme rests. Finally, we present our scheme in Section 4, with the security proof provided in Section 5.

2 Syntax and Security of Blind Signatures

In this section we define the syntax and security of blind signatures. Since we are focusing on signature request phases which are two-move, we specialize the syntax for this case.

Syntax. A blind signature scheme BS (with a two-move signature request phase) in the CRS model consists of six probabilistic polynomial time algorithms

$$\mathsf{BS} = (\mathsf{Setup}_{\mathsf{BS}}, \mathsf{KeyGen}_{\mathsf{BS}}, \mathsf{Request}_{\mathsf{BS}}, \mathsf{Issue}_{\mathsf{BS}}, \mathsf{Unblind}_{\mathsf{BS}}, \mathsf{Verify}_{\mathsf{BS}}).$$

The syntax of these algorithms is defined as follows; where to aid notation all algorithms (bar $\mathsf{Setup}_{\mathsf{BS}}$ and $\mathsf{KeyGen}_{\mathsf{BS}}$) are assumed to take as implicit input $\mathsf{CRS}_{\mathsf{BS}}$ as output by $\mathsf{Setup}_{\mathsf{BS}}$;

– $\mathsf{Setup}_{\mathsf{BS}}(1^\lambda)$: Takes as input a security parameter λ and outputs a common string $\mathsf{CRS}_{\mathsf{BS}}$. We assume $\mathsf{CRS}_{\mathsf{BS}}$ contains a description of the key and message spaces for the scheme.
– $\mathsf{KeyGen}_{\mathsf{BS}}(\mathsf{CRS}_{\mathsf{BS}})$: Takes as input the reference string $\mathsf{CRS}_{\mathsf{BS}}$ and outputs a pair of public/secret keys $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$ for the signer.
– $\mathsf{Request}_{\mathsf{BS}}(m, \mathsf{pk}_{\mathsf{BS}})$: This algorithm, run by the user, takes a message m in the space of messages $\mathcal{M}$ and the public key $\mathsf{pk}_{\mathsf{BS}}$, and produces a signature request ρ, plus some state St (which is assumed to contain m).
– $\mathsf{Issue}_{\mathsf{BS}}(\rho, \mathsf{sk}_{\mathsf{BS}})$: This algorithm, run by the signer, takes the signature request ρ and the secret key $\mathsf{sk}_{\mathsf{BS}}$, and produces a pre-signature β.
– $\mathsf{Unblind}_{\mathsf{BS}}(\beta, \mathsf{St}, \mathsf{pk}_{\mathsf{BS}})$: On input of β, St and the public key $\mathsf{pk}_{\mathsf{BS}}$, this algorithm produces a blind signature σ on m, or it outputs $\perp$.
– $\mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}})$: This is the public signature verification algorithm. This should output 1 if σ is a valid signature on m and 0 otherwise.

Correctness of the blind signature algorithm is that if both parties behave honestly then signatures should verify, i.e. for all CRS's output by $\mathsf{Setup}_{\mathsf{BS}}$ we have,

$$\begin{aligned}\Pr\,[\,&(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}}) \leftarrow \mathsf{KeyGen}_{\mathsf{BS}}(\mathsf{CRS}_{\mathsf{BS}}), m \leftarrow \mathcal{M},\\ &(\rho, \mathsf{St}) \leftarrow \mathsf{Request}_{\mathsf{BS}}(m, \mathsf{pk}_{\mathsf{BS}}), \beta \leftarrow \mathsf{Issue}_{\mathsf{BS}}(\rho, \mathsf{sk}_{\mathsf{BS}}),\\ &\sigma \leftarrow \mathsf{Unblind}_{\mathsf{BS}}(\beta, \mathsf{St}, \mathsf{pk}_{\mathsf{BS}}) : \ \mathsf{Verify}_{\mathsf{BS}}(m, \sigma, \mathsf{pk}_{\mathsf{BS}}) = 1\,] = 1.\end{aligned}$$

Security. The standard security model for blind signatures [20,28] consists of two properties, blindness and unforgeability. Intuitively, blindness says that an adversarial signer who chooses two messages m_0 and m_1 and then interacts with an honest user who requests signatures on those messages (in an order unknown to the signer), is unable to tell the order in which the messages were signed upon being presented with the final unblinded signatures. On the other hand, unforgeability deals with an adversarial user whose goal is to obtain $k+1$ distinct message/signature pairs given only k interactions with the honest signer. To define blindness, we consider an adversary $\mathcal{A}$ which has three modes $\mathsf{find}, \mathsf{issue}$ and guess, running in an experiment as in Figure 1. Note that the experiment is defined for honestly-chosen keys. Our security results will hold, regarding blindness, for adversarially-chosen keys as long as the challenger in the blindness game is given access to the secret key as well.

We can obtain full security against adversarially-chosen keys by simply requesting a secret key holder to prove, in zero-knowledge, knowledge of the underlying secret key for a given public key, and then using, within our proof for blindness, a knowledge extractor for the zero-knowledge proof to extract the witness (i.e. the secret key). The additional security obtained from adversarially-chosen keys comes however at the expense of the zero-knowledge proof. To obtain the same number of rounds, we will require such a proof to be non-interactive, and hence costly with currently known techniques; or it can be efficient and interactive, and hence cost more rounds. Another alternative would be in the setup phase for the signer to prove knowledge of their secret keys via an (interative or non-interactive) zero-knowledge proof. In the interactive case, we would however

```
Experiment $\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$

– $\mathsf{CRS_{BS}} \leftarrow \mathsf{Setup_{BS}}(1^\lambda)$.
– $(\mathsf{pk_{BS}}, \mathsf{sk_{BS}}) \leftarrow \mathsf{KeyGen_{BS}}(\mathsf{CRS_{BS}})$.
– $(m_0, m_1, \mathsf{St_{find}}) \leftarrow \mathcal{A}(\mathsf{find}, \mathsf{pk_{BS}}, \mathsf{sk_{BS}}, \mathsf{CRS_{BS}})$.
– $b \leftarrow \{0,1\}$.
– $(\rho_b, \mathsf{St}_b) \leftarrow \mathsf{Request_{BS}}(m_0, \mathsf{pk_{BS}})$:
– $(\rho_{1-b}, \mathsf{St}_{1-b}) \leftarrow \mathsf{Request_{BS}}(m_1, \mathsf{pk_{BS}})$:
– $(\beta_0, \beta_1, \mathsf{St_{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, \rho_0, \rho_1, \mathsf{St_{find}})$.
– $\sigma_0 \leftarrow \mathsf{Unblind_{BS}}(\beta_b, \mathsf{St}_b, \mathsf{pk_{BS}})$.
– $\sigma_1 \leftarrow \mathsf{Unblind_{BS}}(\beta_{1-b}, \mathsf{St}_{1-b}, \mathsf{pk_{BS}})$.
– If $\sigma_0 = \perp$ or $\sigma_1 = \perp$ Then Set $(\sigma_0, \sigma_1) \leftarrow (\perp, \perp)$.
– $b^* \leftarrow \mathcal{A}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{St_{issue}})$.
– Return 1 If $b = b^*$ Else Return 0.
```

Fig. 1. Security experiment for blindness of blind signatures

```
Experiment $\mathsf{Exp}^{Unforge}_{\mathsf{BS},\mathcal{A}}(\lambda)$

– $\mathsf{CRS_{BS}} \leftarrow \mathsf{Setup_{BS}}(1^\lambda)$.
– $(\mathsf{pk_{BS}}, \mathsf{sk_{BS}}) \leftarrow \mathsf{KeyGen_{BS}}(\mathsf{CRS_{BS}})$.
– $\{(m_i, \sigma_i)\}_{i=1}^{k+1} \leftarrow \mathcal{A}^{\mathsf{Issue_{BS}}(\cdot, \mathsf{sk_{BS}})}(\mathsf{pk_{BS}}, \mathsf{CRS_{BS}})$.
– Return 0 If any of the following is true:
    • $\mathcal{A}$ called its oracle more than $k$ times.
    • $\exists i, j \in \{1, \ldots, k+1\}$, with $i \neq j$, s.t. $m_i = m_j$.
    • $\exists i \in \{1, \ldots, k+1\}$ s.t. $\mathsf{Verify_{BS}}(m_i, \sigma_i, \mathsf{pk_{BS}}) = 0$.
– Else Return 1.
```

Fig. 2. Security experiment for unforgeability of blind signatures

require the ability to rewind the signer to the start of the setup phase in order to extract the secret within our proof.

Our focus is on a protocol which is efficient in the honestly-generated keys model, but extending our results (admittedly with a loss of efficiency) to the adversarially-generated keys model in one of the ways described above is trivial. We define $\mathsf{Adv}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = \left|2 \cdot \Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1] - 1\right|$ and we say that the scheme is *blind* if $\mathsf{Adv}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time adversary $\mathcal{A}$.

To define unforgeability, we consider an adversary $\mathcal{A}$, having oracle access to $\mathsf{Issue_{BS}}(\cdot, \mathsf{sk_{BS}})$, for adversarially-chosen first parameter, running in an experiment as in Figure 2. We define the advantage $\mathsf{Adv}^{Unforge}_{\mathsf{BS},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{Unforge}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1]$ and we say that the scheme is *unforgeable* if the advantage $\mathsf{Adv}^{Unforge}_{\mathsf{BS},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time adversary $\mathcal{A}$.

Note that our definition of forgery is not that of strong unforgeability; we do not require the adversary to not be able to output a new signature on an old message. This is because the final signature in our scheme will be a CL signature, which is always randomizable. Hence, our blind signature scheme may not be suitable in applications which require the strong unforgeability definition.

3 Bilinear Groups and Associated Hard Problems

In this section, we introduce the basic mathematical constructs needed to present our scheme.

Bilinear Groups. Bilinear groups are a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, along with a bilinear map $\hat{t} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$. We shall write $\mathbb{G}_1$ and $\mathbb{G}_2$ additively (with identity element 0), and $\mathbb{G}_T$ multiplicatively (with identity element 1), and write $\mathbb{G}_1 = \langle P_1 \rangle, \mathbb{G}_2 = \langle P_2 \rangle$, for two explicitly given generators P_1 and P_2. We let $\mathbb{G}_i^\times = \mathbb{G}_i \setminus \{0\}$ for $i = 1, 2$. Multiplication by an integer x in the group $\mathbb{G}_1$ (resp. $\mathbb{G}_2$) will be denote by $[x]P_1$ (resp. $[x]P_2$). We define $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{t}, P_1, P_2)$ to be the set of pairing group parameters.

The function $\hat{t}$ must have the following three properties:

1. Bilinearity: $\forall Q_1 \in \mathbb{G}_1$, $\forall Q_2 \in \mathbb{G}_2$, $\forall x, y \in \mathbb{Z}$, we have $\hat{t}([x]Q_1, [y]Q_2) = \hat{t}(Q_1, Q_2)^{xy}$.
2. Non-Degeneracy: The value $\hat{t}(P_1, P_2)$ generates $\mathbb{G}_T$.
3. The function $\hat{t}$ is efficiently computable.

Following [15], we categorize pairings into three distinct types (other types are possible, but the following three are the main ones utilized in practical protocols).

– **Type-1**: This is the symmetric pairing setting in which $\mathbb{G}_1 = \mathbb{G}_2$.
– **Type-2**: Here we have $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$ where $\psi(P_2) = P_1$.
– **Type-3**: Again $\mathbb{G}_1 \neq \mathbb{G}_2$, but now there is no known efficiently computable isomorphism.

In this paper, we shall always consider Type-3 pairings. Such pairings can be efficiently realized; by taking $\mathbb{G}_1$ to be the set of points of order p of an elliptic curve over $\mathbb{F}_q$ with a "small" embedding degree k; by taking $\mathbb{G}_2$ to be the set of points of order p on a twist of the same elliptic curve over $\mathbb{F}_{q^e}$, for some divisor e of k; and $\mathbb{G}_T$ to be the subgroup of order p in the finite field $\mathbb{F}_{q^k}$.

For a security parameter λ, we let $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$ denote an algorithm which produces a pairing group instance $\mathcal{P}$ of Type-3.

Hard Problems. The security of our scheme rests on a variant of the following assumption, introduced in [22] in the case of Type-1 pairings; we present this problem in its generality for all pairings,

Definition 1 (LRSW Assumption). *If $\mathcal{A}$ is an algorithm which is given access to an oracle $\mathcal{O}_{[x]P_2,[y]P_2}(\cdot)$ that on input of $m \in \mathbb{F}_p$ outputs $(A, B, C) = (A, [y]A, [x + m \cdot x \cdot y]A)$, for some random $A \in \mathbb{G}_1^\times$, we let Q denote the set of queries made by $\mathcal{A}$ to $\mathcal{O}_{[x]P_2,[y]P_2}(\cdot)$.*

The LRSW assumption is said to hold for the output of $\mathsf{Setup}_{\mathsf{Grp}}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$, and all outputs of $\mathsf{Setup}_{\mathsf{Grp}}$, the following probability is negligible in the security parameter λ,

$$\begin{aligned}\Pr\,[&x, y \leftarrow \mathbb{F}_p,\ X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2,\\ &(Q, m, A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}_{X,Y}(\cdot)}(\mathcal{P}, X, Y)\\ &:\ m \in \mathbb{F}_p \setminus Q \cup \{0\}\ \wedge\ A \in \mathbb{G}_1^\times\\ &\ \wedge\ B = [y]A\ \wedge\ C = [x + m \cdot x \cdot y]A\,]\end{aligned}$$

In [22], it was shown that the LRSW assumption holds in the GGM and is independent of the DDH assumption. The LRSW assumption is the underlying hard problem behind the Camenisch–Lysyanskaya (CL) signature scheme.

Definition 2 (Camenisch–Lysyanskaya Signature Scheme). *The CL signature scheme is given by the following triple of algorithms given an output $\mathcal{P}$ of $\mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$.*

- $\mathsf{KeyGen}(\mathcal{P})$: *The secret key is $\mathsf{sk}_{CL} \leftarrow (x, y) \in \mathbb{F}_p^2$, whereas the public key is $\mathsf{pk}_{CL} \leftarrow (X, Y) = ([x]P_2, [y]P_2)$.*
- $\mathsf{Sign}(m, \mathsf{sk}_{CL})$: *Select $A \leftarrow \mathbb{G}_1^\times$, and then set $B \leftarrow [y]A$, $C = [x + m \cdot x \cdot y]A$. Output (A, B, C).*
- $\mathsf{Verify}(m, (A, B, C), \mathsf{pk}_{CL})$: *Output true if and only if $\hat{t}(B, P_2) = \hat{t}(A, Y)$ and $\hat{t}(C, P_2) = \hat{t}(A, X) \cdot \hat{t}(B, X)^m$.*

Indeed, the LRSW problem and the EF-CMA security of the CL signature scheme are equivalent since the oracle in the LRSW assumption produces CL signatures, and the output of the adversary against the LRSW assumption corresponds precisely to a forger who can construct one more CL signature. This is mirrored below in the relationship between the hardness of our E-LRSW assumption and the unforgeability of our scheme.

We shall require a strengthening of the LRSW assumption, a variant of which was first proposed by Chen et al. [12]. We first present the strengthening of Chen et al., called the Blind-LRSW (B-LRSW) assumption, and then we present our modification which we call the Extended-LRSW (E-LRSW) assumption.

In the B-LRSW problem, the oracle provided to the adversary does not take as input an element $m \in \mathbb{F}_p$, but instead takes as input an element $M \in \mathbb{G}_1$. The output of the oracle is a CL signature on the discrete logarithm m of M with respect to P_1. The output of the adversary against this modified LRSW assumption is still a valid CL signature on a new element m. It was called the Blind-LRSW assumption because the adversary is given access to an oracle which can produce what are in effect blinded signatures. Formally we define

Definition 3 (B-LRSW Assumption). *If $\mathcal{A}$ is an algorithm which is given access to an oracle, denoted by $\mathcal{O}^B_{[x]P_2,[y]P_2}(\cdot)$, that on input of $M = [m]P_1 \in \mathbb{G}_1$ outputs $(A, B, C) = (A, [y]A, [x + m \cdot x \cdot y]A)$, for some random $A \in \mathbb{G}_1^\times$, we let Q denote the set of queries made by $\mathcal{A}$ to $\mathcal{O}^B_{[x]P_2,[y]P_2}(\cdot)$.*

The B-LRSW assumption is said to hold for the output of $\mathsf{Setup}_{\mathsf{Grp}}$ if for all probabilistic polynomial time adversaries $\mathcal{A}$, and all outputs of $\mathsf{Setup}_{\mathsf{Grp}}$, the following probability is negligible in the security parameter λ,

$$\begin{aligned}\Pr\,[&x, y \leftarrow \mathbb{F}_p,\ X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2,\\ &(Q, m, A, B, C) \leftarrow \mathcal{A}^{\mathcal{O}^B_{X,Y}(\cdot)}(\mathcal{P}, X, Y)\\ &: [m]P_1 \notin Q\ \wedge\ m \in \mathbb{F}_p \setminus \{0\}\ \wedge\ A \in \mathbb{G}_1^\times\\ &\wedge\ B = [y]A\ \wedge\ C = [x + m \cdot x \cdot y]A\ ]\end{aligned}$$

Note that the oracle in the B-LRSW assumption, given access to x and y, can compute its response by choosing $a \in \mathbb{F}_p$ at random and outputing the triple $([a]P_1, [a \cdot y]P_1, [a \cdot x]P_1 + [a \cdot x \cdot y]M)$, i.e. the oracle does not need to be able to solve discrete logarithms if it has access to x and y.

Also note, since an oracle query to $\mathcal{O}_{X,Y}(\cdot)$ can be simulated with an oracle query to $\mathcal{O}^B_{X,Y}(\cdot)$, and the output of the adversaries in the two problems is essentially identical, that an adversary $\mathcal{A}$ against the LRSW assumption can be turned into an adversary against the B-LRSW assumption (but it appears not vice-versa). Thus, the B-LRSW assumption is stronger than the LRSW assumption.

In our hard problem, shown to hold in the GGM, we extend the oracle in the B-LRSW assumption to provide some additional data about the values A, x and y, with respect to a new public key element $Z = [z]P_1$. The output of the adversary still being a Camenisch–Lysyanskaya signature.

Definition 4 (E-LRSW Assumption). *If $\mathcal{A}$ is an algorithm which is given access to an oracle, denoted by $\mathcal{O}^E_{[x]P_2,[y]P_2,[z]P_1}(\cdot)$, that on input of $M = [m]P_1 \in \mathbb{G}_1$ (for some unknown value of m) outputs $(A, B, C, D) = (A, [y]A, [x + m \cdot x \cdot y]A, [x \cdot y \cdot z]A)$, for some random $A \in \mathbb{G}_1^\times$, we let q denote the number of queries made by $\mathcal{A}$ to $\mathcal{O}^E_{[x]P_2,[y]P_2,[z]P_1}(\cdot)$.*

The E-LRSW assumption is said to hold for the output of $\mathsf{Setup}_{\mathsf{Grp}}$, if for all probabilistic polynomial time adversaries $\mathcal{A}$, and all outputs of $\mathsf{Setup}_{\mathsf{Grp}}$, the following probability is negligible in the security parameter λ,

$$\begin{aligned}\Pr\,[&x, y, z \leftarrow \mathbb{F}_p, X \leftarrow [x]P_2,\ Y \leftarrow [y]P_2,\ Z \leftarrow [z]P_1,\\ &\{(m_i, A_i, B_i, C_i)\}_{i=1}^{q+1} \leftarrow \mathcal{A}^{\mathcal{O}^E_{X,Y,Z}(\cdot)}(\mathcal{P}, X, Y, Z)\\ &:\ m_i \in \mathbb{F}_p \setminus \{0\} \textit{ for all } 1 \le i \le q+1\ \wedge\ A_i \in \mathbb{G}_1^\times\ \wedge\ B_i = [y]A_i\\ &\ \wedge\ C_i = [x + m_i \cdot x \cdot y]A_i\ \wedge\ (\textit{if } i \neq j \textit{ then } m_i \neq m_j)\ ]\end{aligned}$$

Note we present this assumption in terms of a one-more problem rather than in the form of the B-LRSW assumption. This is because the extra item D allows the adversary to "open" the signature in one of a number of ways, if he can recover z by solving the underlying discrete logarithm problem. However, the intuition is that he needs to commit to how he is going to open the signature before he sends the request to the oracle. We formalize this intuition when we present a proof in the GGM of the difficulty of the E-LRSW problem.

We note that our E-LRSW assumption is in some sense to the LRSW assumption, as the HSDH assumption from [8] is to the SDH assumption. In that the "queries" $([1/(x + c_i)]P_1, c_i)$ in the SDH assumption are replaced by "blinded queries" $([1/(x + c_i)]P_1, [c_i]Q_1, [c_i]P_2)$ in the HSDH assumption, and the output value $([1/(x + c^*)]P_1, c^*)$ in the SDH assumption is replaced by the blinded output value $([1/(x + c^*)]P_1, [c^*]Q_1, [c^*]P_2)$ in the HSDH assumption, where $P_1 \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$, $Q_1 \leftarrow \mathbb{G}_1$, $x, c_i \leftarrow \mathbb{Z}_p^*$ and some $c^* \in \mathbb{Z}_p^*$ where $c^* \notin \{c_i\}$.

Recall the blind signature scheme below is akin to the blind-unblind schemes in the ROM, such as those based on RSA. It should therefore not be surprising that we need to strengthen the security assumption of the underling signature scheme so as to cope with blindness properties. This is exactly what is required for the ROM-based RSA and discrete logarithm based schemes [3,5].

The following theorem proves that the assumption holds in the GGM.

Theorem 1. *Let $\mathcal{A}$ denote an adversary in the generic group model against the E-LRSW assumption. Assume $\mathcal{A}$ makes q_G group operation queries, q_P pairing queries and q_O queries to the E-LRSW oracle $\mathcal{O}^E$. If we set $n = 5+q_G+4q_O+q_P$ then the probability of the adversary winning the E-LRSW game is $O(n^2 \cdot q_O/p)$, where p is the (prime) order of the generic groups.*

Due to lack of space, we provide the proof for this theorem in the full paper [17].

4 Our Scheme

In Figure 3, we define the algorithms which make up our blind signature scheme. Notice, that β contains a CL signature on the "hidden" message $m + z \cdot r$, but no party knows the value of z. The signer is signing a message he does not know. Indeed, the user also does not know the message (unless he picks $r = 0$), but he is able to unblind this signature to produce a valid CL signature on m using his value r. To see that the unblinded signature is valid, notice that the value of C (before multiplication by t) is equal to

$$\begin{aligned} C &= ([a \cdot x]P_1 + [a \cdot x \cdot y]\mathsf{Co}) - [r]D, \\ &= [x]A + [a \cdot x \cdot y] \cdot ([m]P_1 + [r]Z) - [r]D, \\ &= [x + m \cdot x \cdot y]A + [a \cdot x \cdot y \cdot r]Z - [r]D, \\ &= [x + m \cdot x \cdot y]A. \end{aligned}$$

Then notice, that even the revealed signature provides no linkage with the values signed, due to the fact that CL signatures for Type-3 pairings are unlinkable once randomized. In our security proof we will confirm this intuition. Besides yielding short signatures (i.e. of size $\mathbb{G}_1^3$), our scheme has a low communication complexity where the user sends only one element from $\mathbb{G}_1$, whereas the signer's reply consists of 4 group elements from $\mathbb{G}_1$. In addition, our scheme has a very short public key and CRS which are of size $\mathbb{G}_2^2$ and a single element from $\mathbb{G}_1$, respectively. Note, that in a concrete setting to obtain security equivalent to 128-bit symmetric key sizes one can select parameters such that the number of bits needed to represent an element of $\mathbb{G}_1$ is 256.

```
Setup_BS(1^λ)
  – $\mathcal{P} \leftarrow \mathsf{Setup}_{\mathsf{Grp}}(1^\lambda)$.
  – $z \leftarrow \mathbb{F}_p$, $Z \leftarrow [z]P_1$.
  – $\mathcal{M} := \mathbb{F}_p \setminus \{0\}$.
  – $\mathsf{CRS}_{\mathsf{BS}} \leftarrow (\mathcal{P}, Z, \mathcal{M})$.
  – Output $\mathsf{CRS}_{\mathsf{BS}}$.
```

```
KeyGen_BS(CRS_BS)
  – $x, y \leftarrow \mathbb{F}_p$.
  – $X \leftarrow [x]P_2$.
  – $Y \leftarrow [y]P_2$.
  – $\mathsf{sk}_{\mathsf{BS}} \leftarrow (x, y)$, $\mathsf{pk}_{\mathsf{BS}} \leftarrow (X, Y)$.
  – Output $(\mathsf{pk}_{\mathsf{BS}}, \mathsf{sk}_{\mathsf{BS}})$.
```

```
Request_BS(m, pk_BS)
  – $r \leftarrow \mathbb{F}_p$.
  – $\mathsf{Co} \leftarrow [m]P_1 + [r]Z$.
  – $\rho \leftarrow \mathsf{Co}$, $\mathsf{St} \leftarrow (m, r)$.
  – Output $(\rho, \mathsf{St})$.
```

```
Issue_BS(ρ, sk_BS)
  – Parse $\rho$ as $\mathsf{Co}$.
  – $a \leftarrow \mathbb{F}_p \setminus \{0\}$.
  – $A \leftarrow [a]P_1$.
  – $B \leftarrow [a \cdot y]P_1$.
  – $C \leftarrow [a \cdot x]P_1 + [a \cdot x \cdot y]\mathsf{Co}$.
  – $D \leftarrow [a \cdot x \cdot y]Z$.
  – $\beta \leftarrow (A, B, C, D)$.
  – Output $\beta$.
```

```
Unblind_BS(β, St, pk_BS)
  – Parse $\beta$ as $(A, B, C, D)$.
  – Parse $\mathsf{St}$ as $(m, r)$.
  – $C \leftarrow C - [r]D$.
  – If $\mathsf{Verify}_{\mathsf{BS}}(m, (A, B, C), \mathsf{pk}_{\mathsf{BS}}) = 0$
      • Return $\perp$.
  – $t \leftarrow \mathbb{F}_p \setminus \{0\}$.
  – $A \leftarrow [t]A$, $B \leftarrow [t]B$, $C \leftarrow [t]C$.
  – $\sigma \leftarrow (A, B, C)$.
  – Output $\sigma$.
```

```
Verify_BS(m, σ, pk_BS)
  – Parse $\sigma$ as $(A, B, C)$.
  – If $A = 0$ or $\hat{t}(A, Y) \neq \hat{t}(B, P_2)$
    or $\hat{t}(C, P_2) \neq \hat{t}(A, X) \cdot \hat{t}(B, X)^m$
      • Return 0.
  – Return 1.
```

Fig. 3. Our blind signature scheme

5 Proof of Security

First, we prove that the scheme is blind, then we show that it is unforgeable.

Theorem 2. *The above blind signature scheme is perfectly blind. In particular, if $\mathcal{A}$ is an adversary against the blindness of the above blind signature scheme, then*

$$\mathsf{Adv}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 0.$$

Proof. We reduce blindness to the hiding property of Pedersen commitments [27], which is defined by the experiment in Figure 4:

If $\mathcal{C}$ is an adversary in the experiment in Figure 4 then we define

$$\mathsf{Adv}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda) = |2 \cdot \Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda) = 1] - 1|.$$

That Pedersen commitments are perfectly hiding is a classic result. Thus, we have $\mathsf{Adv}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda) = 0$, i.e. $\Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda) = 1] = 1/2$.

We now turn to the security game for blindness for our blind signature scheme. We let G_0 denote the experiment played by the adversary $\mathcal{A}$ against the blindness property of the scheme, see Figure 5. We let E denote the event that the guess

```
Experiment $\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda)$

– $\mathsf{sk} \leftarrow \mathbb{F}_p$, $b \leftarrow \{0,1\}$, $r_0, r_1 \leftarrow \mathbb{F}_p$.
– $\mathsf{pk} \leftarrow [\mathsf{sk}]P_1$.
– $(m_0, m_1, \mathsf{St}) \leftarrow \mathcal{C}_1(\mathsf{pk})$, with $m_i \in \mathbb{F}_p$.
– $C_0 \leftarrow [m_b]P_1 + [r_0]\mathsf{pk}$.
– $C_1 \leftarrow [m_{1-b}]P_1 + [r_1]\mathsf{pk}$.
– $b' \leftarrow \mathcal{C}_2(C_0, C_1, \mathsf{St})$.
– If $b = b'$ then return 1, else return 0.
```

Fig. 4. Security experiment for the hiding property of Pedersen commitment

```
Game $\mathsf{G}_0$
– $z \leftarrow \mathbb{F}_p$, $Z \leftarrow [z]P_1$.
– $x, y \leftarrow \mathbb{F}_p$, $(X, Y) \leftarrow ([x]P_2, [y]P_2)$.
– $(m_0, m_1, \mathsf{St}_{\mathsf{find}}) \leftarrow \mathcal{A}(\mathsf{find}, (X, Y), (x, y), Z)$.
– $b \leftarrow \{0,1\}$.
– $r_0, r_1 \leftarrow \mathbb{F}_p \setminus \{0\}$.
– $\mathsf{Co}_b \leftarrow [m_0]P_1 + [r_0]Z$, $\mathsf{Co}_{1-b} \leftarrow [m_1]P_1 + [r_1]Z$.
– $(\beta_0, \beta_1, \mathsf{St}_{\mathsf{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, \mathsf{Co}_0, \mathsf{Co}_1, \mathsf{St}_{\mathsf{find}})$.
– Parse $\beta_0$ as $(A_b, B_b, C_b, D_b)$ and $\beta_1$ as $(A_{1-b}, B_{1-b}, C_{1-b}, D_{1-b})$.
– $C_0 \leftarrow C_0 - [r_0]D_0$, $C_1 \leftarrow C_1 - [r_1]D_1$.
– $t_0, t_1 \leftarrow \mathbb{F}_p \setminus \{0\}$.
– $\sigma_0 \leftarrow ([t_0]A_0, [t_0]B_0, [t_0]C_0)$.
– $\sigma_1 \leftarrow ([t_1]A_1, [t_1]B_1, [t_1]C_1)$.
– If $\mathsf{Verify}_{\mathsf{BS}}(m_0, \sigma_b, \mathsf{pk}_{\mathsf{BS}}) = 0$ or $\mathsf{Verify}_{\mathsf{BS}}(m_1, \sigma_{1-b}, \mathsf{pk}_{\mathsf{BS}}) = 0$
    • $\sigma_0 \leftarrow \perp$ and $\sigma_1 \leftarrow \perp$.
– $b^* \leftarrow \mathcal{A}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{St}_{\mathsf{issue}})$.
– Return 1 if $b = b^*$ else return 0.
```

Fig. 5. Game G_0

stage of the adversary is passed the pair $(\sigma_0, \sigma_1) = (\perp, \perp)$. We clearly have

$$\begin{aligned}\Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1] \leq & \Pr[E] \cdot \Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1 | E] \\ & + \Pr[\neg E] \cdot \Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1 | \neg E].\end{aligned}$$

It is clear that in the case that the event E happens, then the adversary $\mathcal{A}$ can be turned into an adversary against the Pedersen commitments Co_0 and Co_1. We hence have that

$$\Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1 | E] = \Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda)] = 1/2.$$

In the following, we will show that we also have

$$\Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1 | \neg E] = \Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda)] = 1/2,$$

and so

$$\Pr[\mathsf{Exp}^{Blind}_{\mathsf{BS},\mathcal{A}}(\lambda) = 1] \leq (\Pr[E] + \Pr[\neg E])/2 = 1/2,$$

from which our result will follow.

So from now on assume that the event $\neg E$ happens. In which case there is no need for the challenger to check the returned pre-signatures are valid. Thus, the challenger can obtain valid signatures by ignoring the values returned by the adversary and simply generating the signatures himself; since he knows the secret key. Hence, from game G_0 we can make a hop to game G_1 shown in Figure 6.

```
Game G_1
  – $z \leftarrow \mathbb{F}_p$, $Z \leftarrow [z]P_1$.
  – $x, y \leftarrow \mathbb{F}_p$, $(X, Y) \leftarrow ([x]P_2, [y]P_2)$.
  – $(m_0, m_1, \mathsf{St}_{\mathsf{find}}) \leftarrow \mathcal{A}(\mathsf{find}, (X, Y), (x, y), Z)$.
  – $b \leftarrow \{0, 1\}$.
  – $r_0, r_1 \leftarrow \mathbb{F}_p \setminus \{0\}$.
  – $\mathsf{Co}_b \leftarrow [m_0]P_1 + [r_0]Z$, $\mathsf{Co}_{1-b} \leftarrow [m_1]P_1 + [r_1]Z$.
  – $(\beta_0, \beta_1, \mathsf{St}_{\mathsf{issue}}) \leftarrow \mathcal{A}(\mathsf{issue}, \mathsf{Co}_0, \mathsf{Co}_1, \mathsf{St}_{\mathsf{find}})$.
  – $a_0, a_1 \leftarrow \mathbb{F}_p \setminus \{0\}$.
  – $\sigma_0 \leftarrow ([a_0]P_1, [a_0 \cdot y]P_1, [a_0 \cdot x + a_0 \cdot x \cdot y \cdot m_0]P_1)$.
  – $\sigma_1 \leftarrow ([a_1]P_1, [a_1 \cdot y]P_1, [a_1 \cdot x + a_1 \cdot x \cdot y \cdot m_1]P_1)$.
  – $b^* \leftarrow \mathcal{A}(\mathsf{guess}, \sigma_0, \sigma_1, \mathsf{St}_{\mathsf{issue}})$.
  – Return 1 if $b = b^*$ else return 0.
```

Fig. 6. Game G_1

If we let $\mathcal{S}_0(m)$ denote the distribution of signatures returned to the final stage of the adversary in game G_0 on message m assuming event $\neg E$ happens, and $\mathcal{S}_1(m)$ the distribution in game G_1, we see that $\mathcal{S}_0(m)$ is identically distributed to $\mathcal{S}_1(m)$. This is because in G_0 we randomize a specific CL signature, whereas in G_1 we produce a new independent CL signature. It is easy to see that these two operations have the same effect on the distribution of the final signature passed to the final stage of the adversary. If we let $\Pr[\mathsf{G}_i]$ denote the probability of the adversary winning in game G_i, we have $\Pr[\mathsf{G}_0|\neg E] = \Pr[\mathsf{G}_1]$.

We now show that an adversary playing G_1, must have zero advantage. To see this note that the value Co returned by $\mathsf{Request}_{\mathsf{BS}}$ is a Pedersen commitment to the value m, hence an adversary in game G_1 can be turned into an adversary against the hiding property of Pedersen commitments. That we have $\Pr[\mathsf{G}_1] = \Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{C}}(\lambda)]$ follows immediately; we simply translate the (m_0, m_1) output by the first stage of adversary $\mathcal{A}$ in game G_1, into the challenge for the Pedersen hiding game; this provides the input into the second stage of adversary $\mathcal{A}$; the input to the third stage of adversary $\mathcal{A}$ is then independent of the values returned by our first game hop.

Note that the above proof holds even if the adversary can select the value of the secret key $\mathsf{sk}_{\mathsf{BS}}$, as long as the challenger is able to extract it so as to answer the queries for game G_1.

We now turn to show that our scheme is unforgeable.

Theorem 3. *If the E-LRSW assumption holds then the above blind signature scheme is unforgeable. In particular, if $\mathcal{A}$ is an adversary against the unforgeability of the above blind signature scheme, then there is an adversary $\mathcal{B}$ which solves the E-LRSW problem such that*

$$\mathsf{Adv}_{\mathsf{BS},\mathcal{A}}^{Unforge}(\lambda) = \mathsf{Adv}_{\mathsf{E-LRSW},\mathcal{B}}(\lambda).$$

Proof. Let $\mathcal{A}$ denote an adversary against the unforgeability experiment of the blind signature scheme. We shall use $\mathcal{A}$ to construct an algorithm which solves the E-LRSW problem. Let $(\mathcal{P}, X, Y, Z)$ denote the input to the E-LRSW problem instance. Algorithm $\mathcal{B}$ sets up the CRS and the public keys for the blind signature scheme by setting $\mathsf{CRS}_{\mathsf{BS}} \leftarrow (\mathcal{P}, Z, \mathbb{F}_p \setminus \{0\})$ and $\mathsf{pk}_{\mathsf{BS}} \leftarrow (X, Y)$.

Algorithm $\mathcal{B}$ now calls adversary $\mathcal{A}$. At some point $\mathcal{A}$ will make one of q queries to its $\mathsf{Issue}_{\mathsf{BS}}$ oracle. Algorithm $\mathcal{B}$ responds to a query on Co as follows: It passes the value Co to its E-LRSW oracle $\mathcal{O}_{X,Y,Z}^{E}$ so as to obtain a tuple

$$(A, B, C, D) = (A, [y]A, [x + t \cdot x \cdot y]A, [x \cdot y \cdot z]A),$$

where $\mathsf{Co} = [t]P_1$. Notice that if $t = m + r \cdot z$ then (A, B, C, D) is a valid response for the commitment to the message m. The tuple (A, B, C, D) is now passed back to algorithm $\mathcal{A}$.

Eventually, $\mathcal{A}$ will terminate by outputting a set of $q+1$ tuples (m_i, A_i, B_i, C_i) where (A_i, B_i, C_i) is a valid CL signature on m_i. By returning this list to its challenger, the adversary thereby solves the E-LRSW assumption.

Note that if we have modified the protocol such that the signer appends a NIZK proof of correctness of its response, then the above proof can be applied in the case where the adversary generates his own secret keys; as long as one adds an additional game-hop to enable the extraction of the witness underlying the NIZK proof system. This is a standard technique so we leave it to the reader.

Acknowledgements. The authors would like to thank the EU projects CACE, eCrypt-2 and the ERC Advanced Grant to Nigel Smart for funding the work in this paper. The second author was supported by the EPSRC under grant EP/H043454/1, and was also partially funded by a Royal Society Wolfson Merit Award, and Google Inc. The authors would like to thank Bogdan Warinschi for useful conversations whilst the work was carried out.

References

1. Abe, M.: A Secure Three-Move Blind Signature Scheme for Polynomially Many Signatures. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 136–151. Springer, Heidelberg (2001)
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-Preserving Signatures and Commitments to Group Elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)

3. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. Journal of Cryptology 16, 185–215 (2008)
4. Bellare, M., Rogaway, P.: The Exact Security of Digital Signatures - How to Sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (1996)
5. Boldyreva, A.: Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In: Desmedt, Y.G. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (2003)
6. Boneh, D., Boyen, X.: Short signatures without random oracles and the SDH assumption in bilinear groups. Journal of Cryptology 21, 149–177 (2008)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. Journal of Cryptology 17, 297–319 (2004)
8. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
9. Camenisch, J., Lysyanskaya, A.: Signature Schemes and Anonymous Credentials from Bilinear Maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Heidelberg (2004)
10. Camenisch, J., Koprowski, M., Warinschi, B.: Efficient Blind Signatures Without Random Oracles. In: Blundo, C., Cimato, S. (eds.) SCN 2004. LNCS, vol. 3352, pp. 134–148. Springer, Heidelberg (2005)
11. Chaum, D.: Blind signatures for untraceable payments. In: Advances in Cryptology – CRYPTO 1982, pp. 199–203 (1982)
12. Chen, L., Morrissey, P., Smart, N.P.: DAA: Fixing the pairing based protocols. Cryptology ePrint Archive, Report 2009/198 (2009), http://eprint.iacr.org/
13. Fischlin, M.: Round-Optimal Composable Blind Signatures in the Common Reference String Model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)
14. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320 (2009), http://eprint.iacr.org/
15. Galbraith, S., Paterson, K., Smart, N.P.: Pairings for cryptographers. Discrete Appl. Math. 156, 3113–3121 (2008)
16. Garg, S., Rao, V., Sahai, A., Schröder, D., Unruh, D.: Round Optimal Blind Signatures. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 630–648. Springer, Heidelberg (2011)
17. Ghadafi, E., Smart, N.P.: Efficient two-move blind signatures in the common reference string model. Cryptology ePrint Archive, Report 2010/568 (2010), http://eprint.iacr.org/
18. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155 (2007), http://eprint.iacr.org/
19. Groth, J., Sahai, A.: Efficient Non-interactive Proof Systems for Bilinear Groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)
20. Juels, A., Luby, M., Ostrovsky, R.: Security of Blind Digital Signatures. In: Kaliski Jr., B.S. (ed.) CRYPTO 1997. LNCS, vol. 1294, pp. 150–164. Springer, Heidelberg (1997)
21. Kiayias, A., Zhou, H.-S.: Concurrent Blind Signatures Without Random Oracles. In: De Prisco, R., Yung, M. (eds.) SCN 2006. LNCS, vol. 4116, pp. 49–62. Springer, Heidelberg (2006)

22. Lysyanskaya, A., Rivest, R.L., Sahai, A., Wolf, S.: Pseudonym Systems (Extended Abstract). In: Heys, H.M., Adams, C.M. (eds.) SAC 1999. LNCS, vol. 1758, pp. 184–199. Springer, Heidelberg (2000)
23. Maurer, U.: Abstract Models of Computation in Cryptography. In: Smart, N.P. (ed.) Cryptography and Coding 2005. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (2005)
24. Meiklejohn, S., Shacham, H., Freeman, D.M.: Limitations on Transformations from Composite-Order to Prime-Order Groups: The Case of Round-Optimal Blind Signatures. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 519–538. Springer, Heidelberg (2010)
25. Naor, M.: On Cryptographic Assumptions and Challenges. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 96–109. Springer, Heidelberg (2003)
26. Okamoto, T.: Efficient Blind and Partially Blind Signatures Without Random Oracles. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 80–99. Springer, Heidelberg (2006)
27. 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)
28. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of Cryptology 13, 361–396 (2000)
29. Seo, J.H., Cheon, J.H.: Beyond the Limitation of Prime-Order Bilinear Groups, and Round Optimal Blind Signatures. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 133–150. Springer, Heidelberg (2012)
30. Shoup, V.: Lower Bounds for Discrete Logarithms and Related Problems. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (1997)

Compliance Checking for Usage-Constrained Credentials in Trust Negotiation Systems

Jinwei Hu[1], Khaled M. Khan[2], Yun Bai[3], and Yan Zhang[3]

[1] Department of Computer Science, TU Darmstadt, Germany
hu@mais.informatik.tu-darmstadt.de

[2] Department of Computer Science and Engineering, Qatar University, Qatar
k.khan@qu.edu.qa

[3] School of Computing and Mathematics, University of Western Sydney, Australia
{ybai,yan}@scm.uws.edu.au

Abstract. We propose an approach to placing usage-constraints on RT credentials; issuers specify constraints by designing non-deterministic finite automata. We show by examples that this approach can express constraints of practical interest. We present a compliance checker in the presence of usage-constraints, especially for trust negotiation systems. Given an RT policy, the checker is able to find all minimal satisfying sets, each of which uses credentials in a way consistent with given constraints. The checker leverages answer set programming, a declarative logic programming paradigm, to model and solve the problem. We also show preliminary experimental results: supporting usage-constraints on credentials incurs affordable overheads and the checker responds efficiently.

1 Introduction

Compliance checking aims to, given a policy p and a set $\mathcal{C}$ of credentials, answer the questions of whether and how a subset of $\mathcal{C}$ satisfies p. Such a subset is called a *satisfying set* of p in $\mathcal{C}$. A credential is a cryptographic certificate from a credential issuer, who asserts attributes about a principal. For example, "University says Alice is a student" is a credential, where University is the issuer, Alice is the principal, and "is-a-student" is the attribute. A policy is a statement to be proved, e.g., "Alice can access files".

Compliance checkers can be broadly categorized into three types [9]. Type-1 checkers return no satisfying set in any case but a Boolean value indicating whether p is satisfied. Type-2 checkers return one satisfying set of p, if any. Type-3 checkers are able to find *all* minimal satisfying sets of p in $\mathcal{C}$. This feature distinguishes types-3 checkers from the other two and makes them proper checkers for trust negotiation (TN) systems [6,11]. In TN, two participants, say Alice and Bob, iteratively exchange credentials to gradually increase trust between each other. When requesting (sensitive) attributes of Alice, Bob may receive a (release) policy p from Alice, which specifies attributes that Bob should exhibit before Alice discloses her attribute. In this case, Bob uses his compliance checker to search for satisfying sets of p in his credential set. There are two main reasons why type-3 checkers are more appropriate for TN than type-1 and type-2 checkers. On the one hand, a type-3 checker ensures that TN establishes trust whenever possible. In event of not establishing trust with a negotiation tactic, TN may restart negotiation with an alternative tactic, owing to the use of type-3 checkers. On the other hand, it also broadens choices of negotiation strategies. For example, one may choose to disclose the set of credentials that reveals least sensitive information.

When a compliance checker finds a satisfying set of a policy, it actually composes a proof of the policy (e.g., as defined in Section 4.1), using the credentials in the set. Credential issuers, however, may not foresee all the proofs that their credentials contribute to and all the consequences that their credentials make possible, largely because of the open nature of TN systems. This uncertainty annoys credential issuers sometimes; they may want to restrict the circumstances under which their credentials are used. For example, a bank may require its credentials to be used for limited purposes set out in agreements.[1] Therefore, credential issuers may specify a constraint stating valid ways to use its credentials. For example, the bank may attach to its credentials a constraint listing proofs allowed by the agreements. Compliance checkers (e.g., Bob's checker) ought to work out satisfying sets which prove a policy in a way consistent with constraints. A principal who uses the proof (e.g., Alice) should verify the consistency.

In this paper, we propose such a compliance checker. We first devise a mechanism for credential issuers to specify usage-constraints. A constraint is defined based on NFAs (*non-deterministic finite automaton*). We employ ASP (*answer set programming*) to encode the checking problem and ASP solvers to compute all satisfying sets.

The remainder of the paper is organized as follows. We state the assumptions made in this work and review RT (a family of role-base trust-management languages) in Section 2. We discuss related work in Section 3. In Section 4 we define usage-constraints on credentials and propose the type-3 compliance checking problem in the presence of constraints. Section 5 describes an ASP encoding of the problem, while the detailed encoding is presented in Appendix A. In Section 6 we undertake a group of experiments to evaluate the performance of our approach. Finally, we conclude in Section 7.

2 Background

Assumptions When Bob searches for proofs of Alice's policy p, we assume that Bob's credential set is fixed. That is, the compliance checker works on a static set of credentials. We refer to this set as a *credential context*.

After receiving a proof, Alice is obliged to verify any usage-constraints on credentials that are used in the proof. This is because any otherwise forbidden access that results from a proof not complying with constraints may only cause loss to Alice. For example, suppose that a companyC issues a qualification certificate to Bob but puts a constraint that the certificate be used only in companyC. Suppose further Alice accepts a proof using this certificate and therefore allows Bob to access her sensitive files. The companyC is not held responsible for this access, because it already states the restriction on the certificate. On the contrary, it is Alice who did not enforce the constraint.

[1] `http://www.nbnz.co.nz/onlineservices/directtrade/PDFs/PKIagreements.pdf`

Policy language RT [8] is a family of role-base trust-management languages. We assume that TN uses two of its sub-language, RT_0 and RT_1, to represent credentials. RT_0 credentials make assertions by defining role memberships. Memberships can be defined in four ways, corresponding four kinds of RT_0 credentials. Since an NFA reads a string from left to right in convention, we reverse the arrow direction in RT.

Type-1: $D \rightarrow A.r$, where A and D are principals and r is a role name. A issues this credential to assert that D is a member of $A.r$. $A.r$ is called a *normal role*.

Type-2: $B.r' \rightarrow A.r$, where A and B are principals and r and r' are role names. By issuing this credential, A asserts that a member of $B.r'$ is also a member of $A.r$.

Type-3: $A.r_1.r_2 \rightarrow A.r$, where A is a principal and r_1 and r_2 are role names. This credential means that a principal is a member of $A.r$ if it is a member of $B.r_2$ for any principal B who is a member of $A.r_1$. $A.r_1.r_2$ is called a *linked role*.

Type-4: $B_1.r_1 \cap \cdots \cap B_n.r_n \rightarrow A.r$, where A and B_i are principals, and r and r_i are role names. The credential asserts that a principal is a member of $A.r$ if it is a member of $B_i.r_i$ for all $i \in [1..n]$. $B_1.r_1 \cap \cdots \cap B_n.r_n$ is called an *intersection role*.

We recall some notions of RT [12]. We say a credential of the form $e \rightarrow A.r$ *defines* the role $A.r$, where e can be D, $B.r'$, $A.r_1.r_2$, or $B_1.r_1 \cap \cdots \cap B_n.r_n$. Given a set $\mathcal{C}$ of RT_0 credentials, let $Prin(\mathcal{C})$ be the set of principals in $\mathcal{C}$; define $NRole(\mathcal{C})$ as the set of normal roles $A.r$ such that A is a principal in $\mathcal{C}$ and r is a role name in $\mathcal{C}$. Let $LRole(\mathcal{C})$ be the set of linked roles in $\mathcal{C}$ and $IRole(\mathcal{C})$ be the set of intersection roles in $\mathcal{C}$. Further, let $Role(\mathcal{C}) = NRole(\mathcal{C}) \cup LRole(\mathcal{C})$ be the set of roles in $\mathcal{C}$, containing normal and linked roles; let $\mathcal{E}(\mathcal{C}) = Prin(\mathcal{C}) \cup Role(\mathcal{C}) \cup IRole(\mathcal{C})$ be the set of role expressions in $\mathcal{C}$, containing principals, roles, and intersection roles.

Definition 1. *[12] The semantics of $\mathcal{C}$ is given by the least $\twoheadrightarrow_{\mathcal{C}} \subseteq \mathcal{E}(\mathcal{C}) \times \mathcal{E}(\mathcal{C})$ such that (1) for any $e \in \mathcal{E}(\mathcal{C})$, $e \twoheadrightarrow_{\mathcal{C}} e$, (2) if $e_1 \twoheadrightarrow_{\mathcal{C}} e_2$ and $e_2 \twoheadrightarrow_{\mathcal{C}} e_3$ then $e_1 \twoheadrightarrow_{\mathcal{C}} e_3$, (3) for any credential $e \rightarrow A.r$ in $\mathcal{C}$, $e \twoheadrightarrow_{\mathcal{C}} A.r$, (4) if $A.r_1.r_2 \in \mathcal{E}(\mathcal{C})$ and $B \twoheadrightarrow_{\mathcal{C}} A.r_1$, then $B.r_2 \twoheadrightarrow_{\mathcal{C}} A.r_1.r_2$, and (5) if $B_1.r_1 \cap \cdots \cap B_n.r_n \in \mathcal{E}(\mathcal{C})$ and $e \twoheadrightarrow_{\mathcal{C}} B_i.r_i$ for $i \in [1..n]$, then $e \twoheadrightarrow_{\mathcal{C}} B_1.r_1 \cap \cdots \cap B_n.r_n$. We write $\twoheadrightarrow$ if $\mathcal{C}$ is clear from the context.*

Example 1. Suppose that a parking lot provides parking services to staff of its partners (e.g., a medical center). The lot provides special service for people with disability. We let $\mathcal{C}_{lot}$ be the credential context consisting of the following credentials.

$c_1 : Bob \rightarrow Med.staff$ $c_2 : Med \rightarrow Lot.partner$ $c_3 : Lot.partner.staff \rightarrow Lot.pk$
$c_4 : Bob \rightarrow HR.dis$ $c_5 : HR.dis \rightarrow Med.dis$ $c_6 : Med.dis \rightarrow Lot.dis$
$c_7 : Lot.pk \cap Lot.dis \rightarrow Lot.spk$

We have $Bob \twoheadrightarrow_{\mathcal{C}_{lot}} Lot.pk$ and $Bob \twoheadrightarrow_{\mathcal{C}_{lot}} Lot.spk$. □

RT_1 credentials allow the use of parameterized roles. For ease of exposition, we focus mainly on RT_0 credentials.

3 Related Work

Bauer et al. [3] present an approach to constraining credential usage for *proof-carrying authorization (PCA)*. It has a heavy influence on this work: we follow the idea to place constraints on proofs. Nevertheless, we observe some essential differences between

PCA and TN; these differences dissuade us from adapting their approach to TN. First, PCA does not search for proofs but only verifies proofs submitted by access-requester. This makes it possible to define constraints as arbitrary functions over proofs. In contrast, a compliance checker of TN ought to search for proofs by itself. It appears difficult for an efficient search algorithm to work with arbitrary functions. This consideration leads to our definition of constraints as NFA properties. It seems restrictive to define constraints as NFA properties, instead of arbitrary functions over proofs; but, as it turns out, most example constraints in [3] could be expressed. Second, compliance checkers are expected to find all proofs of a conclusion. Even though some mechanisms [2] assist principals in composing proofs in PCA, they only find one proof and thus provide inadequate help for TN. Finally, PCA policy languages are seldom used in TN.

Lee and Winslett present an efficient type-3 compliance checker CLOUSEAU [6]. CLOUSEAU compiles credentials and policies into an intermediate representation that is analyzed using efficient pattern matching algorithms. Another type-3 compliance checker SSgen is proposed by Smith et al. [11]. Given an input, SSgen iteratively invokes a type-2 checker, which generates only one satisfying set with the input; SSgen feeds the type-2 checker with modified input so that previously generated proofs are excluded and an alternative proof can be found. It remains unclear, however, how to extend existing type-3 checkers like CLOUSEAU and SSgen to deal with constraints on credential usage.

To restrict credential usage, one may attempt to employ a type-3 checker to generate satisfying sets and remove the sets that violate usage-constraints; so the remaining sets respect the constraints. Here, the constraints can be arbitrary functions over proofs, as one only needs to verify proofs against them. We take a closer look at this approach with CLOUSEAU as the checker.

First, CLOUSEAU has to supply all satisfying sets but not only the minimal ones. For a counter-example, suppose that $C_1 = \{c_1, c_2, c_3, c_4, c_5\}$ is a satisfying set and $C_2 = \{c_1, c_2\}$ a minimal one. Suppose further that a constraint requires that the use of c_2 be accompanied by the use of c_3.[2] In this case, C_1 is a set conforming to the constraint, but C_2 is not. In the worst case, the number of minimal satisfying sets for a given policy can be exponential [6,11], not to speak of the number of all satisfying sets. It is challenging to efficiently verify the sets against constraints, in spite of the efficient algorithms used in CLOUSEAU.

Second, CLOUSEAU returns credential sets, instead of proofs. From a credential set, however, more than one proof might be constructed. For example, one can write two proofs using credentials in C_1, as mentioned earlier. In this case, it is ambiguous which proof a constraint concerns with.

Finally, this approach lacks an intuitive way to capture constraints over attributes required of credential issuers [3]. For example, Alice may say that her credentials can only be used in a proof involving credentials from herself or her friends. We are not aware of any simple extensions to CLOUSEAU which are able to enforce this constraint.

[2] Suppose that the given policy is $A \twoheadrightarrow C.r$ and that the credentials are $c_1 : A \rightarrow B.r$, $c_2 : B.r \rightarrow C.r$, $c_3 : C.r.r' \rightarrow C.r$, $c_4 : D \rightarrow B.r$, and $c_5 : A \rightarrow D.r'$. One can compose two distinct proofs using credentials $\{c_1, c_2\}$ and $\{c_1, c_2, c_3, c_4, c_5\}$, respectively. Note that the second proof relies on c_3 rather than just spuriously include it.

One might consider specifying credentials in more complicated languages. For example, besides RT_0 and RT_1, the RT family includes more expressive languages such as RT^T. Similarly one might redesign policy languages so that principals can specify in more detail how their assertions and judgements are to be used. This could entail the introduction of new modalities [4] and of complex inference rules [3]. This approach is less attractive for the following reasons. First, apart from the compliance checker, it may result in changes to other TN components. Since a TN system uses its policy language to represent credentials and release policies [9], this representation might need revisions. Second, one has to design a new compliance checker that is able to efficiently find all minimal satisfying sets in the presence of the new, more complicated language. Finally, in this manner, only a set of predefined constraints can be enforced [3].

To sum up, based on these these observations, an alternative is arguably worth investigating. The challenge is to support constraints that capture practical requirements while enabling efficient compliance checking. As such, we propose a novel definition of constraints based on NFAs and employ ASP to enforce constraints.

Note that we do not concern ourselves with the information leakage and hiding problem in TN; instead, we simply focus on how to find all minimal proofs which use credentials in a way consistent with constraints.

4 Usage-Constraints on Credentials

To support usage-constraints on credentials, we follow an idea similar to the one in [3]: allow credential issuers to specify constraints on proofs where their credentials are to be used. A proof is decomposed into a set of strings. Constraints are defined based on NFAs; a proof is required to be accepted by NFAs, either partially or entirely as specified. A compliance checker is designed to return all such minimal proofs.

4.1 Defining Proofs

Suppose $e_1 \twoheadrightarrow_{\mathcal{C}} e_2$; there is a proof justifying this statement. The proof can take various forms. For example, Fig. 1 shows a proof tree of $Bob \twoheadrightarrow_{\mathcal{C}_{lot}} Lot.spk$. A node of a proof tree is a pair of principal-role, which means the principal becomes a member of the role. A proof like this hinders a simple definition of constraints. First, it could

$$\begin{tikzcd}
 & Bob : Lot.spk & & \\
Bob : Lot.dis \arrow[ur] & & Bob : Lot.pk \arrow[ul] & \\
Bob : Med.dis \arrow[u] & & Bob : Lot.partner.staff \arrow[u] & \\
Bob : HR.dis \arrow[u] & & Bob : Med.staff \arrow[u] & Med : Lot.partner \arrow[ul, "*"'] \\
Bob : Bob \arrow[u] & & Bob : Bob \arrow[u] & Med : Med \arrow[u]
\end{tikzcd}$$

Fig. 1. A proof tree of $Lot.spk \twoheadrightarrow_{\mathcal{C}_{lot}} Bob$. The star edge denotes a linked membership derived from the outgoing node of the edge.

contain redundant reasoning. For example, the tree in Fig. 1 also serves as a proof of $Bob \twoheadrightarrow_{\mathcal{C}_{lot}} Lot.pk$; in this case, it includes superfluous applications of credentials. Second, placing constraints on trees raises the bar for credential issuers to correctly specify constraints.

A slight revision will provide a simpler solution but no less capability. In brief, we borrow an idea of model checking linear temporal logic properties [5]. We obtain a set of sequences of principal-role pairs, when a tree is traversed: a traversal starts from a leaf node and proceeds until it encounters a star edge or the root node. Consider again the tree in Fig. 1; we have the following sequences.

$[Bob : Bob, Bob : HR.dis, Bob : Med.dis, Bob : Lot.dis, Bob : Lot.spk]$
$[Bob : Bob, Bob : Med.staff, Bob : Lot.partner.staff, Bob : Lot.pk, Bob : Lot.spk]$
$[Med : Med, Med : Lot.partner]$

Observe that each sequence concerns only one principal; it could be shortened as a *role path* defined below.

Definition 2. *A* role path *in $\mathcal{C}$ is a tuple $[e_0, e_1, \cdots, e_n]$ where (1) $e_0 \in Prin(\mathcal{C})$, (2) $\{e_1, \cdots, e_n\} \subseteq Role(\mathcal{C})$, and (3) if $e_i = e_j$ then $i = j$.*

Condition 1 says that a role path begins with a principal. Condition 2 says that a principal is followed by a sequence of roles. Condition 3 says there is no cycle in a role path.[3] For example, we have the following role paths in $\mathcal{C}_{lot}$; there, h_1, h_2, and h_3 correspond to the tree in Fig. 1.

$h_1 : [Bob, HR.dis, Med.dis, Lot.dis, Lot.spk]$
$h_2 : [Bob, Med.staff, Lot.partner.staff, Lot.pk, Lot.spk]$
$h_3 : [Med, Lot.partner]$
$h_4 : [Bob, Med.staff, Lot.partner.staff, Lot.pk]$

Definition 3. *Let $\mathcal{H}$ be the set of role paths in $\mathcal{C}$. We define a relation $\vdash \subseteq 2^{\mathcal{H}} \times 2^{\mathcal{C}} \times \twoheadrightarrow$; for any $(H, C, e_1 \twoheadrightarrow e_2) \in \vdash$, we write $H \vdash_C e_1 \twoheadrightarrow e_2$. For any $H \subseteq \mathcal{H}$ and $C \subseteq \mathcal{C}$, the followings hold:*

1. *If $e \in h$ for some $h \in H$, then $H \vdash_C e \twoheadrightarrow e$.*
2. *If $H \vdash_C e_1 \twoheadrightarrow e_2$ and $H \vdash_C e_2 \twoheadrightarrow e_3$, then $H \vdash_C e_1 \twoheadrightarrow e_3$.*
3. *For any $[\cdots, e_i, e_{i+1}, \cdots] \in H$, if $e_i \rightarrow e_{i+1} \in C$ then $H \vdash_C e_i \twoheadrightarrow e_{i+1}$.*
4. *If $H \vdash_C B \twoheadrightarrow A.r_1$, then for any $[\cdots, B.r_2, A.r_1.r_2, \cdots] \in H$, $H \vdash_C B.r_2 \twoheadrightarrow A.r_1.r_2$.*
5. *For any $B_1.r_1 \cap \cdots \cap B_n.r_n \rightarrow A.r \in C$, if for $i \in [1..n]$ both $[\cdots, e, B_i.r_i, A.r, \cdots] \in H$ and $H \vdash_C e \twoheadrightarrow B_i.r_i$ hold, then $H \vdash_C e \twoheadrightarrow A.r$.*

Example 2. Continue with Example 1. Consider the sets $H_{pk} = \{h_3, h_4\}$ and $C = \{c_1, c_2, c_3\} \subset \mathcal{C}_{lot}$. One can derive $H_{pk} \vdash_C Bob \twoheadrightarrow Lot.pk$. H_{pk} shows how $Bob \twoheadrightarrow Lot.pk$ is concluded using credentials in C: From h_4, Bob first becomes a $Med.staff$; from h_3, Med is a $Lot.partner$; continuing with h_4, Bob turns into a $Lot.partner.staff$ because of Med's membership in $Lot.partner$, and finally becomes a $Lot.pk$.

If we let $H_{spk} = \{h_1, h_2, h_3\}$ and $C = \mathcal{C}_{lot}$, we have $H_{spk} \vdash_C Bob \twoheadrightarrow Lot.spk$. H_{spk} explains $Bob \twoheadrightarrow Lot.spk$ in the same way as the proof tree in Fig. 1 does. Path h_2

[3] This is not to be confused with policy cycles [7].

seems to indicate that Bob gains $Lot.spk$ simply because of his membership in $Lot.pk$, which is granted to any $Lot.partner.staff$ without the requirement of a membership in $Lot.dis$. As indicated by the fact that $\{h_2, h_3\} \vdash_C Bob \twoheadrightarrow Lot.spk$ does not hold, however, this understanding is not correct. Actually, the intuition here is that Bob's membership in $Lot.spk$ is preceded by his membership in $Lot.pk$. □

We write $H_1 \leq H_2$ if $H_1 \subseteq H_2$ or for all $h \in H_1$, there exists $h' \in H_2$ such that h is a prefix of h',[4] and $H_1 < H_2$ if $H_1 \leq H_2$ and $H_1 \neq H_2$.

Definition 4 (Proof). *We say H is a* proof *of $e_1 \twoheadrightarrow e_2$* using *$C$ if $H \vdash_C e_1 \twoheadrightarrow e_2$ and for all $H' < H$ it does not hold that $H' \vdash_C e_1 \twoheadrightarrow e_2$.*

For example, $\{h_3, h_4\}$ is a proof of $Bob \twoheadrightarrow Lot.pk$ using $\{c_1, c_2, c_3\}$.

4.2 Defining Constraints

Semantically, a constraint defines a set of allowable proofs. A proof, in turn, is a set of role paths. Further, a role path, when viewed as a string, is accepted or denied by an NFA (non-deterministic finite automaton). Hence, credential issuers could design NFAs to define allowable proofs.

Definition 5. *[10] An NFA N is a tuple $(S, \Sigma, \delta, s_0, F)$, where S is a finite set of states, Σ is a finite alphabet, $\delta : S \times (\Sigma \cup \{\epsilon\}) \mapsto 2^S$ is the transition function, $s_0 \in S$ is the start state, and $F \subseteq S$ is the set of accept states. For $\Sigma' \subseteq \Sigma$, we write $\delta(s_1, \Sigma') = s_2$ as a shorthand for the set $\{\delta(s_1, v) = s_2 \mid v \in \Sigma'\}$ for any $s_1, s_2 \in S$.*

Let w be a string over the alphabet Σ; we say N accepts *w if we can write w as $v_1 v_2 \cdots v_m$, where each v_i is a member of $\Sigma \cup \{\epsilon\}$ and a sequence of states $q_0, q_1, \cdots, q_m$ exists in S with three conditions: (1) $q_0 = s_0$, (2) $q_{i+1} \in \delta(q_i, v_{i+1})$ for $i \in [0..m-1]$, and (3) $q_m \in F$. Let $L(N)$ be the set of strings that N accepts.*

Example 3 (Final-usage constraint). Continue with Example 1. Recall that the medical center issues the credential $HR.dis \rightarrow Med.dis$ defining the role $Med.dis$. Suppose that the center confines the usage of the memberships of $Med.dis$ to limited purposes. For example, it can be used in proving entitlement to special parking service; rather, it cannot be used in any commercial promotions where, for instance, people with disability are given coupons. To this end, the center requires that each role path of a proof H be accepted by the NFA N_{lot} in Fig. 2. Consider a path $[e_0, e_1, \cdots, e_n] \in H$. N_{lot} says that e_0 must be a principal such as Bob. Next, if N_{lot} does not confront $Med.dis$ (i.e., no credential defining $Med.dis$ is used) all the way through the path, it will accept the path. Suppose otherwise that $e_i = Med.dis$ (i.e, a credential of the form $e_{i-1} \rightarrow Med.dis$ is used); N_{lot} accepts the path if it ends with $Lot.spk$. Consequently, H is a proof where credentials affirming disability by Med is only used for special parking service if and only if all role paths in H are accepted by N_{lot}. □

One may have noticed that N_{lot} works only in the context of $\mathcal{C}_{lot}$. It is unlikely for the center to specify a similar NFA for every situation where a credential defining $Med.dis$

[4] A path $[e_0, \cdots, e_n]$ is a prefix of a path $[e_0, \cdots, e_n, \cdots, e_{n+m}]$ where $m \geq 0$.

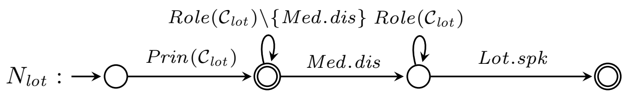

N_{lot} :
$Prin(\mathcal{C}_{lot})$
$Role(\mathcal{C}_{lot})\backslash\{Med.dis\}$
$Med.dis$
$Role(\mathcal{C}_{lot})$
$Lot.spk$

Fig. 2. An NFA N_{lot} that restricts the usage of credentials defining $Med.dis$ to special parking service in the context of $\mathcal{C}_{lot}$

may be used. Instead, the center is more willing to write a special NFA with a credential context placeholder. In event of compliance checking, as the credential context is fixed, a context like $\mathcal{C}_{lot}$ is substituted for the placeholder.

Definition 6. *A* context-dependent NFA*, denoted as γ, is an NFA $(S, \Sigma, \delta, s_0, F)$ where the alphabet Σ consists of the following:*

1. *a finite set* `Prin` *of principals and a finite set* `Role` *of roles, and*
2. *a set of special symbols $\{Prin(\underline{\mathcal{C}}), Role(\underline{\mathcal{C}})\} \cup \{Role(\underline{\mathcal{C}})\backslash A \mid A \subset \texttt{Role}\}$, where $\underline{\mathcal{C}}$ denotes the credential context placeholder.*

The set `Role` contains roles that a principal is aware of, when designing a context-dependent NFA; likewise, `Prin` contains principals. Take Example 3 for instance; the center knows the roles $Med.dis$ and $Lot.spk$, for it means to restrict the usage of credentials like $HR.dis \rightarrow Med.dis$ to the special parking service that is represented by $Lot.spk$. In this case, $\texttt{Prin} = \{Med, Lot\}$ and $\texttt{Role} = \{Med.dis, Lot.spk\}$.

A special symbol is treated as a single unit. Consider for example the NFA γ_{lot} in Fig. 3; it accepts the following string.

$$Prin(\underline{\mathcal{C}})\ Role(\underline{\mathcal{C}})\backslash\{Med.dis\}\ Med.dis\ Role(\underline{\mathcal{C}})\ Lot.spk \tag{1}$$

A special symbol turns into a set expression after a credential context $\mathcal{C}$ is substituted for the placeholder. For example, if $\mathcal{C}_{lot}$ takes place of $\underline{\mathcal{C}}$, $Role(\underline{\mathcal{C}})$ turns into $Role(\mathcal{C}_{lot})$, which is the set of roles in $\mathcal{C}_{lot}$. Supposing that v is a special symbol, we write the set expression obtained from the substitution as $v|_{\mathcal{C}}$.

Definition 7. *Given a role path $h = [e_0, \cdots, e_n]$ in a context $\mathcal{C}$, we say $(\gamma, \mathcal{C})$* allows *h if γ accepts a string $v_0 \cdots v_n$ such that for $i \in [0..n]$, either $v_i = e_i$ or v_i is a special symbol and $e_i \in v_i|_{\mathcal{C}}$. Denote the set of paths allowed by $(\gamma, \mathcal{C})$ as $L(\gamma, \mathcal{C})$.*

For instance, $(\gamma_{lot}, \mathcal{C}_{lot})$ allows the path $[Bob, HR.dis, Med.dis, Lot.dis, Lot.spk]$, for γ_{lot} accepts the string in (1). The center can specify the context-dependent NFA γ_{lot} in Fig. 3 so as to restrict the usage of credentials defining $Med.dis$ to special parking service in any credential context. As discussed in Example 3, those credentials will not be used for other purposes as long as role paths are allowed by $(\gamma_{lot}, \mathcal{C})$.

We notice that $L(\gamma, \mathcal{C})$ is still a regular language, which can be directly captured by an NFA. For example, N_{lot} in Fig. 2 recognizes $L(\gamma_{lot}, \mathcal{C}_{lot})$ (i.e., $L(N_{lot}) = L(\gamma_{lot}, \mathcal{C}_{lot})$). When designing a context-dependent NFA, credential issuers could first specify an NFA for a specific context, and abstract it away later. Throughout the rest of this paper, unless otherwise stated, references to an NFA imply a context-dependent NFA. When $\mathcal{C}$ is clear from the context, we say γ allows a role path instead of $(\gamma, \mathcal{C})$.

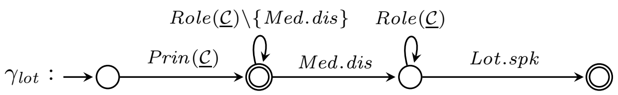

γ_{lot} :
$Prin(\underline{\mathcal{C}})$
$Role(\underline{\mathcal{C}})\backslash\{Med.dis\}$
$Med.dis$
$Role(\underline{\mathcal{C}})$
$Lot.spk$

Fig. 3. A context-dependent NFA γ_{lot} restricting the usage of credentials defining $Med.dis$ to special parking service

Definition 8 (Usage-constraint). *A* credential constraint *is a tuple* $\langle\gamma, \pi\rangle$ *where* γ *is an NFA and* $\pi \in \{\forall, \exists\}$*. We say a proof* H *in* $\mathcal{C}$ *is* valid with respect to (wrt) $\langle\gamma, \pi\rangle$ *if* $H \subseteq L(\gamma, \mathcal{C})$ *when* $\pi = \forall$ *and* $H \cap L(\gamma, \mathcal{C}) \neq \emptyset$ *when* $\pi = \exists$*. A constraint* $\langle\gamma, \forall\rangle$ *requires that each role path of* H *be allowed by* $(\gamma, \mathcal{C})$*. A constraint* $\langle\gamma, \exists\rangle$ *says there is at least one role path of* H *allowed by* $(\gamma, \mathcal{C})$*.*

Example 4 (Delegation depth). Recall that the parking lot delegates the judgement of disability to the center by credential $c_6 : Med.dis \rightarrow Lot.dis$. No restriction is placed on the delegation; the center could re-delegate the judgement to any principal. The lot may deem this over-permissive and want to control the delegation. For example, the lot might decide that a principal's disability should be asserted directly by the center; namely, it allows no re-delegation from the center. As such, the lot designs an NFA γ_{d0} in Fig. 4 and put a constraint $\langle\gamma_{d0}, \forall\rangle$. When designing γ_{d0}, the lot is aware of credential c_6 and thus of the sets $\texttt{Prin} = \{Lot, Med\}$ and $\texttt{Role} = \{Lot.dis, Med.dis\}$.

We now examine the semantics of $\langle\gamma_{d0}, \forall\rangle$. Consider for example the credential context $\mathcal{C}_{lot}$. According to the constraint, all role paths of a valid proof wrt it should be allowed by γ_{d0}. A path is allowed if it meets one of the conditions: (1) $Lot.dis$ does not show up and (2) $Lot.dis$ is preceded by $Med.dis$, which in turn is preceded by a principal. In the latter case, $Med.dis$ follows immediately a principal; this indicates the use of a credential $e \rightarrow Med.dis$, where e is a principal. Hence, a proof involving $Lot.dis$ uses no delegation of $Med.dis$ if and only if it is valid wrt the constraint. Consider a proof H_{depth} containing a path $[Bob, HR.dis, Med.dis, Lot.dis, Lot.spk]$. Since γ_{d0} does not allow this path, the proof H_{depth} is not valid. On the other hand, H_{depth} does use a delegation that the lot tries to prevent (e.g., a credential $HR.dis \rightarrow Med.dis$).

Suppose that the lot now relaxes its requirement: it permits the center to re-delegate to another principal, but disallows any further delegation. This time the lot designs an NFA γ_{d1}, as shown in Fig. 4. In comparison with γ_{d0}, γ_{d1} permits an optional role between a principal and $Med.dis$; this models a possible one-step re-delegation. □

One can proceed to define more constraints using logical connectives as below and define their semantics as in the propositional logic.

$$con ::= \langle\gamma, \forall\rangle \mid \langle\gamma, \exists\rangle \mid (\neg con) \mid (con \wedge con) \mid (con \vee con) \mid (con \Rightarrow con)$$

We could also define a constraint like $\langle\neg\gamma_1 \vee \gamma_2, \forall\rangle$ so that a path is allowed by γ_2 if allowed by γ_1. Since regular languages are closed under the operations union, intersection, difference, and complement, there is a constraint $\langle\gamma, \forall\rangle$ to the same effect.

Definition 9. *Given a credential context* $\mathcal{C}$*, a set* Γ *of credential constraints, and a goal* $D \twoheadrightarrow A.r$*, we say* $C \subseteq \mathcal{C}$ *is a* proving set *of* $e_1 \twoheadrightarrow e_2$ *if for all* $C' \subset C$ *there is no*

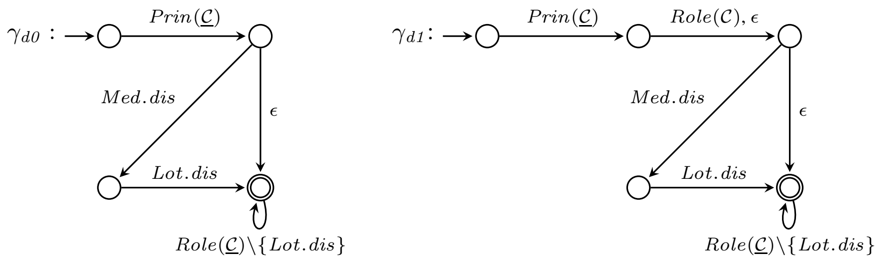

γ_{d0} :
$Prin(\underline{\mathcal{C}})$
$Med.dis$
ϵ
$Lot.dis$
$Role(\underline{\mathcal{C}})\backslash\{Lot.dis\}$
γ_{d1}:
$Prin(\underline{\mathcal{C}})$
$Role(\mathcal{C}), \epsilon$
$Med.dis$
ϵ
$Lot.dis$
$Role(\underline{\mathcal{C}})\backslash\{Lot.dis\}$

Fig. 4. NFA γ_{d0} forbids re-delegation. NFA γ_{d1} allows an optional one-step re-delegation.

proof H' of $e_1 \twoheadrightarrow e_2$ using C' such that H' is valid wrt all constraints in Γ. Type-3 compliance checking problem in the presence of constraints is to find all the proving sets $C_1, \ldots, C_n$. We denote this problem as $\langle \mathcal{C}, \Gamma, D \twoheadrightarrow A.r \rangle$.

5 ASP Representation

To solve the type-3 compliance checking problem in the presence of constraints, we encode it in ASP. The use of ASP is motivated by, among others, its ability to return all solutions to a problem. Intuitively, we view RT_0 credentials as *actions* of adding principals to roles' member sets. For example, the application of a credential $D \rightarrow A.r$ adds the principal D to $A.r$'s member set; another credential $A.r \rightarrow B.r'$, if applied, further makes D a member of $B.r'$. Therefore, a credential context is considered as a set of actions that may be executed to grant principals role-memberships. Now, to decide if a principal is a member of a role is to decide if there exists an action plan which ultimately adds the principal to the role's member set. The ASP encoding is parameterized on proof size. Given a proof H, define its size as $size(H) = \sum_{h \in H} size(h)$, where, assuming $h = [e_0, \cdots, e_n]$, $size(h) = n$. Given $\langle \mathcal{C}, \Gamma, D \twoheadrightarrow A.r \rangle$ and a proof size parameter k, for any answer set that the ASP program returns, it corresponds to a proving set with a proof H such that $size(H) < k$; on the other hand, for any proving set C with a proof H such that $size(H) < k$, then the program returns an answer set corresponding to C.

6 Experimental Results

In this section, we evaluate the performance of our compliance checker and the overheads resulting from the support of usage-constraints on credentials. Our concerns lie mainly in the computing time required to find all proving sets for a given policy. Experiments were carried out on a Windows 7 laptop with Intel Core 2.66GHz i5-560M processor and 4GB RAM. ASP programs were executed with the grounder gringo 3.0.3 and the solver clasp 2.0.3.[5] In each test, the compliance checking is performed in a credential context of 50 RT_0 credentials, i.e., $|\mathcal{C}| = 50$. All results for a specific parameter setting were averaged over 5 independent tests.

[5] http://sourceforge.net/projects/potassco/

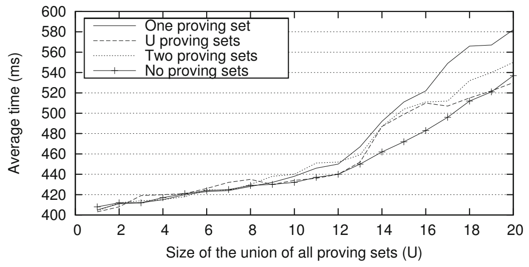


Fig. 5. The computing time as a function of the size of the union of all proving sets

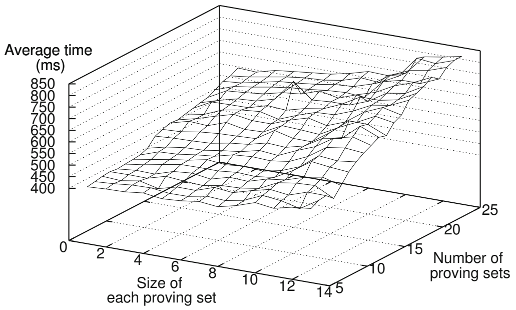


Fig. 6. Computing time as a function of the number of proving sets and the size of each set

We conducted three groups of experiments. First, we evaluated the checker's performance with respect to the size of proving sets. We considered three cases: (i) the policy had one proving set of size U, (ii) the policy had U proving sets of size one, (iii) the policy had two proving sets of size $\frac{3U}{4}$, where U is the size of the union of all proving sets. These are the most interesting cases explored in literature [6,11]. We placed on credentials 20 manually created constraints of types in [3]; these constraints concerned at least 80% of the credentials in $\mathcal{C}$. We set k as 25; namely we only searched for proofs of size smaller than 25. Fig. 5 shows the results. In all cases, it took the checker less than 600 ms to find all proving sets. Besides, we also tested the cases when no proving sets existed. To perform such tests, when we obtained the single proving set in cases of (i), we included one more constraint to rule it out so that no such set existed. In all four cases, the running time grew linearly.

Further we examined the computing time as a function of the number of proving sets and the size of each proving set. Again we put 20 manually created constraints and set k, the limit of proof size, as 25. Fig. 6 shows the results, which confirm that the computing time grew linearly with respect to the number and the size of proving sets.

Similar experiments were conducted on CLOUSEAU in [6]. Although our checker is several times slower than CLOUSEAU, it responded within 850 ms in all previous tests.

Second, we examined the overheads incurred by supporting constraints; we varied the number of constraints $|\Gamma|$. In this experiment, we set $k = 25$. Initially, we had $|\Gamma| = 0$ and 10 proving sets each of size 15. Later, we incrementally added constraints.

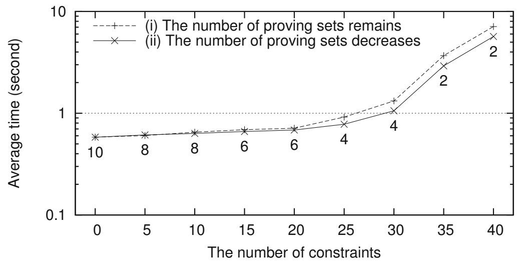


Fig. 7. Computing time as a function of the number of constraints. Below the data points of case (ii) is labelled the number of proving sets in each case of $|\Gamma|$.

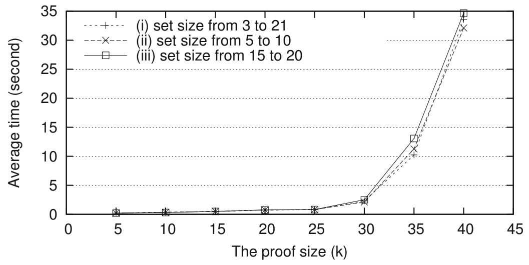


Fig. 8. Computing time as a function of the proof size parameter k

We considered two cases: (i) All proving sets remained, i.e., constraints did not invalidate any of them; (ii) some sets were not valid wrt some added constraints. Recall that a proof should be valid wrt every constraint in Γ; hence, the ASP program encoded all constraints in Γ. Fig. 7 shows the results. In both cases, the checker performed well before $|\Gamma|$ grew to 30 and turned impractical afterwards. Comparing the times for $|\Gamma| = 0$ and $0 < |\Gamma| < 30$, we note that constraints incurred overheads less than 1 second.

Finally, we evaluated the influence of proof size on performance by varying the parameter k. In this experiment, we had 20 constraints and 10 proving sets. We considered three cases according to the size of proving sets: (i) the size ranged from 3 to 21, (ii) the size ranged from 5 to 10, and (iii) the size ranged from 15 to 20. Fig. 8 shows the results. We see that k had a heavy influence on performance. The indistinguishable time difference between the three cases also implies that k played a major role in performance. The checker responded within 3 seconds when $k = 30$; but the performance degenerated rapidly as k grew. This is in accordance with our ASP encoding, which is parametric to k. We observed from practical proofs that their size seldom exceeds 30.

7 Conclusions

In this paper, we presented a definition of usage-constraints on credentials based on NFAs (non-deterministic finite automaton). We illustrated by examples that the definition is able to express important constraints in practice. Based on an encoding in ASP (answer set programming), we proposed a compliance checker that is able to find all minimal sets of credentials for a given policy; each such set not only constitutes a proof of the policy but also uses the credentials in a way consistent with given constraints. Experiment results showed the efficiency of our approach.

We assumed that compliance checking is performed in a localized credential context. In practice, however, the context may be distributed. In that case, the problem is more challenging. For one thing, constraints may be stored with credentials and thus be distributed too. For another, we still need to search for multiple, if not all, proving sets for a given policy. We plan to study the compliance checking problem under distributed credential contexts in future work.

Acknowledgment. Khaled M. Khan, Yan Zhang and Yun Bai are supported by an NPRP grant (NPRP 09-079-1-013) from the Qatar National Research Fund (QNRF). Jinwei Hu is supported by CASED (www.cased.de). The statements made herein are solely the responsibility of the authors. We also thank Dieter Gollmann for shepherding the paper and the anonymous reviewers for their helpful comments.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Bauer, L., Garriss, S., Reiter, M.K.: Distributed proving in access-control systems. In: IEEE Symposium on Security and Privacy, pp. 81–95 (2005)
3. Bauer, L., Jia, L., Sharma, D.: Constraining credential usage in logic-based access control. In: CSF, pp. 154–168 (2010)
4. Becker, M.Y.: Information flow in credential systems. In: CSF, pp. 171–185 (2010)
5. Huth, M., Ryan, M.: Logic in Computer Science: modelling and reasoning about systems. Cambridge University Press (2004)
6. Lee, A.J., Winslett, M.: Towards an efficient and language-agnostic compliance checker for trust negotiation systems. In: ASIACCS, pp. 228–239 (2008)
7. Li, J., Li, N., Winsborough, W.H.: Automated trust negotiation using cryptographic credentials. ACM Trans. Inf. Syst. Secur. 13(1) (2009)
8. Li, N., Mitchell, J.C., Winsborough, W.H.: Design of a role-based trust-management framework. In: IEEE Symposium on Security and Privacy, pp. 114–130 (2002)
9. Seamons, K.E., Winslett, M., Yu, T., Smith, B., Child, E., Jacobson, J., Mills, H., Yu, L.: Requirements for policy languages for trust negotiation. In: POLICY, pp. 68–79 (2002)
10. Sipser, M.: Introduction to the Theory of Computation (2005)
11. Smith, B., Seamons, K.E., Jones, M.D.: Responding to policies at runtime in trustbuilder. In: POLICY, pp. 149–158 (2004)
12. Winsborough, W.H., Li, N.: Towards practical automated trust negotiation. In: POLICY, pp. 92–103 (2002)

A ASP Encoding

Since uppercase letters are usually taken as variables in ASP and lowercase letters as constants, we use lowercase letters to denote principals in ASP programs. We translate a compliance checking problem $\langle \mathcal{C}, \Gamma, d \twoheadrightarrow a.r\rangle$ into an ASP program, denoted as $\Pi(\langle \mathcal{C}, \Gamma, d \twoheadrightarrow a.r\rangle, k)$. An ASP program is a finite set of rules of the form

$$a \text{ :- } b_1, \cdots, b_m, \texttt{not}\, b_{m+1}, \cdots, \texttt{not}\, b_n.$$

where $0 \le m \le n$, a is either an atom or $\bot$, b_i is an atom, and `not` denotes (default) negation. We say that a rule is a *fact* if $n = 0$. For simplicity, we omit :- when writing facts. Readers are referred to [1] for details.

$\Pi(\langle \mathcal{C}, \Gamma, d \twoheadrightarrow a.r\rangle, k)$ consists of four parts: $\Pi(\mathcal{C})$, $\Pi(apply)$, $\Pi(d \twoheadrightarrow a.r, k)$, and $\Pi(\Gamma)$. Except $\Pi(apply)$, the sub-programs are parametric to problems. Program $\Pi(\mathcal{C})$ models each credential in $\mathcal{C}$. Program $\Pi(apply)$ states how credentials are applied. Program $\Pi(d \twoheadrightarrow a.r, k)$ tests whether a proof of size smaller than k exists. Program $\Pi(\Gamma)$ interprets the constraints in Γ and ensures that proofs extracted from answer sets are valid wrt the constraints. Program $\Pi(\mathcal{C})$ follows the original semantics of RT [8]; we omit its presentation here.

We first present $\Pi(d \twoheadrightarrow a.r, k)$. Fact (2) below declares our goal as proving that principal d can obtain a membership in $a.r$. Rule (3) says that we prove the goal at step T if the goal holds at T. Rule (4) says that once the goal is proved at T it is proved afterwards. Rule (5) requires that an answer set should contain the atom $prove(k)$ (i.e., the membership is proved within k steps).

$$goal(m(d, r(a.r))). \tag{2}$$

$$prove(T) \text{ :- } hold(M, T), goal(M), step(T). \tag{3}$$

$$prove(T+1) \text{ :- } step(T), T < k, prove(T). \tag{4}$$

$$\bot \text{ :- } \texttt{not}\, prove(k). \tag{5}$$

A term $m(d, r(a.r))$ denotes d's membership in $a.r$. Expression $r(a.r)$ is short for a term $r(a, r, \rho)$, where ρ is a special constant used in our ASP programs. The term $r(a, r, \rho)$ denotes a role $a.r \in NRole(\mathcal{C})$. Similar shorthands are used below.

We proceed to describe $\Pi(apply)$ (i.e., how credentials are applied), as shown in Fig. 9. At each step, some credentials are applicable while others are not. As stated in rule (6), a credential is applicable if there is a principal ready to accept the membership it offers. Rule (7) says that, a principal is ready when his memberships satisfy the credential's condition and he is not forbidden from being so. Rules (8) and (9) list the two cases of a principal being not ready: one is when the principal already owns the membership that the credential offers; the other is when another principal is ready. Rules (10) and (11) are two auxiliary rules defining the satisfiability of conditions.

At each step, one and only one membership is granted; as a result, if more than one membership can be granted at a step, only one of them succeeds and others are blocked. In addition, if the goal $d \twoheadrightarrow a.r$ has been reached, no more membership is needed. Rules (12)-(14) capture this idea. Specifically, rule (12) says that, when the goal has not yet been reached, a credential is applied if not blocked. A credential is blocked in event of another credential being applied at the same time (rule (13)) or a linked role membership being derived (rule (14)). Rule (14) also implies that linked role memberships take priority. Finally, rule (15) says that only applicable credentials may be applied; otherwise there arises a conflict.

$$
\begin{array}{rcll}
apbl(C,T) & \text{:-} & rdy(X,C,T), cred(C), step(T). & (6)\\
rdy(X,C,T) & \text{:-} & \texttt{not}\ not\_rdy(X,C,T), sat(cond(C,X),T). & (7)\\
not\_rdy(X,C,T) & \text{:-} & effect(C,m(X,R)), hold(m(X,R),T). & (8)\\
not\_rdy(X,C,T) & \text{:-} & rdy(Y,C,T), X \neq Y, prin(X), prin(Y). & (9)\\
sat(D,T) & \text{:-} & \texttt{not}\ not\_sat(D,T), in(\_,D), step(T). & (10)\\
not\_sat(D,T) & \text{:-} & in(M,D), \texttt{not}\ hold(M,T), step(T). & (11)\\
apl(C,T) & \text{:-} & cred(C), step(T), \texttt{not}\ goal(T), \texttt{not}\ blk(C,T). & (12)\\
blk(C,T) & \text{:-} & cred(C;C'), step(T), apl(C',T), C \neq C'. & (13)\\
blk(C,T) & \text{:-} & linked(\_,T), cred(C). & (14)\\
\bot & \text{:-} & cred(C), step(T), apl(C,T), \texttt{not}\ apbl(C,T). & (15)\\
obt(m(X,R'),m(X,R),T+1) & \text{:-} & rdy(X,C,T), effect(C,m(X,R)), in(m(X,R'), & \\
 & & cond(C,X)), apl(C,T), T < k-1. & (16)\\
obt(M',M,T+1) & \text{:-} & linked(M',M,T), T < k-1. & (17)\\
hold(M,T) & \text{:-} & obt(\_,M,T'), T' \leq T, step(T;T'). & (18)\\
obt(m(X,r(X)),m(X,r(X)),1) & \text{:-} & prin(X). & (19)\\
need(M) & \text{:-} & goal(M). & (20)\\
need(M) & \text{:-} & need(M'), obt(M,M',\_). & (21)\\
need(m(Y,r(Z.N))) & \text{:-} & obt(m(X,r(Y.N')),m(X,r(Z.N.N')),T), & \\
 & & need(m(X,r(Y.N'));m(X,r(Z.N.N'))). & (22)\\
\bot & \text{:-} & \texttt{not}\ need(M), obt(\_,M,T), T > 1. & (23)
\end{array}
$$

Fig. 9. Rules modelling how credentials are applied

A principal obtains a role membership when a credential is applied (rule (16)) or when a linked role membership is derived (rule (17)). A fact $obt(m_1, m_2, t)$ means that, following m_1 in a path, membership m_2 is obtained at step t; note that m_1 and m_2 are memberships of the same principal. The obtained memberships remain afterwards (rule (18)). Rule (19) declares facts that a principal d is a member of $r(d)$ at the beginning.

The rest of the encoding ensures that the proof is minimal. To retain only necessary memberships, we traverse back from the goal. As stated in rule (20), the goal is needed. Moreover, a membership next to a needed membership is necessary (rule (21)). Rule (22) considers the case of linked role memberships: if y's membership in $z.n$ helps derive x's membership in $z.n.n'$ from x's membership in $y.n'$, then y's membership in $z.n$ is necessary provided that the latter two are also necessary. Rule (23) says that all derived memberships should be necessary; otherwise there arises a conflict.

Finally, we present the program $\Pi(\Gamma)$. For each $con \in \Gamma$, we have a program $\Pi(con)$. Hence, $\Pi(\Gamma) = \bigcup_{con \in \Gamma}(\Pi(con))$. Here we only present the representation of a constraint $\langle \gamma, \exists \rangle$; other types of constraints can be likewise encoded. Assume that the credential context is $\mathcal{C}$ and that $L(N) = L(\gamma, \mathcal{C})$, i.e., N recognizes the language $L(\gamma, \mathcal{C})$. We work with N. First we denote the NFA N in ASP. Suppose $N = (S, \Sigma, \delta, s_0, F)$; we use a fact $start(s_0)$ to denote the start state s_0, a fact

$$\begin{aligned}
\mathit{read}(m(X, r(X)), S) &\text{ :- } \mathit{start}(s_0), \mathit{tran}(s_0, X, S), \mathit{need}(m(X, r(X))). && (24)\\
\mathit{read}(m(X, R), S) &\text{ :- } \mathit{read}(m(X, R'), S'), \mathit{tran}(S', R, S),\\
&\qquad \mathit{obt}(m(X, R'), m(X, R), T). && (25)\\
\mathit{read}(M, S) &\text{ :- } \mathit{read}(M, S'), \mathit{tran}(S', \epsilon, S). && (26)\\
\mathit{final}(M) &\text{ :- } \mathit{read}(M, S), \mathit{accept}(S). && (27)\\
\mathit{exist} &\text{ :- } \mathit{final}(M), \mathit{last}(M). && (28)\\
\perp &\text{ :- } \texttt{not}\ \mathit{exist}. && (29)\\
\mathit{last}(M) &\text{ :- } \texttt{not}\ \mathit{not\_last}(M), \mathit{obt}(\_, M, \_). && (30)\\
\mathit{not\_last}(m(X, R)) &\text{ :- } \mathit{obt}(m(X, R), m(X, R'), \_), R \neq R'. && (31)
\end{aligned}$$

Fig. 10. Rules encoding $\langle \gamma, \exists \rangle$

$\mathit{accept}(s_f)$ to denote an accept state $s_f \in F$, and a fact $\mathit{tran}(s_1, e, s_2)$ to denote a transition $s_2 \in \delta(s_1, e)$ where $e \in \mathit{Prin}(\mathcal{C}) \cup \mathit{Role}(\mathcal{C}) \cup \{\epsilon\}$.

Next we test if a role path is accepted by N. Recall that a path is seen as a sequence of principal-role pairs. For example, a path $[e_0, e_1]$ corresponds to a sequence $[e_0 : e_0, e_0 : e_1]$. In our ASP programs, we do not encode the path, but the sequence with atoms of the form $\mathit{obt}(m(e_0, r(e_0)), m(e_0, r(e_1)), t)$, where $m(e_0, r(e_1))$ denotes the principal-role pair $e_0 : e_1$. Therefore, we verify the sequence.

In Fig. 10, rules (24)-(27) simulate N reading an input string and making state transitions accordingly. Rule (24) says, given a sequence $[e_0 : e_0, \cdots]$ as input, N starts in its start state s_0 and, if there is a transition $s \in \mathit{tran}(s_0, e_0)$, proceeds to another state s. In a state s', N proceeds to another state s, depending on what it reads from the input. Suppose that N reaches s' after reading $e_0 : e_i$ and that the next symbol in a sequence is $e_0 : e_j$. In this case, N transits to another state s if there is a transition $s \in \mathit{tran}(s', e_j)$. Rule (25) captures this idea; there, an atom $\mathit{read}(m(e_0, r(e_j)), s)$ means that N is in state s after reading $e_0 : e_j$. Rule (26) interprets the transition ϵ as usual: in a state s', N moves to a state s reachable from s' via ϵ. This reading and transiting process goes on until the input reaches its end or no transition is available. Rule (27) marks a membership final if an accept state is reached after N reads it. If the final membership is also the last one in a sequence, N accepts this sequence; namely, there exists a sequence accepted by N, as stated in rule (28). As required by the constraint, such a sequence must exist; hence rule (29) excludes answer sets where exist is not true. Rules (30) and (31) define the last membership in a sequence.

A Quantitative Approach for Inexact Enforcement of Security Policies[⋆]

Peter Drábik, Fabio Martinelli, and Charles Morisset

IIT-CNR, Security Group
Via Giuseppe Moruzzi 1, 56124 Pisa, Italy
{peter.drabik,fabio.martinelli,charles.morisset}@iit.cnr.it

Abstract. A run-time enforcement mechanism is a program in charge of ensuring that all the traces of a system satisfy a given security policy. Following Schneider's seminal work, there have been several approaches defining what kind of policies can be automatically enforced, and in particular, non-safety properties cannot be correctly and transparently enforced. In this paper, we first propose to build an enforcement mechanism using an abstract notion of selector. We then propose to quantify the *inexact* enforcement of a non-safety property by an enforcement mechanism, by considering both the traces leading to a non-secure output by this mechanism and the secure traces not output, thus formalizing an intuitive notion of security/usability tradeoff. Finally, we refine this notion when probabilistic and quantitative information is known about the traces. We illustrate all the different concepts with a running example, representing an abstract policy dealing with emergency situations.

Keywords: Runtime Enforcement, Safety, Security/Usability Tradeoff.

1 Introduction

An enforcement mechanism is a program in charge of controlling the actions of a target over a system, such that the sequences of actions submitted to the system satisfy a *security policy*. For instance, a security policy can state that the user of a database cannot execute a request to remove a table she does not own, or that an application downloaded onto a mobile operating system cannot modify the core functionalities of the system.

An enforcement mechanism can therefore be seen as a *monitor* between a target, seen as a black-box, and a system, such that only *secure* sequences of actions are executed by the system. There have been several important approaches aiming at understanding and characterizing what kind of policies are enforceable, and we can cite in particular the seminal work of Schneider [18] and the one of Ligatti *et al.* [15]. In the former, Schneider showed that if a policy is not a safety property (*i.e.*, a non-secure trace might be extended to a secure one), then it

⋆ This research was supported by the EU FP7-ICT project NESSoS under the grant agreement n. 256980.

can only be enforced by a "mechanism that uses more information than would be available only from observing the steps of a target's execution". In the latter, Ligatti *et al.* introduce, among others, the edit-automaton, and prove that any security property (over finite traces), including non-safety ones, can be enforced by a monitor built from an edit-automaton.

Indeed, an edit-automaton is powerful enough to *buffer* the entire trace, and to only release the actions submitted by the target when it is certain that they do not violate the policy. However, in many cases, buffering is not a desirable option, for instance when a non-safety policy is built to deal with critical situations, where a non-secure action needs to be performed immediately, and the control might only come afterwards. Such situations can typically occur in the context of healthcare policies [1,5]. Moreover, since a monitor usually needs to output an action to the system whenever the target submits one, buffering the trace would lead to violate the *transparency*[1] of the monitor.

In this paper, we therefore start from the observation that since non-safety policies cannot be exactly enforced (*i.e.*, correctly and transparently), then a monitor can only enforce *inexactly* a non-safety policy. We formalize this concept by characterizing the non-secure traces output by a monitor together with the secure traces not output. By quantifying over these traces, we show that not all inexact monitors are alike, and that it is possible to define a *strategy* to adopt. Intuitively, this strategy corresponds to defining a *security/usability*[2] *tradeoff.*

The main contributions of this paper are three-fold. Firstly, we present a simple framework for runtime monitors with no editing power using a general concept of *selector* (which can be seen as a generalization of a security-automaton) and we define the notion of n-safety policy, which, intuitively, is a policy allowing for at most n non secure steps. Secondly, we introduce the concept of inexact enforcement with the sets $\mathbf{C}$ and $\mathbf{T}$, which, intuitively, contain the traces violating the correctness of the monitor and those violating its transparency, respectively, and we illustrate it on a detailed running example, inspired by a basic break-glass healthcare policy. In particular, we observe that in a basic setting, it might be better to go for an "all-or-nothing" approach. Finally, we present different levels of quantification over these sets, in particular using probabilistic and quantitative information about traces, paving the way towards a formal risk-aware run-time enforcement.

Outline. The rest of this paper is structured as follows: after presenting some related approaches, we introduce in Section 2 the basic concepts such as trace, security property, monitor and enforcement. In Section 3, we define selectors, and we show how to build a monitor. In Section 4, we define the notion of n-safety

[1] We adopt here a strict notion of transparency, which states that a correct input trace should not be edited at all, even if the result of the editing is somehow transparent. This is the reason why we do not consider expressive editing power, such as insertion or suppression, as they are not helpful with respect to strict transparency.

[2] We use here the notion usability in its general meaning, *i.e.*, stating that a system can be used, rather than as the ability to realize user requirements, as it can sometimes be found in security.

property, together with the one of n-selector. In Section 5, we characterize the inexact enforcement of a property, and we define the notion of security/usability tradeoff. In Section 6, we consider the case where the monitor knows the probability of each possible extension. Finally, in Section 7, we introduce the concept of quantitative policy, and we transform the previous tradeoff into the notion of risk exposure, after which we conclude and detail our future work.

Related Work. There has been a significant bulk of work on enforcement mechanisms, especially about their theoretical foundations. We mention here only the closest to our work, and we refer the reader to [12] for a comprehensive survey.

As already mentioned Schneider formalized Execution Monitoring (EM) mechanisms as security automata in [18]. Later, Ligatti *et al.* [15], starting from this initial model, defined a series of automata with increasing power that act as trace consumers and producers. The main novelty was the capability to alter the trace produced by the target, by suppressing, inserting and editing actions. The most powerful class is called edit automata. The semantics of the enforcement mechanisms was still given in terms of big step-semantics for automata. Later work [16] has shown how to enforce non-safety properties, also proposing a notion of enforcement up to equivalence among traces. In [3], Basin *et al.* consider the case where some actions are uncontrollable (*i.e.*, cannot be stopped), and define what policies can then be enforced, by using a Deterministic Turing Machine to model a monitor.

After these different approaches characterizing what properties one can enforce with a specific mechanism, a slightly different strand of work started to consider what happens when limiting the editing power of the edit-automata, already recognized to be too powerful. We can mention here some work considering a limited capability for trace history recording, e.g., shallow history in [9] and limited memory in [19]. The former considers automata that only record the set of relevant actions that happened in the past, while the latter allows only a limited knowledge. Basically, these approaches are limiting the visibility over the past trace, while we are interested here in investigating a limited visibility over the future trace. Some approaches also try to go beyond the traditional runtime enforcement model, such as the definition of facets values [2], where a variable can take different values according to the security level of the user, or the Secure Multi Execution framework [6], where a program is executed multiple times, with different levels of security, thus being able to consider a wider range of security policies, such as non-interference ones.

Another recent class of work is mainly interested in what behaviour an enforcement mechanism should have when the trace is not secure, since traditional definitions do not specify this case. In [4], the authors use a notion of distance among traces, thus expressing that if a trace is not secure, it should be edited to a secure trace close to the non-secure one.

2 Basic Definitions

As we said in the introduction, a monitor is responsible for enforcing a policy over a *system*, and we identify the three following entities: the *target* is an active entity sending *actions* to the system; the system is a passive entity waiting for actions from the target, and executing them; the *monitor* is an entity between the target and the system, receiving actions from the target and sending actions to the system. We now introduce the notion of traces, together with some usual notations, and then we express the notion of policy and monitor.

2.1 Actions and Traces

Let $\mathcal{A}$ be the set of actions that the target can output, such that $\mathcal{A}$ contains the special action $\ominus$, indicating that the target has finished its execution, and does not output any other actions. We consider here traces consisting of an unbounded number of actions, and we say that a trace is *consistent* when any action $\ominus$ is followed only by $\ominus$ actions. We write $\mathcal{T}$ for the set of all finite, unbounded, consistent traces, and given a natural number n, abusing the notation, we write $\mathcal{T}^n$ for the subset of $\mathcal{T}$ consisting of all traces of length n. For the sake of exposition, we consider that any trace σ can always be extended by adding any finite number of $\ominus$ actions, and conversely that the actions $\ominus$ at the end of a trace can always be removed. Given a trace $\sigma \in \mathcal{T}$, we write $|\sigma|$ for the length of σ (potentially including $\ominus$ actions at the end), and we write ϵ for the empty trace.

Given a trace σ, we write σ^i for the action located at the i-th index of σ, starting from 0. Note that we consider that σ^i is always defined, and returns $\ominus$ if i is greater than the length of σ. We extend this notation to write $\sigma^{i<j}$ for the sub-trace of σ between the locations i and $j-1$. For instance, given the trace $\sigma = abcde$, we have $\sigma^{1<3} = bc$. When the first index is equal to 0, we write $\sigma^{0<j} = \sigma^{<j}$, which corresponds to the prefix of length j of the trace. In particular, we have $\sigma^{<0} = \epsilon$.

Given two traces σ and τ in $\mathcal{T}$, we write $\sigma;\tau$ (or $\sigma\tau$ when no confusion can arise) for the concatenation of σ with τ, we write $\sigma \preceq \tau$ to denote that σ is a prefix of τ. Note that we consider the prefix relation to be built modulo the $\ominus$ actions[3] at the end of the trace, in other words, if $\sigma \preceq \tau$, then $\sigma\ominus \preceq \tau$.

2.2 Security Policy and Property

In general, a security policy is expressed over sets of traces, describing all the secure runs of the system. For instance, a security policy could state that the system cannot execute the same trace twice. However, such a policy cannot be enforced by run-time monitoring, since it is usually considered that the monitor can only see the current trace submitted by the target. Following most existing approaches in the context of run-time enforcement, we only consider here security policies that can be defined using a security *property*, which is a computable predicate $\mathcal{P} : \mathcal{T} \to \mathbb{B}$.

[3] In general, we consider the $\ominus$ actions at the end of a trace as non-significant actions, and we use them mostly to avoid checking the length of a trace every time we want to access an action of a trace.

Moreover, we consider here only *reasonable* properties [16], which, in addition to being defined as computable predicates, also always hold over the empty trace, so that a system cannot start in a non-secure state. We therefore exclude liveness properties that require that a particular action is always executed. The reason we impose this constraint is to ensure that a monitor can always output a secure trace. Note that many access control policies (*e.g.*, the Access Matrix [13], the Bell-Padula model [14], the Role-based model [8], etc), can be expressed as security properties, since they do not depend on previous executions of the target, but only on the current state of the system.

Among reasonable properties, the category of *safety* properties has been particularly studied. Intuitively, with a safety property, a non-secure trace cannot be extended into a secure one. More formally:

Definition 1. *A reasonable property $\mathcal{P}$ is a safety property if, and only if:*

$$\forall \sigma_1, \sigma_2 \in \mathcal{T}\ \neg\mathcal{P}(\sigma_1) \wedge \sigma_1 \preceq \sigma_2 \Rightarrow \neg\mathcal{P}(\sigma_2)$$

Running Example. For instance, consider the set of actions $\mathcal{A}_e = \{\mathsf{b}, \mathsf{d}, \mathsf{s}\}$, where b stands for a *benign* action, d for a *dangerous* action and s for a *supervision* action. We want to express that the target can execute any benign action, but must execute first a supervision action before executing a dangerous action, which is given by the following property $\mathcal{P}_e$, which is clearly a safety property:

$$\mathcal{P}_e(\sigma) \Leftrightarrow (\forall i\ \sigma^i = \mathsf{d} \Rightarrow \exists j < i\ \sigma^j = \mathsf{s}).$$

We introduce in Section 4.1 the concept of *n-safety property*, which, roughly speaking, is a property allowing at most n "bad" steps. One of the problems we try to address in this paper is the enforcement of such properties.

2.3 Monitor

A monitor can be represented as a function $E : \mathcal{T} \to \mathcal{T}$, that is, a function taking a trace as an input from the target, and returning another trace to the system. As we said in Section 1, we are interested here in monitors that have no editing power over the trace, only a truncating one. In other words, the trace output by the monitor is either exactly the one input by the target, or its prefix. Intuitively, we want a monitor to be both *secure* (*i.e.*, that the monitor only output secure traces), and *usable* (*i.e.*, that the monitor does not truncate secure traces), which we express with the traditional notions of correct enforcement, transparent enforcement, and exact enforcement (for both correct and transparent) defined as follows.

Definition 2. *An enforcement mechanism $E : \mathcal{T} \to \mathcal{T}$ is said to:*

- *be* conservative *if, and only if, for any trace $\sigma \in \mathcal{T}$, we have $E(\sigma) \preceq \sigma$;*
- correctly *enforce a property $\mathcal{P}$ if, and only if, for any trace $\sigma \in \mathcal{T}$, we have $\mathcal{P}(E(\sigma))$,*

– transparently *enforce a property $\mathcal{P}$ if, and only if, for any trace $\sigma \in \mathcal{T}$ such that $\mathcal{P}(\sigma)$, we have $E(\sigma) = \sigma$.*
– exactly *enforce a property $\mathcal{P}$, which we denote as $E \vdash \mathcal{P}$ if, and only if, E is conservative, and correctly and transparently enforces $\mathcal{P}$,*

It is worth observing that even though we do not consider editing power here, for any reasonable property, there exists a monitor enforcing exactly this property: since the monitor has access to the whole trace, it can simply check whether the security property holds for the trace, and output it in the positive case, or output ϵ otherwise. In other words, the ability to edit a trace is not necessary to enforce exactly a reasonable property, knowing the future of the trace is enough.

We now present a framework based on the notion of *selector*, which is a simple function taking the past trace, the action to control, and decides whether the action should be kept or the trace stopped. To some extent, the role of a selector is close to the one of a security automaton [15]. A monitor is then defined by calling a selector for each action submitted by the target, thus making explicit the fact that a monitor must make a decision at each step, and cannot simply wait for the rest of the trace. We consider the notion n-selectors in Section 4. This framework allows us to tackle the problem of inexact enforcement in Section 5.

3 Selector-Based Monitor

We introduce our framework by revisiting the definition of the class EM (for Execution Monitoring), defined by Schneider [18] as excluding "mechanisms that use more information than would be available only from observing the steps of a target's execution", which we characterize by defining the concept of selector, which embodies the fact that a monitor must make a decision at each step using only past information.

3.1 Selector

A *selector* is a function $F : \mathcal{T} \times \mathcal{A} \to \mathbb{B}$, such that given a trace σ and an action a, $F(\sigma, a)$ is true if and only if the action a should be kept after having executed the trace σ. Moreover, we consider only selectors stating that if an action should not be kept, then any following action should also not be kept. We say that a monitor satisfying this constraint is *consistent*, which is formally specified as:

$$\forall \sigma \in \mathcal{T}\ \forall i\ F(\sigma^{<i}, \sigma^i) = \mathit{false} \Rightarrow \forall j > i\ F(\sigma^{<j}, \sigma^j) = \mathit{false}$$

For instance, the selectors F_{false} and F_{true}, which return always *false* and always *true*, respectively, are trivially consistent. Clearly, this notion of consistency is related to the one of safety. Indeed, given a computable property $\mathcal{P}$, the selector $F_{\mathcal{P}}$, as defined in Equation (1), is clearly consistent when $\mathcal{P}$ is a safety property.

$$F_{\mathcal{P}}(\sigma, a) = \begin{cases} \mathit{true} & \text{if } \mathcal{P}(\sigma; a) \\ \mathit{false} & \text{otherwise} \end{cases} \tag{1}$$

Intuitively, the selector $F_{\mathcal{P}}$ is analogous to a truncation automaton [15].

3.2 From Selector to Monitor

Given a selector, we can straight-forwardly define a monitor by calling the selector at each step. More formally, given a selector $F : \mathcal{T} \times \mathcal{A} \to \mathbb{B}$, we define the monitor $E_F : \mathcal{T} \to \mathcal{T}$ as follows:

$$\forall i \in \mathbb{N}\ E_F(\sigma)^i = \begin{cases} \sigma^i & \text{if } F(\sigma^{<i}, \sigma^i) = \mathit{true} \\ \ominus & \text{otherwise} \end{cases}$$

Clearly, if F is a consistent selector, for any consistent trace σ, $E_F(\sigma)$ is also consistent.For instance, from the two selectors F_{false} and F_{true} described above, we can define the monitors[4] E_{false} and E_{true}, which respectively always outputs the empty trace and always outputs the input trace. It is worth noting that both monitors are conservative, that E_{false} enforces correctly any property and that E_{true} enforces transparently any property. In other words, it is always trivial to build a correct monitor, and it is always trivial to build a transparent monitor, but in general, such monitors are not exact.

When $\mathcal{P}$ is a safety property, we can however always build the monitor $E_{\mathcal{P}}$ from the selector $F_{\mathcal{P}}$ defined in Equation 1, and we can prove that $E_{\mathcal{P}}$ enforces exactly $\mathcal{P}$ (all proofs can be found in the companion report [7]).

Lemma 1. *For any safety property $\mathcal{P}$, $E_{\mathcal{P}} \vdash \mathcal{P}$.*

Running Example. For instance, consider the trace bbsddb. Following the definition of $\mathcal{P}_e$, we have $E_{\mathcal{P}_e}(\mathsf{bbsddb}) = \mathsf{bbsddb}$, because a supervision action is present before any dangerous action. Now, consider the trace bbdsdb: we have $E_{\mathcal{P}_e}(\mathsf{bbdsdb}) = \mathsf{bb}\ominus\ominus\ominus\ominus = \mathsf{bb}\ominus$. Indeed, even though the second dangerous action is preceded by a supervision action, the first one is not, and since the $E_{\mathcal{P}_e}$ is consistent, every action after the first dangerous action included is transformed into the $\ominus$ action.

We can now formally express and prove that, as stated by Schneider [18] "if the set of executions for a security policy is not a safety property, then an enforcement mechanism from EM does not exist for this policy". Moreover, since we do not consider real-time availability problems or infinite memory, we can even prove the converse, as stated in Proposition 1 (which is simply the reformulation of Schneider's result in our framework).

Proposition 1. *Given a property $\mathcal{P}$, there exists an enforcement mechanism E in EM such that $E \vdash \mathcal{P}$ if and only if $\mathcal{P}$ is a safety property.*

This result, although expected, nonetheless narrows down enforceable properties to safety properties, and does not provide a way to enforce non-safety properties, such as those existing in emergency contexts. We present in the following section such properties, and we show in Section 5 how to enforce them inexactly.

[4] For the sake of clarity, when the name of the selector comes with a subscript, such as F_x, when no ambiguity can arise, we write directly E_x for the corresponding monitor instead of E_{F_x}.

4 *n*-safety Properties and *n*-selectors

In this section, we first introduce the concept of n-safety properties, which are properties where a maximum of n non-secure steps is allowed. We then define n-selectors, which are selectors that can use the following n steps to make a decision about an action. Intuitively, we introduce n-selectors to characterize the selection process with complete information about the future, and we use them when we have incomplete or imperfect information (Section 5.2).

4.1 *n*-safety

Although many security policies are safety properties, there also exist some properties which do not satisfy the safety condition, such as some liveness properties, as stated in Section 2.2, or policies including obligations requiring a given action to be executed after a particular trace, otherwise the entire trace would be non-secure. For instance, it might be authorized to create a temporary copy of a file, as long as this copy is deleted after usage. Another example concerns break-glass policies [1,5], usually defined in healthcare environments, where some violations of the property can be tolerated during an emergency situation, as long as an audit is performed afterwards.

In order to define the notion of n-safety, we first introduce the one of *critical trace*: given a trace σ and a property $\mathcal{P}$, a trace τ is said to be critical if, and only if, extending σ with τ can lead to a secure trace even though each step in τ is non-secure. More formally, we define the predicate $\mathsf{D}_{\mathcal{P}}$ as follows:

$$\forall \sigma, \tau \in \mathcal{T}\ \mathsf{D}_{\mathcal{P}}(\sigma, \tau) \Leftrightarrow \forall \tau' \preceq \tau\ (\tau' \neq \epsilon \Rightarrow\ \neg\mathcal{P}(\sigma;\tau')) \wedge \exists \tau'' \in \mathcal{T}\ \mathcal{P}(\sigma;\tau;\tau'')$$

Note that in the above definition, we implicitly assume that $\sigma;\tau$ belongs to $\mathcal{T}$. Clearly, this is not necessarily the case, for instance if σ ends with $\ominus$. In order to be rigorous, we would need to define the set $\mathcal{T}(\sigma)$, as the set of possible extensions from σ. However, for the sake of readability, we assume this constraint to be implicit, and notationally speaking, we use τ to indicate extensions rather than full traces.

We then define a n-safety property as a property holding for the empty trace and admitting critical traces whose length are at most n.

Definition 3. *A property $\mathcal{P}$ is an n-safety property if, and only if*

$$\forall \sigma, \tau \in \mathcal{T}\ D_{\mathcal{P}}(\sigma, \tau) \Rightarrow |\tau| \leqslant n$$

Clearly, given $m \leqslant n$, a m-safety property is also a n-safety property.

Running Example. One of the limitations of the running example is that it requires the target to first ask for supervision before being able to perform the dangerous operation. In some contexts, such as healthcare, it might not be possible to ask for a supervision first, and it might be needed to do the dangerous action first, and then only to ask for the supervisor to verify that everything was correct. For the sake of this example, let us introduce a new "critical" action c, such that when executing c, the supervision can be done at most k steps *after* the action.

For instance, for $k = 2$, the trace cb is critical for the trace b (or any other secure trace that does not contain s), since bcbs is secure while bcbb is not. However, cbb is not critical, since it is not possible to extend cbb to a secure trace. We can then define the property $\mathcal{P}_{k,e}$, which extends $\mathcal{P}_e$ by tolerating at most k non-secure steps, as:

$$\mathcal{P}_{k,e}(\sigma) \Leftrightarrow \mathcal{P}_e(\sigma) \vee (\forall i\ \sigma^i = \mathsf{c} \Rightarrow \exists (j \leqslant i+k)\ \sigma^j = \mathsf{s})$$

The property $\mathcal{P}_{k,e}$ is a combination of a safety property, $\mathcal{P}_e$, and of a k-safety property, and therefore is also a k-safety property.

4.2 *n*-selector

An *n-selector* is a function $F_n : \mathcal{T} \times \mathcal{A} \times \mathcal{T}^n \to \mathbb{B}$, such that for any n, for any traces σ and τ such that $|\tau| = n$, and any action a, $F_n(\sigma, a, \tau)$ indicates whether the action a should be kept knowing that the past trace is σ and that the future execution is τ. Intuitively, an n-selector is a selector with a visibility of n steps over the future trace, and it follows that n is equal to 0, an n-selector is equivalent to a selector. Let us observe that since it is always possible to ignore the end of a trace, given a selector F_n, it is always possible to build an equivalent selector F_m when $n \leqslant m$. For instance, the selectors F_{false} and F_{true} defined in Section 3.1 are also n-selectors, for any n.

Given a property $\mathcal{P}$, we can define the n-selector $F_{n,\mathcal{P}}$ in a similar fashion than the selector $F_{\mathcal{P}}$:

$$F_{n,\mathcal{P}}(\sigma, a, \tau) = \begin{cases} \mathit{true} & \text{if there exists } \tau' \preceq \tau\ \mathcal{P}(\sigma; a; \tau') \\ \mathit{false} & \text{otherwise} \end{cases} \tag{2}$$

For any n-safety property, the selector $F_{n,\mathcal{P}}$ is consistent, since if for some trace σ and index i, we have $F_{n,\mathcal{P}}(\sigma^{<i}, \sigma^i, \sigma^{i+1<i+n}) = \mathit{false}$, then by definition of n-safety, we know that any extension of σ^{i+n} is non-secure.

We show in [7] that n-selectors can be directly used to enforce exactly n-safety properties. However, in order to use an n-selector, one needs to be able to produce the n following steps of the trace, and although it could be possible in some contexts (for instance when the entire input trace is known in advance, or by buffering the n following steps, such as done by an edit-automaton [15]), in general the monitor does not have access to this information. For instance, in the running example, buffering the trace would defeat the purpose of the n-safety property, that is to accept time-critical actions in the absence of supervision. We study in the next section the problem of *inexact* enforcement.

5 Inexact Enforcement of *n*-safety Properties

An n-safety property can only be enforced *inexactly* (*i.e.*, either not correctly or not transparently, or both) by a mechanism in EM (i.e. with no precise knowledge about the future). We address in this section the problem of inexact enforcement, and we propose to quantify the necessary security/usability tradeoff.

5.1 Security/Usability Tradeoff

In order to quantify the non-correctness of a monitor, given a monitor E and a property $\mathcal{P}$, we introduce the set $\mathbf{C}_{\langle E,\mathcal{P}\rangle}$, which represents all traces for which the monitor outputs a non-secure trace. Similarly, the quantification of the non-transparency is done using the set $\mathbf{T}_{\langle E,\mathcal{P}\rangle}$, which represents the secure traces that the monitor does not output as they are. More formally, we have:

$$\mathbf{C}_{\langle E,\mathcal{P}\rangle} = \{\sigma \in \mathcal{T} \mid \neg\mathcal{P}(E(\sigma))\} \qquad \mathbf{T}_{\langle E,\mathcal{P}\rangle} = \{\sigma \in \mathcal{T} \mid \mathcal{P}(\sigma) \wedge E(\sigma) \neq \sigma\}.$$

When E enforces exactly $\mathcal{P}$, we clearly have $\mathbf{C}_{\langle E,\mathcal{P}\rangle} = \mathbf{T}_{\langle E,\mathcal{P}\rangle} = \emptyset$. Conversely, since an n-safety property cannot be enforced exactly using a selector, at least one of these two sets is necessarily non empty.

We showed in Section 3.2 that it is always trivial to define a monitor that enforces a property either correctly or transparently, with the monitors E_{false} and E_{true}, respectively. It follows that minimizing one set without considering the other is trivial, and thus we are interested here in minimizing both sets together. In other words, we want to establish a tradeoff between security (*i.e.*, correctness) and usability (*i.e.*, transparency), when both cannot be achieved simultaneously. Intuitively, the bigger the set $\mathbf{C}_{\langle E,\mathcal{P}\rangle}$ is, the less secure E is, while the bigger the set $\mathbf{T}_{\langle E,\mathcal{P}\rangle}$, the less usable E is.

5.2 Building a Selector from an n-selector

Given a trace σ and an action a, we can define a selector F from the n-selector $F_{n,\mathcal{P}}$ defined in Section 4.2. In order to illustrate the general approach, let us first consider a very simple case, and let us assume that only two traces τ_1 and τ_2 of length n are possible after $\sigma; a$ (meaning that the whole trace is either $\sigma; a; \tau_1$ or $\sigma; a; \tau_2$). Three cases might occur:

– $F_{n,\mathcal{P}}(\sigma, a, \tau_1) = F_{n,\mathcal{P}}(\sigma, a, \tau_2) = \mathit{true}$, and we can define $F(\sigma, a) = \mathit{true}$;
– $F_{n,\mathcal{P}}(\sigma, a, \tau_1) = F_{n,\mathcal{P}}(\sigma, a, \tau_2) = \mathit{false}$, and we can define $F(\sigma, a) = \mathit{false}$;
– $F_{n,\mathcal{P}}(\sigma, a, \tau_1) = \mathit{true}$ and $F_{n,\mathcal{P}}(\sigma, a, \tau_2) = \mathit{false}$ (the converse case being equivalent), and in this case, the selector F has to make a choice: either to stop and not output the secure trace $\sigma; a; \tau_1$, or to continue and output the non-secure trace $\sigma; a; \tau_2$ (or a non-secure prefix of it).

The first two cases are rather straight-forward, and the difficulty clearly lies in the third case. In general, this choice has to be made with respect to all possible extensions of length n, and not only τ_1 and τ_2. The crucial point is to make a decision for a trace σ and an action a such that $\mathcal{P}(\sigma)$ holds and $\mathcal{P}(\sigma; a)$ does not, and when there exists a trace τ such that $F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{true}$. Indeed, at this point, the selector is left with two choices: either accepting a and taking the risk to output an non-secure trace, for instance if the target stops right after a or if the trace τ is not output, or stopping at this point, and therefore not outputting the secure trace $\sigma; a; \tau$.

Since at this level of formalism, we only compare two traces based on the fact that they are secure or not (we consider in the following sections the probability and the impact of a trace), we propose here a general selector F_r, specifying that the number non-secure extensions must not exceed a given ratio r of the total number of traces:

$$F_r(\sigma, a) = \begin{cases} \mathit{true} & \text{if there exists } \tau \in \mathcal{T}^n \text{ such that } F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{true} \\ & \quad \text{and } |\{\tau \in \mathcal{T}^n \mid F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{false}\}| \leqslant r\,|\mathcal{T}^n| \\ \mathit{false} & \text{otherwise} \end{cases}$$

In particular, the selector F_0 only accepts an action when it is certain that every possible extension is secure, while the selector F_1 accepts actions as long as there is a possibility for a secure extension. Note that given any $0 \leqslant r \leqslant 1$, if a trace is secure at each step, then F_r accepts it, and if a trace is irremediably bad (*i.e.*, there is no possibility to extend it to a secure trace), then F_r stops the trace.

5.3 From Local Ratio to Global Ratio

The local ratio r defined in the previous section acts as "worst-case scenario", since in any case, the selector F_r stops a trace whenever the proportion of non-secure extensions is above this ratio. However, from a global perspective, we are mostly interested in quantifying the proportion of non-secure traces output and of secure traces not output from *all* possible traces. We thus define $\mathbf{QC}\langle E,\mathcal{P},k\rangle$ and $\mathbf{QT}\langle E,\mathcal{P},k\rangle$ as the proportion of the number of traces of length k in $\mathbf{C}\langle E,\mathcal{P}\rangle$ and in $\mathbf{T}\langle E,\mathcal{P}\rangle$ to all the traces in $\mathcal{T}^k$, respectively.

To some extent, these two values can be seen as *utility* functions of a monitor with respect to a property, and therefore establishing the security/usability tradeoff consists in providing a minimization strategy for these utility functions. For instance, a very simple strategy could be to minimize the sum $(\mathbf{QC}\langle E,\mathcal{P},k\rangle + \mathbf{QT}\langle E,\mathcal{P},k\rangle)$, or to define a threshold on $\mathbf{QC}\langle E,\mathcal{P},k\rangle$ and to minimize $\mathbf{QT}\langle E,\mathcal{P},k\rangle$.

The local ratio r can be therefore seen as a pessimistic approach, and it is straight-forward to observe that $\mathbf{QC}\langle E_r,\mathcal{P},k\rangle \leqslant r$. However, as we illustrate in the next section, it is possible to have a situation where r is much greater than $\mathbf{QC}\langle E_r,\mathcal{P},k\rangle$, and in general, given a specific strategy over $(\mathbf{QC}\langle E_r,\mathcal{P},k\rangle, \mathbf{QT}\langle E_r,\mathcal{P},k\rangle)$, it is not trivial to define an r that satisfies it.

Indeed, the selector only makes a decision based on the relative number of possible non-secure extensions, which is not necessarily representative of the global number of non-secure traces. In other words, the local ratio r should be considered as a parameter of the selector specifying which behaviour to adopt when facing a critical trace, rather than a global measure of inexactitude. Let us nonetheless observe that for any specific ratio, given a fixed trace length k, we can derive the values $\mathbf{QC}\langle E_r,\mathcal{P},k\rangle$ and $\mathbf{QT}\langle E_r,\mathcal{P},k\rangle$. In other words, the required ratio r can be *inferred* from the strategy, as we illustrate in the next section. We believe the problem of calculating the optimal ratio r for any strategy is a challenging one, and we plan to address it in future work.

Table 1. Tradeoff values for $\mathcal{P}_{2,e}$ and $k = 4$

E ratio	E_{false}	E_{r_1} $0 \leqslant r_1 < 2/3$	E_{r_2} $2/3 \leqslant r_2 < 16/21$	E_{r_3} $16/21 \leqslant r_3 \leqslant 1$	E_{true}
$\mathbf{QC}\langle\ \rangle$	0	0	0.25	0.21	0.54
$\mathbf{QT}\langle\ \rangle$	0.45	0.11	0.04	0	0

5.4 Running Example

Let us consider the 2-safety property $\mathcal{P}_{2,e}$, as defined in Section 4.1, and a trace length of 4. We recall that a trace consists of any sequence of actions in $\{\mathsf{b}, \mathsf{d}, \mathsf{s}, \mathsf{c}\}$, possibly followed by $\ominus$ actions. It follows that $\left|\mathcal{T}^4\right| = \frac{4^5-1}{3} = 341$, with the following break-down: there are 112 non-secure traces caused by a d not preceded by an s (e.g., $\mathsf{bd}\ominus\ominus$); 73 non-secure traces caused by a c not followed by an s in the next two steps (e.g., $\mathsf{cbb}\ominus$); 12 secure traces with exactly two non-secure steps (e.g., bcbs); 27 secure traces with exactly one non-secure step (e.g., csdb) and 117 secure traces without non-secure step (e.g., $\mathsf{sdc}\ominus$).

The monitor E_{true} outputs all 185 non-secure traces and all secure traces, and therefore we have $\mathbf{QC}\langle E_{true}, \mathcal{P}_{2,e}, 4\rangle \approx 0.54$ and $\mathbf{QT}\langle E_{true}, \mathcal{P}_{2,e}, 4\rangle = 0$. Conversely, the monitor E_{false} does not output any non-secure trace, but only output the secure trace ϵ, meaning that $\mathbf{QC}\langle E_{false}, \mathcal{P}_{2,e}, 4\rangle = 0$ and $\mathbf{QT}\langle E_{false}, \mathcal{P}_{2,e}, 4\rangle \approx 0.45$. Intuitively, those two monitors should act as "worst cases", meaning that defining a monitor worse than them is pointless.

Now, let us identify the different possible ratios r, and the corresponding tradeoff values. Let σ be a trace that does not contain s and the action c, then there are 14 traces τ out of 21 of length 2 such that $F_{2,\mathcal{P}_{2,e}}(\sigma, \mathsf{c}, \tau) = \mathit{false}$. Hence, for any ratio $0 \leqslant r_1 < 2/3$, $F_{r_1}(\sigma, \mathsf{c}) = \mathit{false}$, which means that E_{r_1} does not accept any trace with a non-secure step. On the other hand, if the ratio is above $2/3$, the action c is accepted. Now, given the trace $\sigma; \mathsf{c}$, where σ does not contain s, and an action a either equal to b or to c, then there are 16 traces τ out of 21 of length 2 such that $F_{2,\mathcal{P}_{2,e}}(\sigma; \mathsf{c}, a, \tau) = \mathit{false}$. It follows that given a ratio $2/3 \leqslant r_2 < 16/21$, $F_{r_2}(\sigma; \mathsf{c}, a, \tau) = \mathit{false}$. Finally, it is easy to see that if the ratio is above $16/21$, then the selector only stops when all future traces are non-secure. We summarize these results in Table 1.

Interestingly, we can observe that the monitor E_{r_2} is always worse than E_{r_3}, which is due to the fact that since F_{r_2} always accepts the first c, but stops if the following action is not an s, then it outputs a non-secure trace in this case, even for the correct input traces. In other words, it means that without further information about the probability or the impact of a trace, an "all-or-nothing" approach should be adopted: either all secure traces should be output, with E_{r_3}, or none, with E_{r_1}.

6 Probabilistic Enforcement

In the previous section, we implicitly consider any two traces have the same probability to occur. However, in some cases, it is possible to quantify the *likelihood* of a particular trace to happen, typically when the probabilistic behaviour of the target is provided. For instance, consider the running example: it could be possible to determine the likelihood of a particular nurse to ask for supervision after executing a dangerous action, by statistically analyzing the past activity of this nurse, or by using a pre-established level of trust for this nurse. In general, let us assume that there exists a *conditional probability function* $\psi(\tau|\sigma)$, indicating the probability of the trace τ to occur after the trace σ. We require such a function to be a probability distribution for traces of same length, that is, $\sum_{\tau \in \mathcal{T}^n}(\tau \mid \sigma) = 1$, for any n and any σ.

The security/usability tradeoff can now be expressed with $\mathbf{PC}\langle E,\mathcal{P},k\rangle$, which represents the accumulated probability for the target to output a trace of length n such that the monitor outputs a non-secure trace, and $\mathbf{PT}\langle E,\mathcal{P},k\rangle$, which represents the accumulated probability for the target to output a secure trace of length n such that the monitor does not output the same trace. More formally, given a monitor E, an n-safety property $\mathcal{P}$ and trace length k, we define:

$$\mathbf{PC}\langle E,\mathcal{P},k\rangle = \sum_{\sigma \in \mathcal{T}^k} \{\psi(\sigma|\epsilon) \mid \sigma \in \mathbf{C}\langle E,\mathcal{P}\rangle\}$$

$$\mathbf{PT}\langle E,\mathcal{P},k\rangle = \sum_{\sigma \in \mathcal{T}^k} \{\psi(\sigma|\epsilon) \mid \sigma \in \mathbf{T}\langle E,\mathcal{P}\rangle\}$$

The ratio-based selector F_r defined in Section 5.3 can therefore be refined to a probabilistic ratio-based selector F_{p_c}, such that given a trace σ, an action a and a probability threshold p_c, F_{p_c} is defined by:

$$F_{p_c}(\sigma, a) = \begin{cases} \mathit{true} & \text{if: - there exists } \tau \in \mathcal{T}^n \text{ such that } F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{true}; \\ & \quad\text{- } \psi(\tau|\sigma; a) \neq 0; \text{ and} \\ & \quad\text{- } \sum_{\tau \in \mathcal{T}^n} \{\psi(\tau|\sigma; a) \mid F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{false}\} \leqslant p_c \\ \mathit{false} & \text{otherwise} \end{cases}$$

Following a similar reasoning to the one for F_r, we can see that $\mathbf{PC}\langle E_{p_c},\mathcal{P},k\rangle \leqslant p_c$, since at each step, the monitor only keep accepting actions if the aggregated probability of reaching a bad trace is lower than p_c.

7 Quantitative Enforcement

In some context, it can be argued that outputting a particular non-secure trace does not have the same *impact* as outputting another one. For instance, in the running example, the impact of the trace dddd is intuitively much higher than that of b$\ominus\ominus\ominus$. In other words, quantifying only over the sets $\mathbf{C}\langle E,\mathcal{P}\rangle$ and $\mathbf{T}\langle E,\mathcal{P}\rangle$ might not be enough, even with the probability of traces. In this section, we consider that in addition to the security property, we are provided with a *quantitative property*, associating each trace with a specific impact.

Given a set of traces $\mathcal{T}$ and a domain of values $(\mathbb{V}, \leq, +, -)$, a *quantitative property* is a function $Q : \mathcal{T} \to \mathbb{V}$. For the sake of generality, we assume that both $\mathcal{P}$ and Q coexist, as they serve a different purpose: $\mathcal{P}$ indicates which traces are secure, while Q indicates the impact of each trace. For instance, Q might denote the financial cost on the system to accept a trace, which could be in some instances smaller for a non-secure trace than for a secure one.

Given an n-safety quantitative property $(\mathcal{P}, Q)$, an enforcement mechanism $E : \mathcal{T} \to \mathcal{T}$ and a fixed length of traces k, we define $\mathbf{RC}_{\langle E,\mathcal{P},k\rangle}$, which corresponds to the risk created by accepting non-secure traces, and therefore corresponds to a notion of *damage*, and $\mathbf{RT}_{\langle E,\mathcal{P},k\rangle}$, which corresponds to the risk of not accepting some secure traces, and therefore rather corresponds to a notion of *loss*:

$$\mathbf{RC}_{\langle E,\mathcal{P},Q,k\rangle} = \sum_{\sigma\in\mathcal{T}^k} \{\psi(\sigma|\epsilon)\cdot Q(E(\sigma)) \mid \sigma \in \mathbf{C}_{\langle E,\mathcal{P}\rangle}\}$$
$$\mathbf{RT}_{\langle E,\mathcal{P},Q,k\rangle} = \sum_{\sigma\in\mathcal{T}^k} \{\psi(\sigma|\epsilon)\cdot (Q(\sigma) - Q(E(\sigma)) \mid \sigma \in \mathbf{T}_{\langle E,\mathcal{P}\rangle}\}.$$

It is worth observing that using the monitor E_{false} leads to $\mathbf{RC}_{\langle E_{false},\mathcal{P},k\rangle} = 0$, which corresponds to the intuition that refusing every trace does not create any risk in terms of security, while for the monitor E_{true}, we have $\mathbf{RT}_{\langle E_{true},\mathcal{P},k\rangle} = 0$, which corresponds to the intuition that accepting every trace does not create a risk of loss of usability. Let us also observe that if $\mathcal{P}$ is a safety property, then $\mathcal{P}$ can be exactly enforced by the monitor $E_{\mathcal{P}}$, with $\mathbf{RC}_{\langle E_{\mathcal{P}},\mathcal{P},k\rangle} = \mathbf{RC}_{\langle E_{\mathcal{P}},\mathcal{P},k\rangle} = 0$. This observation is consistent with the intuition that if the future can be ignored, then the monitor does not need to expose itself to any risk.

In general, given an n-safety property $\mathcal{P}$ and a risk threshold λ, indicating the risk limit a monitor can take, we can define the selector F_λ, such that given a trace σ and an action a:

$$F_\lambda(\sigma, a) = \begin{cases} \mathit{true} & \text{if } \sum_{\tau\in\mathcal{T}^n} \{\psi(\tau|\sigma; a)\cdot Q(\sigma; a; \tau) \mid F_{n,\mathcal{P}}(\sigma, a, \tau) = \mathit{false}\} \le \lambda \\ \mathit{false} & \text{otherwise} \end{cases}$$

Following a similar reasoning to the one for F_{p_c}, we can see that $\mathbf{RC}_{\langle E_\lambda,\mathcal{P},k\rangle} \le \lambda$, since at each step, the monitor only keep accepting actions if the aggregated probability of reaching a bad trace is lower than λ.

Running Example. We measure the impact of a trace by the number of non-secure actions executed by the system: we add 1 for each d not preceded by s, and 1 for each c neither preceded by s nor followed by s in the two following steps. For instance, $Q_e(\mathsf{ddsd}) = 2$, and $Q_e(\mathsf{cbbs}) = 1$. Note that in this case $\mathbf{RT}_{\langle E,\mathcal{P}_{2,e},Q_e,4\rangle}$ roughly measures the impact of secure traces that have been output as non-secure traces by E. We present in Table 2 the different thresholds experimentally obtained for the parameter λ, assuming all traces are equiprobable.

Two results are particularly interesting to observe. Firstly, the monitor E_{λ_2}, which corresponds to the monitor E_{r_2}, has a lower impact than the monitor E_{λ_4}, which corresponds to the monitor E_{r_3}. In other words, when the impact of traces is taken into account, then an approach "all-or-nothing" is not necessarily the best, and it might be worth stopping in a non-secure state, even though

Table 2. Tradeoff values for $\mathcal{P}_{2,e}$, Q_e, $k = 4$ and equiprobable traces

E ratio	E_{false}	E_{λ_1} $0 \leqslant \lambda_1 < 1.34$	E_{λ_2} $1.34 \leqslant \lambda_2 < 1.48$	E_{λ_3} $1.48 \leqslant \lambda_3 \leqslant 2.1$	E_{λ_4} $2.1 \leqslant \lambda_4$	E_{true}
QC	0	0	0.25	0.23	0.21	0.54
QT	0.45	0.11	0.04	0.02	0	0
RC	0	0	0.25	0.23	0.28	1.09
RT	0	0	0.04	0.02	0	0

there still is a possibility to find a secure extension. Secondly, the monitor E_{λ_3} is created, and the only difference with E_{λ_2} is that E_{λ_3} accepts the first non-secure c, and accepts the following action only if it is a b or a s. In other words, E_{λ_3} is willing to take the risk of the first non-secure c, but not of a second one.

8 Conclusion

In this paper, we have presented the problem of inexact enforcement of security policies, and of non-safety properties in particular. We have formalized the concept of security/usability tradeoff by quantifying over the set of non-secure traces output by a monitor and the set of secure traces not output. When probabilistic knowledge about the future trace is available, we can define the notion of probability of failure. Moreover, when quantitative information is also available, then it is possible to define the risk exposure of a monitor. We have illustrated that the best approach to adopt when dealing with critical traces depends on whether quantitative information is available or not, which reinforces the idea that security policies need to include a quantitative aspect [11].

The ultimate goal, the construction of the optimal monitor that minimizes the risks or its approximations, is an ongoing and future work. Verification techniques such as probabilistic model checking [10] might be a useful tool, since they enable analysis of quantitative properties of probabilistic models. An interesting approach is to consider the tradeoff as some utility functions that need to optimized, such as it is done in [17], where the authors model access control systems as Markov Decision Processes, thus leading towards a general notion of utility-based security. We also need to validate the applicability of our approach for real-world scenarios, and to understand whether the calculations can be done at runtime, or should rather be done as an analysis of a security property. Finally, we would like to increase the expressiveness of monitors, in order to consider editing of a bad trace, which extends the range of the choices to analyze at run-time.

References

1. Ardagna, C.A., De Capitani di Vimercati, S., Grandison, T., Jajodia, S., Samarati, P.: Regulating Exceptions in Healthcare Using Policy Spaces. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 254–267. Springer, Heidelberg (2008)

2. Austin, T.H., Flanagan, C.: Multiple facets for dynamic information flow. In: Proceedings of POPL 2012, pp. 165–178. ACM, New York (2012)
3. Basin, D., Jugé, V., Klaedtke, F., Zălinescu, E.: Enforceable Security Policies Revisited. In: Degano, P., Guttman, J.D. (eds.) POST 2012. LNCS, vol. 7215, pp. 309–328. Springer, Heidelberg (2012)
4. Bielova, N., Massacci, F.: Predictability of Enforcement. In: Erlingsson, Ú., Wieringa, R., Zannone, N. (eds.) ESSoS 2011. LNCS, vol. 6542, pp. 73–86. Springer, Heidelberg (2011)
5. Brucker, A.D., Petritsch, H.: Extending access control models with break-glass. In: Proceedings of SACMAT 2009, pp. 197–206. ACM, New York (2009)
6. Devriese, D., Piessens, F.: Noninterference through secure multi-execution. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy, SP 2010, pp. 109–124. IEEE Computer Society, Washington, DC (2010)
7. Drábik, P., Martinelli, F., Morisset, C.: A quantitative approach for the inexact enforcement of security policies. Technical Report TR-07-2012, IIT-CNR (2012)
8. Ferraiolo, D.F., Kuhn, D.R.: Role-based access control. In: Proceedings of the 15th National Computer Security Conference, pp. 554–563 (1992)
9. Fong, P.W.L.: Access control by tracking shallow execution history. In: Proceedings of Security and Privacy, pp. 1–13 (2004)
10. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Bernardo, M., Issarny, V. (eds.) SFM 2011. LNCS, vol. 6659, pp. 53–113. Springer, Heidelberg (2011)
11. Kephart, J.: The utility of utility: Policies for self-managing systems. In: Proceedings of POLICY 2011, Pisa, Italy. IEEE Computer Society (2011)
12. Khoury, R., Tawbi, N.: Which security policies are enforceable by runtime monitors? a survey. Computer Science Review 6(1), 27–45 (2012)
13. Lampson, B.: Protection. In: Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems, pp. 437–443. Princeton University (1971)
14. LaPadula, L., Bell, D.: Secure Computer Systems: A Mathematical Model. Journal of Computer Security 4, 239–263 (1996)
15. Ligatti, J., Bauer, L., Walker, D.: Edit automata: Enforcement mechanisms for run-time security policies. Journal of Information Security 4(1-2), 2–16 (2005)
16. Ligatti, J., Bauer, L., Walker, D.: Run-time enforcement of nonsafety policies. ACM Transactions on Information and System Security 12(3), 1–41 (2009)
17. Martinelli, F., Morisset, C.: Quantitative access control with partially-observable markov decision processes. In: Proceedings of ACM CODASPY 2012, pp. 169–180. ACM, New York (2012)
18. Schneider, F.B.: Enforceable security policies. ACM Trans. Inf. Syst. Secur. 3, 30–50 (2000)
19. Talhi, C., Tawbi, N., Debbabi, M.: Execution monitoring enforcement under memory-limitation constraints. Information and Computation 206(2-4), 158–184 (2008)

OSDM: An Organizational Supervised Delegation Model for RBAC

Nezar Nassr[1], Nidal Aboudagga[2], and Eric Steegmans[1]

[1] Katholieke Universiteit Leuven,
Dept. of Computer Science and Engineering,
Leuven, Belgium
{nezar.nassr,eric.steegmans}@cs.Kuleuven.be
[2] SecureICT, Belgium
nidal.aboudagga@secureict.com

Abstract. The dynamic nature of operations in organizations has led to an interest in roles and permissions delegation to enable a seamless continuity of business. Delegation involves assigning a given set of access rights from one user to another. In existing role delegation models, delegation is often authorized and controlled by a relation that specifies who can delegate to whom. The usage of such relations in delegation models has some disadvantages; such as complexity of maintenance, error proneness, inconsistencies and inabilities to define some organizational policies related to delegation. In this paper, we propose a new delegation model that depends on organizational lines of authority to authorize and control delegation. The main advantages of this approach are that it simplifies the management of delegation authorization and complies with organizational behavior. Furthermore, it eliminates inconsistencies related to changes to roles and permissions.

Keywords: Access Control, RBAC, Delegation, Revocation.

1 Introduction

Role-based access control (RBAC)[1] has become the dominant authorization mechanism used in a wide range of organizations. RBAC has gained wide acceptance since it greatly simplifies the management of access rights. Moreover, RBAC attempts to simulate organizational structures at a high level by its hierarchical model. In RBAC, roles are assigned to users and permissions are associated to roles. Users represent staff in organizations. Roles represent the job functions of the users or sub-functions in some cases. Permissions are privileges for accessing objects or performing activities.

High dependability of organizations on access control systems and the dynamic nature of operations have shown a demand for dynamism in the access control systems in place. RBAC supports the principle of least privilege [16], which entails that users are assigned the minimum privileges required for achieving their functions. However, this has led to situations where some specific privileges are assigned only to very few users. Despite the clear advantages of this approach, it restrains access to important resources to a small closed group. At times, none of those users maybe available. This may hamper certain activities within the organization at stake. These situations have led to requirements of increased dynamism of the access control systems in place.

Role delegation is a mechanism of performing a takeover on a user's access rights. Delegation gives authority of a user on another user's access privileges to perform functions of the user originally assigned to the delegated access rights. Existing delegation models suggest that delegation can take two forms: administrative delegation and user delegation. In administrative delegation, an administrative user assigns access rights to a user while the administrative user is not necessarily assigned to the delegated role. In user delegation, a user assigns a subset of his available rights to another user [2].

Most of the existing delegation models [3][4][5][6][7] use a relation to authorize delegation, that is used to determine which user can delegate to whom. The delegation relation often takes the form *can-delegate(R, some conditions)*, where R is the role to be delegated, the conditions specify who can delegate the role, in addition to other parameters such as depth of delegation. The conditions often take the form: a user who has role x can delegate role y. The delegation relation (*can-delegate*) brings some disadvantages to the delegation model:

1. The delegation relations in existing delegation models add complexity to the access control model. Large organizations typically have a rather large number of roles. By adding delegation relations for most of these roles, the entire system may explode. This is blocking for large organizations since they require huge efforts for defining a huge number of relations.
2. The delegation relations cannot express precise conditions on who can delegate a specific role. As an example, consider a delegation relation that states that a professor can delegate the teaching assistant role. This means that a professor in the faculty of arts can delegate the teaching assistant role to a user in the computer science department.
3. The delegation relations may become inconsistent if updates to RBAC relations are allowed such as updates to the role hierarchy [2]. Such updates may occur when new activities are deployed in the organization. This adds huge efforts for the maintenance of the relations, specifically in cases of updates to roles such as adding or removing roles, as well as updates to the hierarchies of roles. Such updates are likely to happen in organizations.
4. User delegation models suggest that a user who is delegating an access right has to be assigned to it [2]. This is not necessarily valid, since it is possible that the user possessing the access right to be delegated is absent. Furthermore, it is not guaranteed that another user who possesses the same access right is available, especially in case of emergency.

In this paper we propose a novel form of delegation, called the organizational supervised delegation model (OSDM). The idea came after surveying some access control models and policies in some organizations. Our surveys included one of the largest European banks, a European university and a software provider.

We have discussed the applicability of the delegation authorization relations provided in existing delegation models [3][4][5][6][7]. All the surveyed organizations commented negatively on these relations, and mainly regarding complexity and the huge number of relations to be defined which makes managing them extremely hard. Furthermore, organizations often adapt a different approach for authorizing delegation. We have found that any user-role assignment or role (or permissions) delegation must be approved by the line managers of the users. When roles are delegated, the delegation request is initiated by a user, then approved by another user then executed by a user or a process. No user can delegate a role assigned to himself without approval. The user who approves delegation is not necessarily assigned to the role or to the permissions to be delegated. OSDM depends on organizational hierarchies to find users who must approve delegation, according to lines of authority defined in the organizational structure. OSDM addresses the above mentioned limitations of existing role delegation models.

The remainder of this paper is organized as follows: In the second section we show an overview of organizational structures. In the third section we review existing role delegation models. The fourth section presents our proposed delegation model. Section five is a discussion on OSDM. Finally, section six concludes our work.

2 Overview of Organizational Structures

Organizational structures outline the planned pattern of positions of individuals, job duties and activities to be achieved. They also describe the lines of authority among different parts of the organization [8]. Organizational structures are modeled using organizational charts that depict the relationships between different positions and the hierarchy that represents authority depending on the rank of users. Organizational structures can take several forms. Most of structures used are hierarchical, matrix, and the flat organizational structures. In a flat structure, all employees report directly to a single manager. In hierarchical structures, each individual reports to one and only one manager at the next higher level [9]. Authority is clear in hierarchical organizations, and managers have absolute authority on their teams.

The matrix structure involves dual authorities, where individuals can report to two managers. Employees often have a functional manager and participate in projects that have a project manager. Users report some activities to the project manager and some activities to their functional manager. Authority of functional managers and project managers in the matrix structure varies according to the type of the matrix. Resource assignment is normally controlled by the functional manager. Therefore, we are interested in functional managers for our delegation model, since we care about who has the power to approve resource assignment and therefore delegation.

We have developed an example users hierarchy to be used for explanation throughout the rest of the paper. Fig. 1 depicts the users hierarchy of a software development department in a technology department of a hierarchical organizational structure.

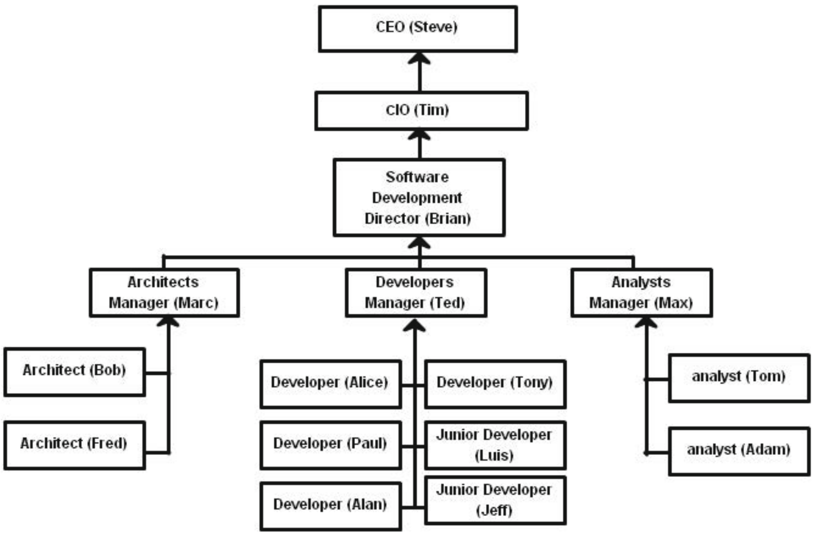


Fig. 1. Software development department users hierarchy of the organizational structure

3 Related Work

In the last few years, several role delegation models [3][4][5][6][7] studying delegation in the context of role based access control have emerged. However, delegation was studied before RBAC was proposed, and there were some predecessors of RBAC delegation such as: the access matrix models which introduced the concept of copy flag, which allows users to delegate rights [10]. Wood and Fernandez [11] introduced the idea of reverting the rights to the upper level after revoking a low-level delegation. Graph-based delegation was introduced in [12]. A variety of delegation approaches were also introduced in [13]. In this paper, we focus on delegation in the context of RBAC. Delegation in RBAC can have several characteristics depending on the requirements of the environment where delegation is applied. The main characteristics of delegation were explained by Barka et al. [14]. These key characteristics include permanence, monotonicity, totality, administration, levels of delegation, multiple delegation, lateral agreements, cascading revocation, and grant-dependency revocation [14].

The first work that studied delegation in RBAC was achieved by Barka and Sandhu [3][14]. They proposed the RBDM0 delegation model [3] which studied delegation in flat roles structure. RBDM0 focused on *grant total delegation* which means that the delegator keeps the power to use the role after delegation and covers only the delegation of roles. RBDM0 does not support partial role permissions delegations. RBDM0 controls user-user delegation by means of the *can-delegate* relation. The *can-delegate* relation takes the form of *(a, b)* $\in$ *can-delegate*. It means that a user who is an original member of the role *a* can delegate his role to a user who is an original member of role *b*. Revocation in RBDM0 can happen in two ways: firstly, by time outs. Delegations are revoked when the delegation period expires. Secondly, any original member of the delegated role can revoke the membership of any delegate member in that role.

RDM2000 [5][15] was the first delegation model to address delegation with hierarchical roles. It also supports multi-step delegation. The *can-delegate* relation in RDM2000 takes the form: *can-delegate* $\subseteq R \times CR \times N$, where R are sets of roles, CR are prerequisite conditions, and N is the maximum delegation depth. The meaning of *(r, cr, n)*$\in$ *can-delegate* is that a user who is a member of role r (or a role senior to r) can delegate role r (or a role junior to r) to any user whose current entitlements in roles satisfy the prerequisite condition cr without exceeding the maximum delegation depth n.

The permission-based delegation model (PBDM) [6] was the first to address permission delegation (partial delegation). PBDM supports role as well as permission delegations with features of multi-step delegation and multi-option revocation. PBDM comprises in two models; PBDM0 and PBDM1. In PBDM0, permission delegation involves three steps. Firstly, a temporary delegation role is created by the delegator. Secondly, the permissions to be delegated are assigned to the temporary role with permission-role assignment. Thirdly, the delegator assigns the temporary role to the delegatee by user-role assignment. Revocation in PBDM0 includes three cases: by revoking the delegated role, by removing one or more pieces of permissions from the delegated role, or by revoking the user-delegation role assignment.

PBDM1 extends PBDM0 with two main features. Firstly, it adds support for role-role delegation, which supports delegating specific permissions of a role to another role rather than to another user. Secondly, it adds means for controlling delegation; to restrict delegation only to authorized users. This is achieved by the *can-delegate* relation, which takes the form: *can-delegate* $\subseteq$ *DBR* $\times$ *Pre-con* $\times$ *P-Range* $\times$ *M*, where *DBR* are sets of delegable roles, *Pre-con* are prerequisite conditions, *P-Range* is the delegation range that specifies which permissions can be delegated, and *M* is the maximum delegation depth.

Crampton et al. [2] proposed a new model for dealing with transfer delegation. In transfer delegation, the delegator loses the power of using the access right after delegation is completed. They also have proposed two relations for controlling delegation. The *can-delegate* and the *can-receive* relations. The advantages of using different relations for controlling delegations include flexibility, greater control, ease of management and is less error prone. They also included constraints on the *can-delegate* and *can-receive* relations to ensure that the relations do not give the authority to a delegator to delegate a right that is not available to him.

The capability based delegation model [7] is an interesting work based on the capability based access control model (CRBAC) presented in the same paper. The CRBAC model integrates a capability-based access control mechanism into the RBAC96 model. Roles and permissions are assigned to capabilities, and capabilities are assigned to users. Delegation is achieved by creating a new capability, then by assigning delegable authority (roles or permissions) to the capability, then the delegator sends the capability to the delegatee. Unlike the other delegation models, delegation is authorized by a permission that the delegator must possess for creating the capability.

We discuss the drawbacks of the reviewed models and how OSDM addresses them in section 5.

4 The Organizational Supervised Delegation Model

We have shown in the previous sections that existing delegation models depend on the delegation relation for determining who can delegate a given role or a set of permissions. We have also shown the drawbacks of using the delegation relation for authorizing delegation. Our surveys of different organizations have revealed that no user is directly allowed to delegate his role to another user without the approval of the direct line managers of both users. Another problem with existing role delegation models is that they require that the user performing the delegation to possess the role to be delegated. This conflicts with one of the major reasons for using delegation; which is when the user is absent without early notification due to an emergency situation, and hence, blocks urgent activities usually performed by that user.

In this section, we provide our delegation model that aims at addressing issues of existing delegation models. We start by extending RBAC to enable the user hierarchy feature, that is used to find the users who must approve the delegation. Then we provide a detailed description of the delegation model followed by the formal model of the organizational supervised delegation model (OSDM). The last subsection explains how revocation is performed in OSDM.

4.1 Extensions to RBAC

In RBAC, roles are assigned to users and permissions are assigned to roles. Roles represent job functions or sub-functions in organizations, while permissions are privileges to access objects or execute operations. The main advantage of RBAC is that it simplifies the management of access rights, since users can be reassigned from one role to another. New permissions can be assigned to roles as new applications and systems are incorporated, and permissions can be revoked from roles as needed [16].

RBAC has paid attention to simulating organizational structures after the concept was originally proposed for user group structures in [17]. RBAC implements role hierarchies that are a natural means of structuring roles in organizations [18]. Role hierarchies are partial orders that express inheritance relations among roles. Although role hierarchies in RBAC can reflect some points from organizational structures, the functional role hierarchy constructed through the existing role engineering approaches does not reflect organizational structures, because they do not take into account the structural characteristics of organizations [19]. Hierarchies in RBAC implement the *is-a* relationships between hierarchical roles. In most organizations, superiors do not need full access on permissions of their inferiors. It is not necessary that a manager inherits from his inferiors since managers are often performing completely different tasks from their subordinates. In consequence, the application of the *is-a* inheritance in these situations results in the assignment of undesired and unnecessary privileges to superiors. This conflicts with the least privilege concept. In many situations, senior users have supervision relationships with junior users. Organizations are seeking flexibility when defining hierarchies in the access control model that can reflect these nuances [20]. More so, in organizational hierarchies, lines in the hierarchy mean different levels in the structure. While in RBAC it is possible that hierarchical roles are assigned to users in the same level in the hierarchy. For example, in the users structure depicted in Fig. 1, the senior developer role might inherit from the role of the junior developer, although both are in the same level in the hierarchy.

From the description above, it is clear that the role hierarchies in RBAC cannot reflect organizational structures, and therefore, we cannot take the role hierarchies as basis for modeling the organizational hierarchies. Our delegation model depends on the organizational structure in order to find the user who must approve the delegation request. It is impossible to know the manager of a given user in RBAC, since the organizational structure is not available or is misleading in RBAC. Therefore, we propose an extension to RBAC to define hierarchies over users. The hierarchy of users must respect the user hierarchies in organizational structures. In our proposed extension, users hierarchies are represented as a general tree, where each user has one parent node corresponding to his direct responsible (line manager). As an example, the structure depicted in Fig. 1 can be represented by the tree model shown in Fig. 2.

Modeling users hierarchies in RBAC with general trees makes it easier for finding the responsible (line manager) of any user, who is in charge of approving the delegation request. The direct line manager is the parent node of the user node, while the next level manager is the parent node of the direct manager node. This complies with hierarchies in organizational structures. Even in the matrix structure, the functional managers are responsible for approving task assignments to their employees. Therefore, the parent nodes in matrix structures represent functional managers.

The proposed model incurs some amendments on the administrative RBAC model [18]. The *AddUser* command now has an extra parameter called *Parent* that corresponds to the user's direct responsible. If the node to be added is a parent node (responsible), then the children nodes can be attached to this parent node as sub-trees. The *DeleteUser* is also modified. If the user node to be deleted has no children, it means that the user is not responsible for other users, then it can be deleted outright. If the user to be deleted is a parent of some other nodes, which means that he is a responsible of some other users. Then we promote the children users and attach them to the parent of the deleted node. The root node can only be replaced and cannot be deleted. Other administrative commands remain unchanged.

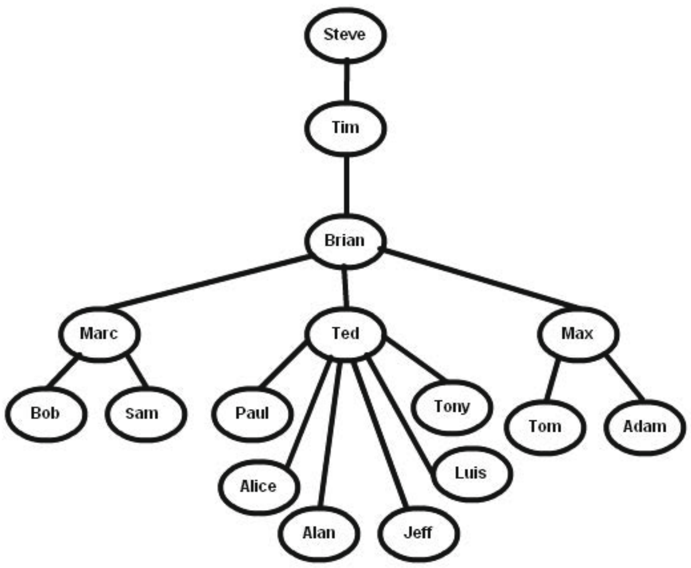


Fig. 2. Modeling the hierarchy of users of an organization structure by a general tree data structure

The extended RBAC model is composed from the following elements and relations:

- U, R, P: are sets of users, sets of roles and sets of permissions, respectively.
- $PA \subseteq P \times R$, a many-to-many permissions to roles association.
- $RA \subseteq U \times R$, a many-to-many users to roles assignments.
- $RH \subseteq R \times R$, a partial order on R. Also represented by $\geq$. If $r, r' \in R$, then $r \geq r'$ means r is higher in hierarchy than r' and that r inherits all permissions of r'.
- $UH \subseteq U \times U$, a partial order on U. Also represented by $>$. If $u, u' \in U$, then $u > u'$ means u is higher in hierarchy than u' and that u has authority on u'.

4.2 Delegation in OSDM

The central notion of OSDM is that delegation must be approved by the line managers of the delegator and the delegatee. OSDM supports both role delegation and permission delegation. More so, OSDM supports delegation in both flat and hierarchical roles. In general, the delegation process in OSDM is accomplished in three main steps: firstly, a delegation request is initiated, secondly, the delegation request is sent for approval, and finally, the delegated access rights are assigned to the delegatee.

There are three types of situations in which delegation takes place. Firstly, backup of roles. When the user is absent, the function needs to be achieved by others. Secondly, centralization of authority. When an organization needs to reorganize functions and distribute functions from higher job positions to lower job positions in the organizational structure. Thirdly, collaboration of work. Users need to collaborate with others to achieve specific tasks [6]. However, user delegation models discussed in the literature requires that the user performing the delegation must possess the ability to use the access right [2]. This is not valid in all cases, since in the backup of role case, the user is absent and cannot initiate the delegation. Therefore, we enable different users to initiate the request. Specifically, the delegation request can be initiated by three different users:

- By the delegator (the user that delegates the role). The delegator asks to delegate his role or permissions to another user.
- By the delegatee (the user that is to be delegated the access right). The delegatee asks to acquire an access right from another user.
- By the line manager of the delegator, either by the direct line manager or by a higher level line manager.

Consider the hierarchy depicted in Fig. 1, if *Alice* needs to delegate one of her access rights to *Bob*, then the delegation request can be initiated either by *Alice* herself, by *Bob*, or by one of *Alice's* line managers, usually *Ted.* In case of absence of *Ted*, the request can be initiated by *Brian*, *Tim* or *Steve.*

Once the delegation request is initiated, the delegation request becomes pending for approval. The request must be approved by a line manager of the delegator and a line manager of the delegatee. The reason for this is that the line manager of the delegator is responsible for achievement of his tasks. The approval of the line manager of the delegatee is required because he is responsible for assigning the resources to the tasks. The line managers entitled for approval receive a request to approve the delegation. This means that the delegation authorization is performed by the parent nodes of the delegator and the delegatee. Once both of them have sent their approvals, the delegation operation step can be accomplished; in which the delegated access rights are assigned to the delegatee. Usually, the delegation request is sent to the first line managers of the delegator and the delegatee, but in case of their absence, the request can be redirected on demand to the higher level line managers of the delegator or the delegatee. In case that the delegator and the delegatee are supervised directly by the same first line manager, then only one delegation request is sent to the line manager. If a user's access right is to be delegated to his direct line manager, then one approval request is sent to the line manager of the delegatee. The steps of the delegation process in OSDM are depicted in the activity diagram shown in Fig. 3.

To explain the approval step of delegation, we continue our example. When the delegation request is initiated for delegating some access rights from *Alice* to *Bob*, the request for approval is sent to both *Ted* and *Marc.* In case of absence of either *Ted* or *Marc* the delegation request can be sent to *Brian*, *Tim* or *Steve.* If *Ted* asks to acquire an access right from *Alice*, then only one request is sent to

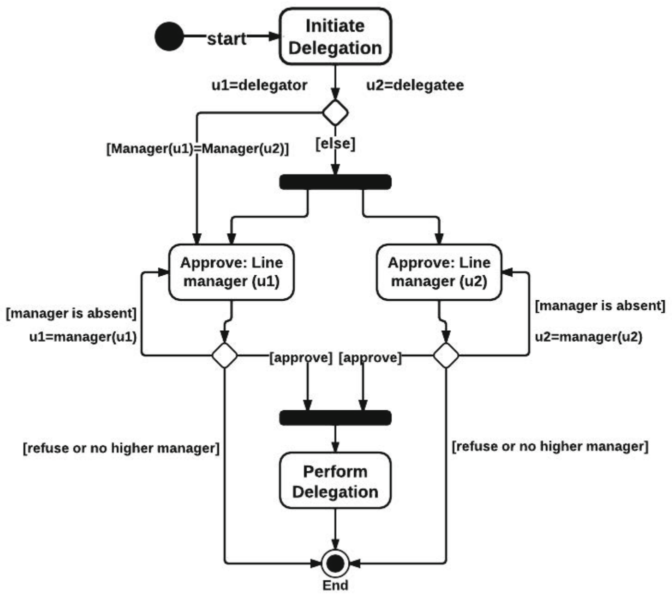


Fig. 3. The activity diagram of the delegation process

Brian for approval. If *Alice* needs to delegate some access rights to *Tony*, then only one approval request is sent to *Ted*.

Once the delegation request is approved, then the third step of delegation can be executed. The delegation operation step depends on the characteristics of delegation to be implemented. The identified characteristics of delegation comprise totality, permanence, monotonicity, administration, levels of delegation, multiple delegation, lateral agreements, cascading revocation, and grant-dependency revocation [14]. The delegation operation using these characteristics has been described by several papers [3][4][5][6][14]. In OSDM, we focus on total and partial grant delegation with flat and hierarchical roles.

We adapted the approach discussed in [3][5][6] for the delegation operation. We start with delegation operations on flat roles then we move to hierarchical roles.

Delegation in Flat Roles:

1. Grant total delegation: In this case, the delegator delegates a role with all its permissions to the delgatee. The delegatee must not be a member in that role before delegation. The delegated role is assigned to the delegatee with delegation relation instead of the original role assignment relation. This is important to identify delegated roles from roles that where originally assigned to users by the system administrators. The delegatee can start using the role after this step, and the delegator retains the power to use the delegated role.

2. Grant partial delegation: In this case, the delegator only grants a subset of permissions of a given role to the delgatee. A temporary role is created and is assigned to the permissions to be delegated. The temporary role is then assigned to the delegatee with delgation. The delegatee can start using the delegated permissions after this step, and the delegator retains the power to use the delegated permissions.

Delegation in Hierarchical Roles:

1. Grant total delegation: In this case, the delegator delegates a role with all its permissions to the delegatee. The delegatee then has the power to use the role plus all the roles in the hierarchy junior to the delegated role. The delegator is explicitly assigned to the roles junior to the delegated role. The delegated role is assigned to the delegatee with delgation and the delegator retains the power to use the delegated role.
2. Grant partial delegation: This case is exactly the same as the grant partial delegation in flat roles structure.

4.3 A UML/OCL Formal Model of OSDM

In this subsection, we include the formal model of OSDM for completeness. We use the Unified Modeling Language (UML) [21] and the Object Constraint Language (OCL) [22] to formalize the definitions of OSDM. Fig. 4. shows the UML class diagram of OSDM. The diagram projects the relationships between the different classes of the extended RBAC model according to the definitions in section 4.1. Furthermore, the diagram depicts the classes required for delegation. The *DelegationRequest* class represents the initiated delegation request. The *ApprovalRequest* represents the requests for approving the delegation request and then assigning the delegated role to the user by delegation. The diagram also depicts the two types of role assignments; the original and delegation assignments.

The following definitions formalize OSDM constraints in OCL:

Definition 1:
In OSDM, the line manager of a user is the user higher in the hierarchy (parent node). In case of absence of the direct line manager, then the line manager is the next level manager.

```
context::User:getLineManager()
post: if self.Manager.Absent = false
      then result = self.Manager
      else result = self.Manager.getLineManager()
      endif
```

Definition 2:
Self delegation is not allowed in OSDM; the delegator and delegatee cannot be the same user.

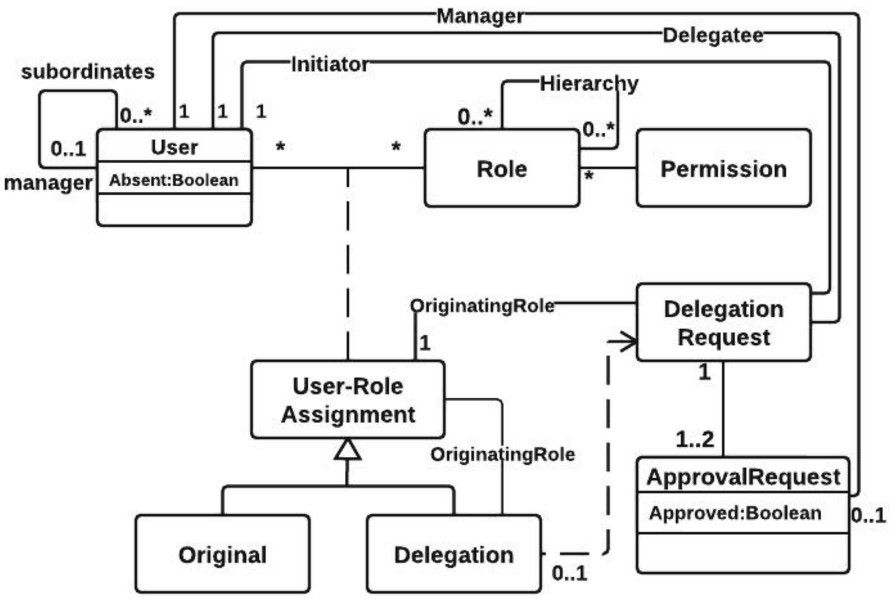


Fig. 4. The class diagram of OSDM

```
context::Delegation
inv: no_self_delegation: self.User <> OriginatingRole.User
```

Definition 3:
The role assigned to the user by the delegation relation must be exactly the role in the delegation request; which is the delegated role.

```
context::Delegation
inv: same_role: self.Role = OriginatingRole.Role
```

Definition 4:
The delegation request is initialized by the delegator, the delegatee or the line manager of the delegator.

```
context::DelegationRequest
inv: InitiatorRule: self.Initiator = self.Delegatee or
     self.Initiator = OriginatingRole.User.getLineManager()
     or self.Initiator = OriginatingRole.user
```

Definition 5:
The delegation request can be initialized only if the delegatee is not assigned to the delegated role.

```
context::DelegationRequest
inv: Delegator_not_member_in_delegated_role:
 self.delegatee.role ->asSet() ->excludes(OriginatingRole.role)
```

Definition 6:
Proper delegation requests must have been forwarded to a manager of both users involved in a delegation request.

```
context::DelegationRequest
inv: requests_initiated:
     ApprovalRequest.Manager ->
            forSome(manages(self.Delegatee)) and
     ApprovalRequest.Manager ->
            forSome(manages(self.Delegation.User))
```

Definition 7:
The role must be assigned to the user once the delegation Request is approved.

```
context::DelegationRequest
inv: delegate_role: if ApprovalRequest.Approved = true
    then delegation ->allInstances ->
    includes(delegation|delegation.role = self.OriginatingRole
    and delegation.user = self.delegatee)
    endif
```

If multi-step delegation is not allowed, then the link between *DelegationRequest* and *UserRoleAssignment* in Fig. 4 must be changed to be between *Delegation-Request* and *Original* class. The link between *UserRoleAssignment* and *DelegationRequest* indicates that the delegation request can only be initiated if the role to be delegated is assigned to the delegator.

4.4 Revocation in OSDM

Revocation is the step that ends delegation and deassigns the previously delegated access rights from the delegatee. Revocation can be performed when the reason for delegation becomes invalid. For example, if the delegation was performed because of user absence and the user returns back to duty. Revocation in OSDM is achieved as follows:

- In case of grant total delegation, the revocation is simply achieved by deassigning the delegated role from the delegatee.
- In case of grant partial delegation, revocation is achieved either by deassigning the delegated temporary role, or by revoking one or more permissions from the temporary role.

In both cases, revocation must be initiated by the delegator, the delegatee or the line manager of the delegator. The revocation must also be approved by the line manager of the delegator. The approval of the line manager of the delegatee is not necessary for revocation, since the line manager of the delegator is responsible for the accomplishment of his tasks, whereas the line manager of the delegatee is involved in cases where tasks are to be assigned to his employees.

5 Discussion

Controlling delegation is the mechanism in delegation models that determine the security of the delegation model. Although the proposed delegation control relations in existing role delegation models provide a means for authorizing delegation, they still suffer from disadvantages that will turn organizations away from using them. Defining such relations introduces complexities and is prone to error. If the definition of a relation is missing then it prevents delegating a role, while an erroneous relation enables delegation to none entitled entities. More so, relations complicate updates to RBAC, organizations need to revise the relations after each update. Even if organizations can tolerate this great overhead caused by relations, they will not be able to express some constraints on delegation such as specifying that manager m should approve delegation of role r of user u. This is due to the fact that conditions in delegation relations depend on roles. This could create inconsistencies, given it is possible that two managers having the same role becomes able to delegate a specific role. Other models such as CRBAC [7] uses a permission to authorize delegation. This eliminates complications introduced by the *can-delegate* relation, but it opens security breaches since any user that has a permission to create a capability can delegate his roles and permissions. Furthermore, role hierarchies in RBAC does not reflect organizations structures as shown in [19][20][23]. This has led to the idea of OSDM, which models organizational structures in RBAC and then utilizes them in controlling and authorizing delegation.

The advantages of OSDM other than addressing the above limitations of existing delegations, are that it complies with organizational policies towards delegation authorization. OSDM also provides a means for reflecting the organizational hierarchies and lines of authority in RBAC. In organizational structures, the functional line managers of users are responsible for resource allocation. Which means that roles are assigned to users based on the agreement of their line managers. This is also valid for delegation.

The implementation of the model is straightforward once the organizational chart is available to be projected and maintained in the access control system of the organization.

6 Conclusion

The central contribution of this paper is a new roles and permissions delegation model for role based access control, the organizational supervised role delegation model (OSDM). This model provides a new means for controlling and authorizing delegation based on the organizational hierarchy. The development of the OSDM model was motivated by surveying some organizations and verifying their delegation and role assignment mechanisms in place. The survey has concluded that such actions are usually approved by managers according to lines of authority in the organization.

The model starts by extending RBAC to adopt changes required for implementing organizational hierarchies. Mainly by adding support for users hierarchies. This enables implementing authority relations among different users by modeling the hierarchy using a general tree data structure. The users hierarchy helps in finding users who need to approve delegations and revocations according the lines of authority in the organization. In existing delegation models, delegation is authorized by using a delegation relation that define who can delegate a given role. We have explained disadvantages of this approach that could prevent organizations from using delegation models based on such relations.

The delegation request can be initiated by three different parties, the delegator, the delegatee, or the line manager of the delegator. Once the request is initiated, the delegation request is sent for approval to the line managers of the delegator and the delegatee. The delegation operation is performed when both approvals are received. OSDM supports both role and permission delegations, as well as flat and hierarchical role structures. Revocation in OSDM takes similar steps to delegation. Firstly, a request for revocation is to be initiated by the delegator, the delegatee or the line manager of the delegator. Afterwards, the revocation request needs to be approved by the line manager of the delegator, before the revocation operation is performed.

The future work on OSDM will focus on defining API classes for the model, and validation on a case study or by implementing our approach at one of the surveyed organizations. More so, we will be looking at extending the model to support parametrized roles delegation. We will be also looking at more efficient modeling of users hierarchies that can implement special lines of authorities and constraints.

References

1. Ferraiolo, D., Kuhn, D.: Role-based access control. In: Proceedings of the 15th National Computer Security Conference (1992)
2. Crampton, J., Khambhammettu, H.: Delegation in role-based access control. Int. J. Inf. Sec. 7(2), 123–136 (2008)
3. Barka, E., Sandhu, R.: A Role-based Delegation Model and Some Extensions. In: Proceedings of 23rd National Information System Security Conference, Baltimore, pp. 101–114 (2000)
4. Crampton, J., Khambhammettu, H.: Delegation in Role-Based Access Control. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 174–191. Springer, Heidelberg (2006)
5. Zhang, L., Ahn, G., Chu, B.: A Rule-based framework for role-based delegation. In: Proceedings of ACM Symposium on Access Control Models and Technologies (SACMAT 2001), Chantilly, VA, pp. 153–162 (2001)
6. Zhang, X., Oh, S., Sandhu, R.: PBDM: a flexible delegation model in RBAC. In: Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT 2003), pp. 149–157. ACM, New York (2003)
7. Hasebe, K., Mabuchi, M., Matsushita, A.: Capability-based delegation model in RBAC. In: Proceedings of the 15th ACM Symposium on Access Control Models and Technologies (SACMAT 2010), pp. 109–118. ACM, New York (2010)

8. Schermerhorn, J., Osborn, R., Uhl-Bien, M.: Organizational Behavior, 12th edn., p. 377. Wiley (2011)
9. Harris, M., Raviv, A.: Organization Design. Management Science INFORMS 48(7), 852–865 (2002)
10. Harrison, M.A., Ruzzo, W.L., Ullman, J.D.: Protection in operating systems. Communications of the ACM 19(8), 461–471 (1976)
11. Wood, C., Fernandez, E.B.: Authorization in a Decentralized Database System. In: Proceedings of the 5th International Conference on Very Large Databases, Rio de Janeiro, pp. 352–359 (1979)
12. Griffiths, P.A., Wade, B.W.: An Authorization Mechanism for a Relational Database System. ACM Transactions on Database Systems (TODS) TODS Homepage Archive 1(3), 242–255 (1976)
13. Majetic, I., Leiss, E.L.: Authorization and Revocation in Object-Oriented Databases. IEEE Transactions on Knowledge and Data Engineering 9(4), 668–672 (1997)
14. Barka, E., Sandhu, R.: Framework for role-based delegation models. In: Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC 2000). IEEE Computer Society, Washington, DC (2000)
15. Zhang, L., Ahn, G., Chu, B.: A rule-based framework for role-based delegation and revocation. ACM Trans. Inf. Syst. Secur. 6(3), 404–441 (2003)
16. Sandhu, R., Coyne, E., Feinstein, H., Youman, C.: Role-based access control models, CA, USA, pp. 38–47. IEEE Computer Society (1996)
17. Fernandez, E.B., Wu, J., Fernandez, M.H.: User group structures in object-oriented databases. In: Proc. 8th Annual IFIP W.G.11.3 Working Conference on Database Security, Bad Salzdetfurth, Germany. Database Security, VIII - Status and prospects, vol. 60, pp. 57–76 (August 1994)
18. ANSI INCITS 359, Standard for Role Based Access Control (2004)
19. Lee, H.-H., Lee, Y.L., Noh, B.-N.: A Framework for Modeling Organization Structure in Role Engineering. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 1017–1024. Springer, Heidelberg (2006)
20. Nassr, N., Steegmans, E.: ROAC: A Role-Oriented Access Control Model. In: Askoxylakis, I., Pöhls, H.C., Posegga, J. (eds.) WISTP 2012. LNCS, vol. 7322, pp. 113–127. Springer, Heidelberg (2012)
21. OMG: The Unified Modelling Language. OMG Available Spec. Version 2.2 (February 2009), `http://www.omg.org/spec/UML/2.2/`
22. OMG: Object Constraint Language. OMG Available Spec. Version 2.0 (May 2006), `http://www.omg.org/spec/OCL/2.0/`
23. Moffett, J., Lupu, E.: The uses of role hierarchies in access control. In: Proceedings of the Fourth ACM Workshop on Role-Based Access Control (RBAC 1999), pp. 153–160. ACM, New York (1999)

GPU-Acceleration of Block Ciphers in the OpenSSL Cryptographic Library

Johannes Gilger, Johannes Barnickel, and Ulrike Meyer

Research Group IT-Security
RWTH Aachen University, Aachen, Germany
{gilger,barnickel,meyer}@ITSec.RWTH-Aachen.de

Abstract. The processing power of *graphic processing units* (GPUs) has been applied for cryptographic algorithms for some time. For AES and DES especially, there is large body of existing academic work and some available code which makes use of the CUDA framework.

We contribute to the field of symmetric-key GPU cryptography by implementing and benchmarking multiple block ciphers on CUDA and OpenCL in the form of an OpenSSL cryptographic engine. We show common techniques to implement and accelerate these block ciphers (AES, DES, Blowfish, Camellia, CAST5, IDEA). Another equally important part of our work presents a guideline on how to perform reproducible benchmarks of these ciphers and similar GPU algorithms.

Keywords: GPU, Graphics Processing Unit, Block Cipher, Cryptography, Symmetric Key, AES, DES, Blowfish, IDEA, CAST-5, Camellia, OpenSSL, Performance, Benchmark.

1 Introduction

Graphic processing units have originally been designed to handle the generation and modification of graphical data. This includes rendering of 2D or 3D scenes by means of *shaders*, relatively small and limited programs to instruct the GPU's programmable rendering pipeline. As GPUs became increasingly powerful (in terms of floating point operations per second) compared to available CPUs in the last few years, the interest in using these devices for tasks other than the interactive generation of graphical output surged.

Although a small number of people had already experimented with using graphically oriented programming APIs such as DirectX and OpenGL to translate more general computation tasks to the GPU, coining the term *GPGPU* (general purpose programming for GPUs), the process was cumbersome and error-prone.

1.1 CUDA and OpenCL

CUDA (Compute Unified Device Architecture) is Nvidia's approach to GPGPU, with its initial release in 2007 aiming at the G80 chip first seen in the GeForce 8800 GTX GPU. CUDA is a programming framework which executes instructions on the CPU ("host") and the GPU ("device") and passes data between the corresponding memory spaces. Programming for the host is done using standard ANSI C, while programming for the GPU introduces some new keywords to the C language.

The *Open Computing Language* (OpenCL [15]) is a standardized GPGPU framework to enable programs to leverage the computing power of GPU and CPU devices by different vendors. The first OpenCL specification was released in 2008 by the Khronos group. In most aspects, OpenCL behaves very similar to CUDA, making the effort to port programs manageable. The important distinction is that OpenCL is a standard rather than a product, and the actual implementation is the responsibility of the respective CPU and GPU vendors.

Many recent research papers describe the acceleration of symmetric block ciphers as well as other cryptographic algorithms like hash functions, public key encryption schemes, and digital signature schemes using GPGPU frameworks. However, with regard to symmetric block ciphers, prior work focused on AES [26] and DES [8] and the CUDA framework, and only very few of these implementations were released as Open Source software.

1.2 Our Contribution

In this paper we present our implementations of the symmetric block ciphers AES [26], DES [8], Blowfish [29], IDEA [17], Camellia [22], and CAST5 [1] using CUDA and OpenCL. We chose these algorithms as they are standard encryption algorithms implemented in the OpenSSL cryptographic library [28]. The OpenSSL implementation includes thoroughly tested CPU implementations of these ciphers against which we benchmarked our own implementation efforts. We show that our GPU implementations can accelerate all block ciphers by a factor of ten (compared to CPU implementations) for practical application scenarios. The fact that we implemented the CUDA and OpenCL ciphers as an OpenSSL engine, based on the engine-cuda project [21], makes these ciphers available to software which already employs OpenSSL with relatively little effort. While the original engine-cuda project included a capable AES implementation for CUDA, we show that our improved AES implementation is almost twice as fast.

In addition, we present guidelines for benchmarking cryptographic and other programs with CUDA and OpenCL. We deduce these guidelines from our own experience with the implementation and various shortcomings of the benchmarking presented in related work (see Section 2).

The rest of the paper is structured as follows: Section 2 summarizes previous work on GPU implementations of symmetric ciphers and their integration in frameworks and applications. The results of our implementation efforts are presented in Section 3. Section 4 summarizes our lessons leading into a guideline for developing and evaluating cryptographic algorithms on GPU platforms.

2 State of the Art

Cryptographic algorithms have been ported to graphics hardware even before general purpose programming tools existed. Using libraries like OpenGL and

DirectX, symmetric key algorithms such as AES have successfully been run and accelerated on GPU devices [6], [10], [31].

With the introduction of general purpose computing on GPUs (GPGPU), the number of publications on implementations of cryptographic algorithms on graphics hardware surged.

2.1 Symmetric Key Ciphers

For symmetric key algorithms, AES is the block cipher most often investigated. The earliest publication using CUDA to accelerate AES was by Manavski [20]. Manavski closely followed the Rijndael reference implementation, as did almost every team after him. He identified the optimal location for storing the T-tables and provided impressive benchmark performances, measuring raw kernel execution time as well as overall data rate, compared to an OpenGL implementation of AES. In 2008, Harrison et al. released their work with AES-CTR and CUDA GPUs [11]. Aside from similar performance benchmarks, their work also discussed general purpose data structures for scheduling serial and parallel execution of block ciphers on GPUs. Luken et al. implemented and benchmarked the AES and DES [19] on an NVIDIA Tesla C870 GPU, a specifically designed GPGPU card. In some cases, AES was sought to be parallelized below the block-level, and this approach was compared to the more common block-level parallelism in [4], which also proposed a different distribution for T-tables in shared memory in order to avoid memory access conflicts. A textbook implementation of AES for the OpenCL framework was given in 2010 by Gervasi et al. [9]. This implementation did not make use of the T-tables and consequently suffered from unsatisfactory performance. Coincidentally, this publication also employed the OpenSSL framework to integrate their GPU implementation, with the goal of ultimately making it available to a broad audience by having it included in the official OpenSSL development tree. Other work on AES includes [23], which closely examines the different memory locations for the T-tables, correctly points out the low ratio of computation vs. memory access of AES and provides benchmarks which reportedly are able to reach the limit imposed by the memory bandwidth on an Nvidia GeForce 9200M.

DES, which is deprecated for new cryptographic systems, has been implemented on GPUs as well. Yang and Goodman published an AES and DES key breaker realized using DirectX and AMD's Close To Metal framework [32]. Another publication about a DES software breaker, this time using CUDA, was released in 2010 by Agosta et al. [3]. The cost of breaking DES using CUDA-enabled GPUs was compared to the COPACOBANA system [16], which is a special-purpose multi-FPGA DES breaker, and found to perform in the same order of magnitude. Noer et al. have announced that they have improved the results of [3], with a 20-fold performance advantage of searching DES keys on a GPU compared to a CPU [27]. They are furthermore planning to port their efforts to OpenCL.

The CAST5 block cipher was implemented as part of an GPU-based dictionary attack on OpenPGP secret keyrings by Milo et al[24]. In their paper, they present a very detailed and well-written explanation of how they implemented their CAST5 key searcher. The optimization techniques described in their paper apply equally well to similar software and will likely provide noticeable performance improvements.

A CUDA implementation of the Serpent block cipher was presented in [25]. Li et al. investigated MD5-RC4 encryption on GPUs [18].

2.2 Integration and Frameworks

Some researchers investigated GPU cryptography in terms of the framework needed to actually integrate it into existing systems. In 2010, Jang et al. introduced their implementation of a reverse SSL-proxy called *SSLShader* [13], which is able to perform the necessary crypto-operations (RSA, AES128-CBC, HMAC-SHA1) for a large number of independent SSL connections by using the GPU starting from a certain threshold, where the latency incurred by the GPU is lower than queueing the operations on the CPU. Recently, Jang et al. built on their previous efforts and released a paper with a detailed account of how they implemented their SSLShader which provides accelerated AES, RSA and HMAC-SHA1 using a combination of GPU and CPU techniques [14].

Harrison and Waldron also investigated how well NVIDIA GPUs can be used to provide a kernel-level service for cryptographic operations [12]. Using the OpenBSD cryptographic framework (OCF) and a userspace daemon they were able to provide GPU-cryptography for kernel- and userspace processes with minimal overhead. The kgpu project [30] also investigates the feasibility of a kernel-level GPU framework using a userspace daemon which to dispatch CUDA operations. To eliminate the latency of kernel invocations, they introduce the concept of a *non-stop kernel* (NSK) which continuously runs on the GPU and can be instructed by passing messages through allocated memory. Their implementation is demonstrated by accelerating eCryptfs with AES using the GPU.

Agosta et al. investigated how GPGPUs can be integrated into the *full-disk encryption* software *TrueCrypt*, using the XTS block mode and Twofish as the block cipher [2].

2.3 Practical GPU-Accelerated Software

Much of the work using GPU software and cryptographic algorithms has been of a strictly academic nature. However, there do exist some end-user software products which have incorporated GPU components to accelerate their functionality.

Hash algorithms and symmetric ciphers in particular can be found in a number of proprietary consumer software products aimed at breaking password protected files using brute force, or finding hash collisions in general. These implementations therefore do not provide facilities to actually use the GPU for the normal application of the cryptographic algorithm, and the nature of the software forbids opening the source code to competing companies. Another popular use of GPU hardware is the generation of *rainbow-tables*, data-structures for pre-computed hashes that provide a tradeoff between storage space and computation time to find a hash collision.

Open Source software employing cryptography has seen relatively few patches to use GPU devices. The engine-cuda project [21] by Paolo Margara is a notable exception to the large number of proprietary software, as it set out to create an Open Source portable library to be used with the popular OpenSSL crypto library which provides GPU-accelerated versions of the algorithms included in the standard library. This project was chosen as the foundation for implementing the algorithms presented in this paper.

2.4 Benchmarking Methodology

To benchmark their implementations of cryptographic algorithms on GPU devices, the involved researchers used a variety of different metrics, which makes it hard to compare the merits of the approaches with one another. While some teams solely focused on the theoretical speed by counting the instructions needed to perform the operations [5], others went a step further and implemented working GPU-versions of these algorithms. For symmetric ciphers, some implemented the key-scheduling on the GPU [9], [14], while others used the CPU for this task [11], [20]. Benchmarking of the implementations differed as well, with some teams only measuring instruction throughput (presumably using the fastest available on-card memory), others doing calculations to and from the GPU main memory [19], and yet other teams measuring the whole computation chain starting from the host to the device and back to the host [19], [20].

While the approach of *microbenchmarking*, only measuring the operation on the GPU itself, is useful when designing and optimizing the algorithm itself, it should not be used to compare the performance to a CPU-implementation, as it does not represent the performance available for encrypting data stored in global memory of the host system.

2.5 Benchmarking Problems

Much of the previous work suffered from minor flaws or omissions in the description of the benchmark process used.

Often, the specific CPU implementation used for comparison was not mentioned, and parameters such as number of CPU cores and compilation were omitted as well. In one case, the performance for the CPU was measured using a Java implementation of the block-cipher in question [7].

Some publications did not clearly state which time interval they measured in order to compute performance results. Also the exact timing method (using either system timers or GPU-specific timing facilities) was often not mentioned, and we suspected this to be a source of error as well. Almost every team used a different GPU, making a direct comparison of the results all the more difficult.

The GPU benchmarks including the host-section of computations often did not state whether the data was already stored in (DMA-accessible) allocated host-memory or if it had to be fetched from background memory first, including the costly memory allocation of a large block. The structure of the payload data also makes a significant difference in performance, since it can affect which memory locations need to be accessed during encryption. Only one team provided a description of the payload data used for their benchmarks.

3 Implementation and Benchmarking Results

To implement and evaluate several popular block ciphers, we used the existing engine-cuda [21] codebase as a foundation for our implementation efforts. The original project was implemented as an OpenSSL engine (i.e. a shared library) and already included the AES for CUDA in ECB and CBC mode [26] for all key sizes. We implemented the following popular block ciphers:

- **AES** [26]: AES-128, AES-192, AES-256, 128 bits blocksize
- **DES** [8]: 64 bits keysize, 64 bits blocksize
- **Blowfish** [29]: 128-448 bits keysize, 64 bits blocksize
- **IDEA** [17]: 128 bits keysize, 64 bits blocksize
- **Camellia** [22]: 128, 192 and 256 bits keysize, 128 bits blocksize
- **CAST5** [1]: 128 bits keysize, 64 bits blocksize

All of these algorithms were implemented with the electronic codebook (ECB) block mode for encryption and decryption, as well as the cipher block chaining (CBC) mode for decryption. We did not implement CBC encryption on the GPU, as the CBC block mode cannot be parallelized during encryption and would not benefit from the GPU platform. Other block modes which can be computed in parallel (e.g. Counter mode (CTR)) could use our implementations with only minimal modifications and a small constant overhead. We limited ourselves to ECB and CBC as these are the block modes available in the current OpenSSL packages.

Since the original engine-cuda project only supported AES with CUDA, we had to significantly extend it to support OpenCL and other block ciphers as well.

To the best of our knowledge, the Blowfish, IDEA, Camellia and CAST5 implementations represent the first published CUDA and OpenCL versions of these algorithms. Although there have been previous efforts to implement AES and DES using CUDA, none of these were readily available to be benchmarked against our implementation, as none of them were Open Source.

3.1 Optimizations

We started implementing each block cipher by porting the implementation included in OpenSSL to CUDA and OpenCL. After we ensured basic functionality, we began to optimize the block cipher for these GPU platforms in terms of performance. Most importantly, we looked at register usage per GPU thread, uncoalesced memory access and diverging paths, trying to optimize each aspect using a variety of techniques, some of which could certainly be classified as "compiler work-arounds":

1. Removal of unnecessary flexibility. For example, we implemented separate GPU kernels for ECB, CBC and the different key lengths rather than passing these as parameters to a single kernel. In general, we passed as few parameters as possible, e.g. by storing data such as T-Tables in one big chunk which can be referenced using a single pointer. This resulted in reduced register use, enabling us to better utilize the SMs of the GPU.

With BF_KEY structure

```
__shared__ BF_KEY bs;
__device__ BF_KEY bsg;

if(TX < 18)
  bs.P[TX] = bsg.P[TX];

bs.S[TX] = bsg.S[TX];
bs.S[TX+256] = bsg.S[TX+256];
bs.S[TX+512] = bsg.S[TX+512];
bs.S[TX+768] = bsg.S[TX+768];
```

Disregarding BF_KEY structure

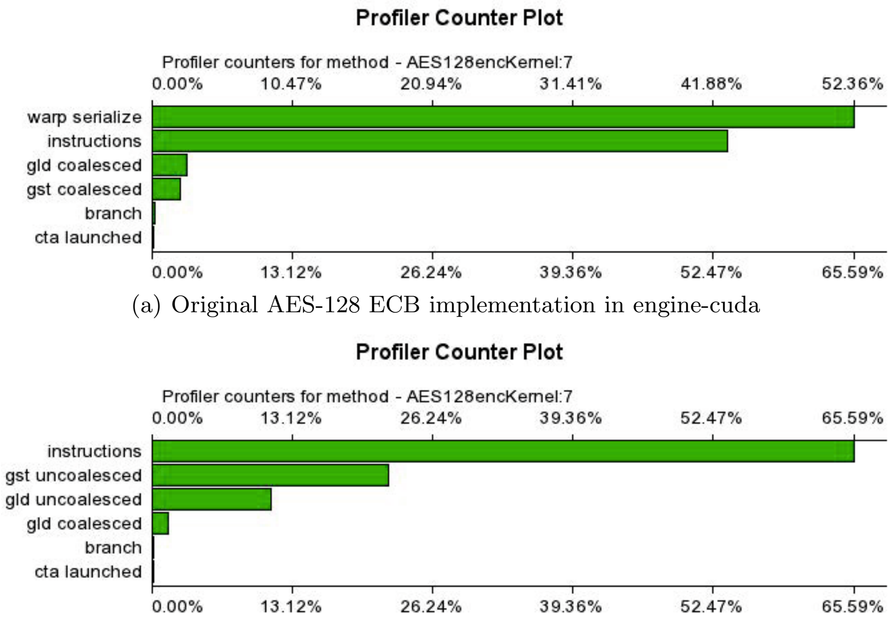
```
__shared__ uint32_t bs[1042];
__device__ uint32_t bsg[1042];

bs[TX] = bsg[TX];
bs[TX+256] = bsg[TX+256];
bs[TX+512] = bsg[TX+512];
bs[TX+768] = bsg[TX+768];

if(TX < 18)
  bs[TX+1024] = bsg[TX+1024];
```

Fig. 1. Misaligned and aligned memory copy of the Blowfish key schedule

2. Removing uncoalesced memory access which would otherwise require multiple memory transactions. Figure 1 shows how we were able to get rid of all the misaligned memory accesses for the Blowfish cipher with a simple change. Sometimes, independent 32-bit values were fetched from memory into a 64-bit variable and split up manually, so that the compiler would generate a single coalesced `ld.global.u64` instruction instead of two strided `ld.global.u32` instructions.
3. Removal of bank conflicts when using shared memory. This was the case for the AES version included with the original engine-cuda codebase. Figure 3 shows the time spent due to warp serialization for the original and our improved AES implementation.
4. Reordering statements to reduce register use. In some cases, statements could be reordered without influencing the computation. A variant of this approach was to re-use variables for different steps of the computation.
5. Modification of the key schedule to avoid endian conversion of the payload data, as many algorithms were developed with Big-Endian CPU architectures in mind.
6. Use of native integer functions and synchronization. We used the preprocessor heavily to employ functions only available on specific compute capabilities, such as native 24-bit or 32-bit multiplication. A simple example is given in Figure 2.
7. Unrolling loops to reduce register use. Unrolling is a technique often performed by CPU compilers, which we had to perform manually for CUDA, for example for the encryption rounds.

```
#ifndef TX
  #if (__CUDA_ARCH__ < 200)
    #define TX (__umul24(blockIdx.x,blockDim.x) + threadIdx.x)
  #else
    #define TX (blockIdx.x * blockDim.x + threadIdx.x)
  #endif
#endif
```

Fig. 2. Preprocessor macro for calculating the thread-ID `TX`



(a) Original AES-128 ECB implementation in engine-cuda

(b) Our improved AES-128 ECB implementation for engine-cuda

Fig. 3. The CUDA Visual Profiler showing the original engine-cuda (Figure 3(a)) and our improved version of AES-128 ECB (Figure 3(b))

Table 1 shows how much resources are used by each kernel of our implementations of the block-ciphers as reported by `nvcc`. For better comparability, the table lists the *decryption* kernels for ECB and CBC. Register use relates to each thread, while the shared memory is counted block-wise and the constant memory is reported for the whole kernel, following the visibility of each class of memory.

Depending on the compute capability the kernels are compiled for, `nvcc` will use more or fewer registers for each kernel, up to twice as many registers for our newer (CC 1.3) cards compared to the older card (CC 1.1).

Optimizations were also performed in the backend, which includes the memory transfer functions, the setup of the GPU context and the memory allocation. It is important to mention that we did not implement any stage of key-scheduling

Table 1. Memory consumption for the implemented decryption kernels in engine-cuda (compiled for CUDA Compute Capability 1.3)

Cipher	Mode	Registers	Shared Memory	Constant Memory
AES-{128,192,256}	ECB	13 / 14	4376 bytes	264 bytes
	CBC	15	4384 bytes	264 bytes
Blowfish	ECB	10	4176 bytes	8 bytes
	CBC	12	4184 bytes	8 bytes
DES	ECB	9	2056 bytes	136 bytes
	CBC	10	2064 bytes	136 bytes
CAST5	ECB	10	4104 bytes	144 bytes
	CBC	12	4112 bytes	144 bytes
Camellia-128	ECB	14	4104 bytes	296 bytes
	CBC	14	4112 bytes	296 bytes
IDEA	ECB	10	224 bytes	216 bytes
	CBC	12	232 bytes	224 bytes

on the GPU but left that task to be performed by the CPU. We ensured that our GPU implementations worked correctly by comparing their output to the stock OpenSSL CPU output for a variety of different payload data and keys.

3.2 Benchmark System

We thoroughly benchmarked the resulting implementation of each algorithm on two systems with different CPUs and GPUs, using an older GeForce 8600 GT as well as a more recent GeForce GTX 295. The results presented in this paper are based on the benchmark system described in Table 2.

Table 2. Benchmark system

CPU	Intel Core i7 960 3.20GHz
GPU	GeForce GTX 295 (CC 1.3)
RAM	12 GB DDR3 RDIMM
HDD	Intel X-25 M II SSD (160GB)
Kernel	Linux 3.0.0-17-generic x86_64
CUDA	CUDA toolkit 4.1
Driver	NVIDIA UNIX x86_64 285.05.33
CC	GCC 4.4.6

The results for our second system were omitted due to space constraints and will be provided as part of an extended version of this paper. The GeForce GTX 295 in the benchmark system contains two discrete GPUs on two different PCBs (printed circuit boards), each of which is presented as an independent GPU to the system. We only used one of these GPUs for our benchmarks. CPU reference speeds were obtained using OpenSSL v1.0.1 and using one of the four CPU cores of the benchmark system. We measured the theoretical kernel execution speed, which does not take into account the time needed to transfer the data to and from the GPU, as well as measuring the practical encryption speed, which includes transfers from and back to host memory. In the following, we discuss the results of both measurement techniques.

3.3 Impact of Payload on Kernel Benchmarks

The *microbenchmarks* of our cipher kernels are shown in Table 3[1] and Table 4. These tables measure the theoretical ECB performance of the cipher kernels, which only includes the execution time of the CUDA (resp. OpenCL) kernel itself. The resulting time is given in milliseconds, from which the theoretical performance (given in megabytes per second) can be derived. To demonstrate the importance of the payload data, we measured performance with pseudo-random data (obtained from `/dev/urandom`) and zero-bytes and used Δ to denote the speed ratio of zero- over random bytes.

Table 3. ECB encryption kernel performance for different input data (8192 KB, GeForce GTX 295)

Cipher	Engine	Random bytes		Zero bytes		Δ
		Kernel ms	MB/s	Kernel ms	MB/s	
AES-128	CUDA	2.16	29613	1.36	47021	1.59
	OpenCL	2.52	25408	1.72	37248	1.47
AES-192	CUDA	2.57	24922	1.58	40461	1.62
	OpenCL	2.94	21760	1.98	32384	1.49
AES-256	CUDA	2.97	21572	1.87	34234	1.59
	OpenCL	3.35	19072	2.23	28736	1.51
Blowfish	CUDA	2.12	30249	1.49	43077	1.42
	OpenCL	30.07	2112	5.34	11968	5.67
Camellia-128	CUDA	2.43	26302	1.66	38647	1.47
	OpenCL	2.43	26304	1.68	38016	1.45
CAST5	CUDA	2.19	29203	1.54	41519	1.42
	OpenCL	2.45	26112	1.79	35648	1.37
DES	CUDA	4.14	15460	2.75	23279	1.51
	OpenCL	4.12	15488	2.74	23296	1.50
IDEA	CUDA	1.75	36512	1.71	37388	1.02
	OpenCL	1.60	40064	1.59	40128	1.00

The payload data is used as to determine the memory access for certain operations (such as an index in a lookup table). When the payload consists only of uniform bytes, the same memory area will be queried for every byte of payload data, which results in excellent performance for constant memory (which is cached after the first memory access). However, when the payload data exhibits a certain amount of entropy, different memory areas will be queried in

[1] We achieved slow results for Blowfish OpenCL because we had to work around an apparent compiler bug which forced us to use slower memory.

each request. In this case, shared memory shows a clear advantage over constant memory.

No previous publication measured and compared both payload types and some of the previous work did not even mention at all which payload they used.

3.4 Improved AES Implementation

Table 4 shows our improved AES-128 ECB implementation benchmarked against the implementation which was already included with engine-cuda. Originally, AES was implemented using a "fine-grained" approach, which employs four threads for each cipher block of 128 bits. Our "coarse-grained" approach uses just one GPU thread for each cipher block. As expected in [4], our coarse-grained approach delivered a noticeable performance increase. We also measured the speed of storing the AES T-Tables, which represent the AES functionality in the form of static lookup tables, in constant and in shared GPU memory. The results are nearly identical when zero bytes are used as payload data, while random data (which is much more practically relevant) shows the clear superiority of using shared memory. This obvious discrepancy prompted our investigation into the benchmark methodology of other teams, leading to the advice given in Section 4.

Table 4. Fine- and coarse-grained AES-128 ECB (8192 KB, CUDA)

Approach	T-Table	Random bytes		Zero bytes		Δ
		Kernel	MB/s	Kernel	MB/s	
Fine	Constant	15.86	4036	3.54	18097	4.48
	Shared	3.44	18584	3.53	18138	0.97
Coarse	Constant	14.35	4460	1.45	44123	9.89
	Shared	2.16	29613	1.36	47021	1.58

3.5 OpenSSL Benchmarks

To evaluate the practical benefit of using GPU-accelerated block ciphers, we also measured the performance using the OpenSSL `speed` command, which schedules runs of increasingly large blocks of zero-byte data and measures the achieved throughput. These results were averaged over five consecutive runs. This kind of benchmarks includes the time needed to transfer the payload from the host to the device memory, the cipher kernel execution, and the time for transferring the data back to the host, and as such can be directly compared to the execution time on the CPU. Figure 4 shows the results of GPU and CPU benchmarks for CUDA and the ECB mode of each block cipher. It is obvious that because of the latency incurred by invoking a GPU kernel and the necessary memory transfer, GPU-accelerated block ciphers can only outperform the CPU if the payload data is large enough, larger than 16KB in our case.

Although we also implemented the CBC-decryption, we chose to use the ECB mode as the primary tool for benchmarking as it can be more easily compared

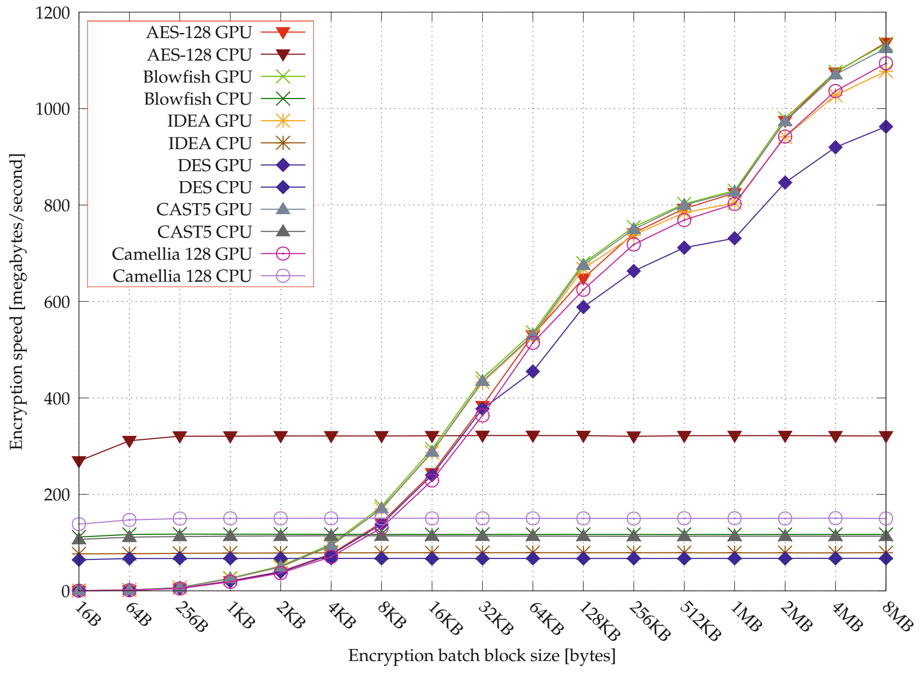


Fig. 4. ECB encryption with CUDA on one PCB of a GeForce GTX 295 (OpenSSL `speed`)

to existing and new academic work with the same algorithms. The CBC mode in our implementation only adds a small performance penalty in the form of an additional memory access and does not invalidate the performance results compared to the CPU, as shown in Figure 5.

We intentionally refrain from making statements concerning the performance relative to existing work for the reasons stated in Section 4. We want to emphasize that the 8-10 fold improvement in speed shown in Figure 4 was actually measured on our test system, but that the superiority of the GPU heavily depends on the GPU generation, the CPU and the application area.

Other features of our extended GPU crypto library include timing facilities for CUDA and OpenCL, benchmarking tools for trying different kinds of payload data, keys and payload size and different levels of debug output.

4 Performing Reproducible Benchmarks

During the process of implementing the cryptographic algorithms presented in this work, we gathered a number of benchmarking details which are important for the production of reproducible comparable benchmark results. In the following, we provide an overview on the details as guideline for future implementation efforts, not only in the are of cryptography.

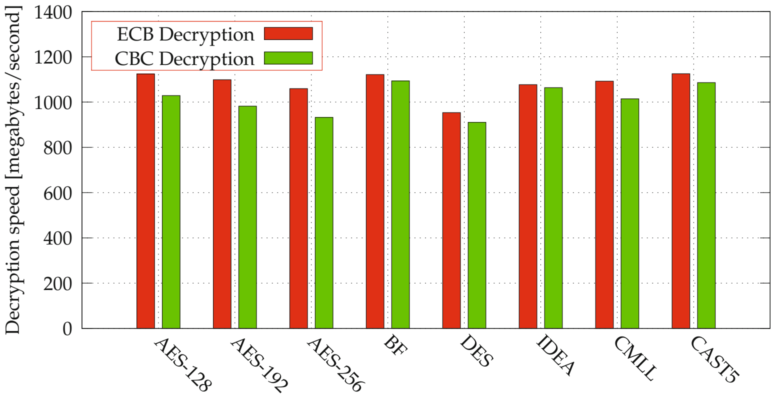


Fig. 5. ECB and CBC decryption on 8MB blocks (CUDA, GeForce GTX 295, OpenSSL `speed`)

Kernel and Framework. Depending on the framework, host operations and the time for memory transfers dwarfs the actual kernel execution time. Any research which does not only focus on the raw kernel should therefore include benchmarks of both the kernels and the complete chain of operation.

Structure of Payload. When benchmarking the speed of a block cipher on a GPU, it is important to mind the kind of data being used. While zero bytes are the obvious choice for ease of generation and reproducibility, they can result in artificially good result with unknown average-case behaviour. Block ciphers should be developed using zero bytes as well as random data, and publications should clearly indicate the source of payload data. Especially the behaviour with random data should influence the choice of storing read-only lookup tables.

Ensuring Correctness. If block ciphers are implemented on GPU platforms, the correctness of the cipher should be ensured at every step of the development process. GPUs are hard to debug, so a combination of different keys, key sizes, payload structure and payload length should be tested at all times. Especially payload sizes which are not multiples of the thread block configurations are often subject to failure. For platforms like OpenCL, tests for correctness should be run repeatedly, since problems do occur non-deterministically.

Measuring Time. CUDA and OpenCL include their own timing functions for the simple reason that kernel calls are nonblocking and may return before the kernel is finished.

Scheduling Priorities. CUDA has an option to use spinning to wait on the return of the GPU kernel, which for many repeated small invocations can make a measurable difference. Understanding and verifying the method to poll for the GPU kernel is important.

Eliminating Side Effects. When performing benchmarks, the GPU should be switched to compute exclusive mode and any running X server should be stopped. Benchmarking cryptography means allocating large blocks of page locked memory on the host, so the system should have enough free RAM. The CPU should not be busy with other tasks except executing the host-thread. It is important to use the GPU driver supplied by the vendor alongside the GPGPU framework. Other drivers might work correctly as well but perform much worse when used with CUDA or OpenCL.

Using Reference Implementations. When directly comparing the GPU to existing CPU implementations, an established and publicly accessible CPU implementation of the block cipher should be used. It is also important to verify whether the reference implementation uses more than one CPU core, whether it uses native instructions tailored to the CPU platform and if special instructions like AES-NI are employed.

5 Conclusion

In this work, we showed the potential and limitations of GPU-accelerated block ciphers as implemented within the OpenSSL cryptographic library. We were able to clearly accelerate symmetric block ciphers using GPUs compared to traditional CPU implementations, in terms of theoretical as well as practical speed. As the first contribution we used the exact same setup to implement these ciphers using CUDA and OpenCL. Perhaps not surprisingly, we were able to show that both GPU frameworks are equally capable of delivering practical performance for all of the implemented block ciphers.

In addition to the implementation and benchmarking we discuss common problems encountered when trying to compare the benchmarking results of different implementations of symmetric ciphers. From this discussion, we compile and present a list of recommendations for future implementations.

We hope that our paper and the future release of our source-code to the engine-cuda project can assist in presenting correct benchmark results in this and other GPU-related research areas, with the ultimate goal of raising the scientific standard of similar work and to further the cause of open standards within the GPGPU community.

References

1. Adams, C.: The CAST-128 Encryption Algorithm. RFC 2144 (Informational) (May 1997)
2. Agosta, G., Barenghi, A., Santis, F.D., Biagio, A.D., Pelosi, G.: Fast Disk Encryption through GPGPU Acceleration. In: PDCAT, pp. 102–109. IEEE Computer Society (2009)
3. Agosta, G., Barenghi, A., Santis, F.D., Pelosi, G.: Record Setting Software Implementation of DES Using CUDA. In: ITNG, pp. 748–755. IEEE Computer Society (2010)

4. Biagio, A.D., Barenghi, A., Agosta, G., Pelosi, G.: Design of a parallel AES for graphics hardware using the CUDA framework. In: IPDPS, pp. 1–8. IEEE (2009)
5. Bos, J.W., Stefan, D.: Performance Analysis of the SHA-3 Candidates on Exotic Multi-core Architectures. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 279–293. Springer, Heidelberg (2010)
6. Cook, D.L., Ioannidis, J., Keromytis, A.D., Luck, J.: CryptoGraphics: Secret Key Cryptography Using Graphics Cards. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 334–350. Springer, Heidelberg (2005)
7. Daniel, T.R., Mircea, S.: AES on GPU using CUDA. In: 2010 European Conference for the Applied Mathematics & Informatics. World Scientific and Engineering Academy and Society Press (2010)
8. N. I. for Standards and Technology. Data Encryption Standard (DES). NIST FIPS PUB 46-3 (1999)
9. Gervasi, O., Russo, D., Vella, F.: The AES Implantation Based on OpenCL for Multi/many Core Architecture. In: International Conference on Computational Science and its Applications, pp. 129–134 (2010)
10. Harrison, O., Waldron, J.: AES Encryption Implementation and Analysis on Commodity Graphics Processing Units. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 209–226. Springer, Heidelberg (2007)
11. Harrison, O., Waldron, J.: Practical Symmetric Key Cryptography on Modern Graphics Hardware. In: USENIX Security Symposium, pp. 195–210. USENIX Association (2008)
12. Harrison, O., Waldron, J.: GPU accelerated cryptography as an OS service. Transactions on Computational Science 11, 104–130 (2010)
13. Jang, K., Han, S., Han, S., Moon, S., Park, K.: Accelerating SSL with GPUs. In: SIGCOMM, pp. 437–438. ACM (2010)
14. Jang, K., Han, S., Han, S., Moon, S., Park, K.: SSLShader: Cheap SSL acceleration with commodity processors. In: Proceedings of NSDI 2011 (2011)
15. Khronos OpenCL Working Group. The OpenCL Specification, version 1.1 (September 30, 2010)
16. Kumar, S., Paar, C., Pelzl, J., Pfeiffer, G., Schimmler, M.: Breaking Ciphers with COPACOBANA –A Cost-Optimized Parallel Code Breaker. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 101–118. Springer, Heidelberg (2006)
17. Lai, X., Massey, J.L.: A Proposal for a New Block Encryption Standard. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 389–404. Springer, Heidelberg (1991)
18. Li, C., Wu, H., Chen, S., Li, X., Guo, D.: Efficient implementation for MD5-RC4 encryption using GPU with CUDA. In: Proceedings of the 3rd International Conference on Anti-Counterfeiting, Security, and Identification in Communication, pp. 167–170. IEEE Press (2009)
19. Luken, B.P., Ouyang, M., Desoky, A.H.: AES and DES encryption with GPU. In: ISCA PDCCS, pp. 67–70. ISCA (2009)
20. Manavski, S.A.: CUDA compatible GPU as an efficient hardware accelerator for AES cryptography. In: International Conference on Signal Processing and Communications, ICSPC 2007, pp. 65–68. IEEE (2008)
21. Margara, P.: engine-cuda (2011), http://code.google.com/p/engine-cuda/ (accessed April 25, 2012)
22. Matsui, M., Nakajima, J., Moriai, S.: A Description of the Camellia Encryption Algorithm. RFC 3713 (Informational) (April 2004)

23. Mei, C., Jiang, H., Jenness, J.: CUDA-based AES parallelization with fine-tuned GPU memory utilization. In: IPDPS Workshops 2010, pp. 1–7 (2010)
24. Milo, F., Bernaschi, M., Bisson, M.: A fast, GPU based, dictionary attack to OpenPGP secret keyrings. Journal of Systems and Software 84(12), 2088–2096 (2011)
25. Nazlee, A.M., Hussin, F.A., Ali, N.B.Z.: Serpent encryption algorithm implementation on Compute Unified Device Architecture (CUDA). In: IEEE Student Conference on Research and Development (SCOReD), 2009, pp. 164–167. IEEE (2010)
26. NIST. Advanced Encryption Standard (AES). National Institute of Standards and Technology (2001)
27. Noer, D., Engsig-Karup, A., Zenner, E.: Improved Software Implementation of DES Using CUDA and OpenCL
28. OpenSSL. OpenSSL: The Open Source toolkit for SSL/TLS (2011), http://www.openssl.org/ (accessed April 25, 2012)
29. Schneier, B.: Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In: Anderson, R. (ed.) FSE 1993. LNCS, vol. 809, pp. 191–204. Springer, Heidelberg (1994)
30. Sun, W.: kgpu: Augmenting Linux with the GPU (2011), http://code.google.com/p/kgpu/ (accessed April 25, 2012)
31. Yamanouchi, T.: AES encryption and decryption on the GPU. In: Nguyen, H. (ed.) GPU Gems 3, ch. 36. Addison Wesley Professional (August 2007)
32. Yang, J., Goodman, J.: Symmetric Key Cryptography on Modern Graphics Hardware. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 249–264. Springer, Heidelberg (2007)

A Highly-Efficient Memory-Compression Approach for GPU-Accelerated Virus Signature Matching

Ciprian Pungila and Viorel Negru

West University of Timisoara,
Blvd. V. Parvan 4, Timisoara 300223, Timis, Romania
{cpungila,vnegru}@info.uvt.ro
http://info.uvt.ro/

Abstract. We are proposing an approach for implementing highly compressed Aho-Corasick and Commentz-Walter automatons for performing GPU-accelerated virus scanning, suitable for implementation in real-world software and hardware systems. We are performing experiments using the set of virus signatures from ClamAV and a CUDA-based graphics card, showing how memory consumption can be improved dramatically (along with run-time performance), both in the pre-processing stage and at run-time. Our approach also ensures maximum bandwidth for the data transfer required in the pre-processing stage, between the host and the device memory, making it ideal for implementation in real-time virus scanners. Finally, we show how using this model and an efficient combination of the two automata can result in much lower memory requirements in real-world implementations.

Keywords: gpu, gpu-accelerated, cuda, commentz-walter, aho-corasick, wu-manber, memory efficient, virus scan, malicious code detection.

1 Introduction

The problem of efficient virus detection has had a lot of attention over the past few years and, as the numbers of viruses grew exponentially, is now one of the most computationally-intensive tasks found in intrusion detection systems. With the performance boost that technology and, in particular, graphics cards have shown lately, the solution has been improved dramatically, by a good factor of times compared to normal CPU operational speed.

In essence, the problem of virus detection is related to the challenge of efficient pattern matching (given that there are multiple patterns in the database that need to be matched), both in terms of memory usage and processing time. Aho et al[1] were the first to propose an algorithm for exact multiple pattern matching (Aho-Corasick, or A-C) that is fast and runs in linear time. A-C is optimal in the worst case, however, as Boyer-Moore have shown in [2], in practice there are situations when the running time can be improved by jumping over redundant characters, an idea that was later on studied by Commentz-Walter in [3].

The result was an algorithm (Commentz-Walter, or C-W) that is not optimal in the worst case, but that can produce the same results in sublinear time in the average case. Better performance had been obtained by Wu et al in [4] (the Wu-Manber algorithm, or W-M) after implementing a hashing approach that can significantly reduce the number of overall operations performed. We relate in this paper to ClamAV[5], one of the most commonly used open-source antivirus programs today.

This paper focuses on performing memory and time-efficient virus signature matching in real-world systems by taking advantage of the massively parallel processing capabilities of GPUs, and proposes a highly-efficient memory-organization and virus signature scanning model that helps achieve these goals without compromising performance. Furthermore, we show how using our approach, an efficient combination of the A-C and C-W algorithms can result in much lower memory requirements. We discuss related work in section II, presenting the CUDA architecture, the A-C, C-W and W-M algorithms along with the ClamAV virus signature format and its implementation, while in section III we present our own approach and the proposed model and discuss it in depth. Section IV shows the experimental results we have obtained with a variety of implementation scenarios performed, both on the GPU and on the CPU.

2 Background

2.1 The CUDA Architecture and Programming Model

For this paper we have focused on the feasibility of adding GPU-accelerated virus scanning support to antivirus software in generic consumer hardware. We have used a Core i7 CPU and a CUDA-based (PCI-E 2.0) GTX 560Ti 1GB DDR5 graphics card, with the engine clock running at 900MHz, shader clock at 1800MHz, memory clock at 4200MHz and having 384 CUDA cores. The Compute Unified Device Architecture (CUDA) is described in [7] and is the basis for programming all CUDA-based GPU cards, offering an abstraction of the underlying hardware through additional libraries that can be easily called from C code. For our implementation, we have used Visual Studio 2010 and CUDA version 4.1.

The GTX 560Ti card is based on the GF114 chip and has two processing clusters with 4 multiprocessors each. Each multiprocessor consists of 48 stream cores and is accompanied by 8 texture-processing units (see Figure 1). Stream processors are operating on SIMD (Single Instruction Multiple Data) programs. CUDA programming allows the GPU to be used as a massively parallel execution machine, supporting a very high number of threads. Programs issued by the host computer to the GPU are called kernels and are executed on the device as one or more threads, organized in one or more blocks. Each multiprocessor executes one or more thread blocks, those active containing the same number of threads and being scheduled internally. It is possible to manage shared memory and the host can access global, constant and texture memory (constant and texture memories are cached, so therefore faster).

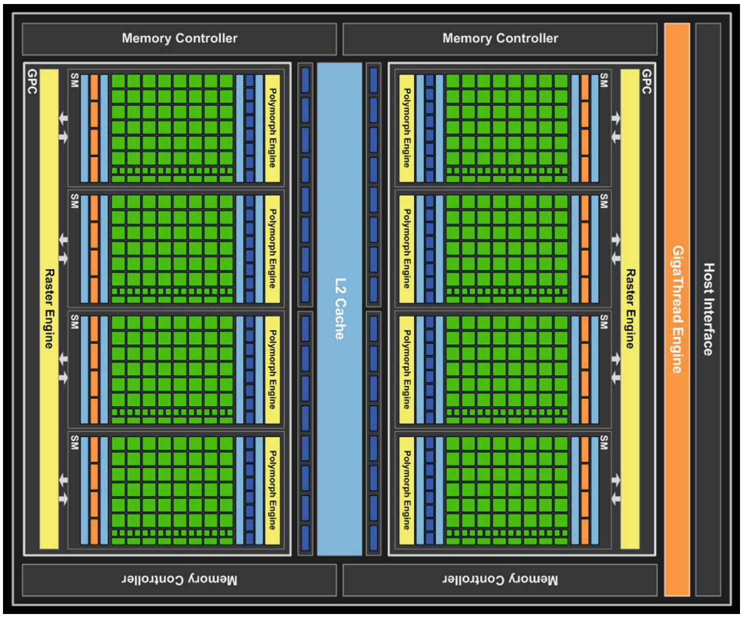


Fig. 1. The GF114 chip layout

2.2 Multiple Pattern Matching

The Aho-Corasick algorithm is based on the trie tree concept and is in fact a deterministic finite automaton (DFA) that has, at each level, a failure pointer associated to the node, which gets called in case of a mismatch in the DFA at the current state. The failure, for the root of the trie, as well as for all the nodes at the first level in it, is always pointing back to the root. For the other levels of the trie, the failure is being computed as being the longest possible suffix of the word (at the current node in the trie) that is also a word in the tree (see Figure 2) - if no such suffix exists, the failure points back to the root.

The Commentz-Walter algorithm however uses a different approach, based on the Boyer-Moore shift approach from simple pattern matching. It creates a trie tree from the set of reversed patterns, and assigns to each node two shift values, *shift1* and *shift2* and two additional functions, *out* and *char*. The C-W performance heavily relies on the length of the smallest keyword in the tree (w_{min}). *shift1* is being computed as the smallest between w_{min} and (if it exists) the distance between the current level of the current node (v) and all superiors levels of the other nodes, that have as suffix the word formed at the initial node v. *shift2* is computed similarly, except that in this case we are computing the smallest between w_{min} and (if it exists) the distance between the current level of the current node (v) and all superiors levels of the leaf nodes that have as proper suffix the word formed at the initial node v. *out* returns the pattern for the current node, if its word is a keyword in the tree, while *char(a)* is computed

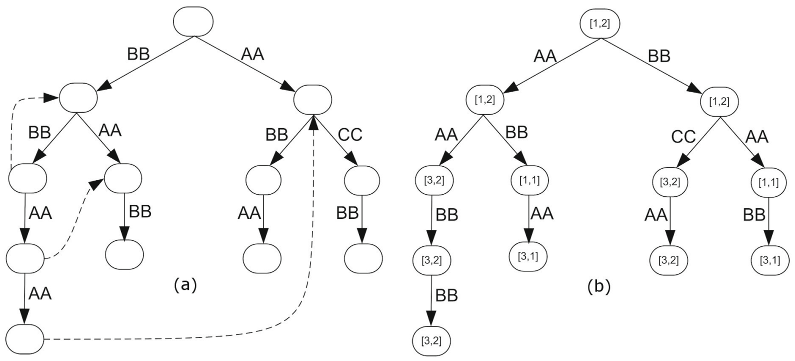


Fig. 2. a) The Aho-Corasick tree for the input set {BBBBAAAA, BBAABB, AABBAA, AACCBB}. Sample failures are marked dashes. b) The Commentz-Walter automaton for the same input set (includes distances as [shift1, shift2]).

as being the minimum of w_{min}*+1* and the level of all nodes having a as label. The processing begins with a right-side comparison starting at position w_{min}; in case of a mismatch, the current position at node v is increased by *min*{*shift2(v), max*{*shift1(v), char(text[pos-j])-j-1*}}, where *pos* is the current position and j indicates the depth in the tree. Our experiments have shown that the *char* function is always 1 for the set of signatures used in ClamAV, which is why the shifting distance for C-W becomes *min*{*shift1(v), shift2(v)*}.

The Wu-Manber algorithm uses three tables built in the pre-computation stage, called *shift*, *hash* and *prefix*. The *shift* table is similar to the one precomputed for the Boyer-Moore bad-character skip table, while the other two are only used when *shift* table indicates not to shift (since there may be a potential match at the current position). Just like Commentz-Walter and Boyer-Moore, the performance highly depends on the length of the shortest pattern. The algorithm looks at blocks of text, and the authors suggest a size of 2 for short patterns and 3 for longer ones, leading to significant memory-requirements.

Numerous applications of multiple pattern matching have been proposed for intrusion detection systems: an A-C implementation based on the ClamAV signature format is proposed in [8], another A-C-based approach for detecting malicious code through system-call heuristics is proposed in [9], while in [10] bloom filters are used to speed-up the scanning process. Lin et al[11] have observed that the W-M approach can be used successfully on longer patterns from the ClamAV database, while A-C can still be used for regular-expression matching with shorter patterns, however their approach is lacking parallelism and is still relying on the performance of the A-C machine, and although W-M performs much better for longer patterns, A-C remains the primary bottleneck.

2.3 ClamAV and GPU-Accelerated Virus Scanning

Cha et al in [12] have observed that 95% of the scanning time of ClamAV is focused on matching the regular expressions, although their number is very small (only about 16% of the number of signatures). ClamAV's A-C tree is limited in depth, as shown in Figure 6a [13]. The majority of the ClamAV signatures (specifically, about 96.6% in the version of the *main.cvd* ClamAV database we used) uses three types of constraints: {*n*-}, for specifying at least *n* characters, {*n*-*m*} (for specifying between *n* and *m* characters) and {-*m*} (for specifying at most *m* characters). The remaining signatures could be easily transformed into this type of constraints, since for instance *?* can be replaced with {*1-1*}, *** with {*0,*-} and subexpressions of type {ABCD(EE|FF)00} can be distributed into separate patterns as follows: {ABCDEE00} and {ABCDFF00}. In our our tests, from the 62,302 signatures we had used for testing from the ClamAV database, only 6,831 contained regular expressions (about 11%).

A GPU-based pattern matching approach for virus scanning has been presented by Vasiliadis et al in [6] as GrAVity, with significant improvements over the performance of a single-core CPU, by limiting the depth of the Aho-Corasick tree used for scanning. Their approach however uses significant amount of memory (each node in their implementation uses 1KB, which leads to 400 MB of memory for 400,000 nodes). Tuck et al have presented a compression-approach for the A-C automaton through path-compression in [14] for the Snort network intrusion detection system[15], improved even further through bitmapped nodes by Zha et al in [16].

2.4 Motivation of Our Work

The motivation of our work is based on two important aspects for malware detection: the first is that static signatures are still the only completely accurate way of detecting malicious code (and according to antivirus manufacturers they will not disappear anytime soon, with so many thousands of viruses in the wild), and the second is that the process of exact matching signatures is usually the slowest, compared to dynamic heuristics. The recent improvements in hybrid CPU/GPU solutions have shown that GPU-based solutions outperform greatly CPU-based ones, however the main problem remains the data transfer between the host and the device, a problem we aim to solve in this paper. For real-world implementations, an antivirus based on GPU acceleration should offer a few important advantages: low-memory footprint, good performance and WDDM TDR-compliance.

The ClamAV implementation was modified to allow GPU-processing in GrAVity[6], however the approach used a very inefficient implementation of the DFA since every node was using 1 KB of data (given that at each node, there are a maximum of 256 possible transitions, and that each transition is a pointer, that will lead to at least 4*256 =1,024 bytes per node). That leads to 400 MB of usage for the 400,000 nodes that the authors had obtained when using a tree limited to depth 14. While Tuck et al[14] have shown how memory requirements can be improved through path-compression using Snort signatures[15] (they had replaced consecutive nodes having a single child with one single compressed node), an approach which was also used by Zha et al in [16] for reducing memory usage even further through bitmapped nodes, a real performance bottleneck in GPU-based implementations is also the way the memory is allocated (see Figure 3).

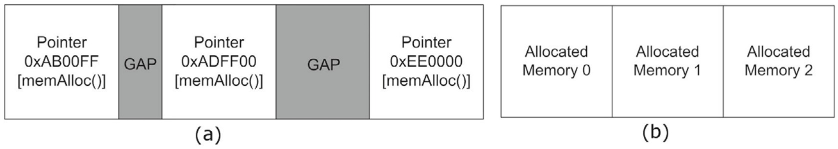


Fig. 3. a) Sparse, consecutive memory allocations and gaps that appear in GPU memory; b) A single, sequentially allocated memory block in the GPU device

Vasiliadis et al with MIDeA[17] have presented an architecture for intrusion detection which uses a compacted state table representation, that only stores for each state the elements between the lower and the upper limits. While this approach does offer improvements over the one presented in [6], it does still store additional redundant information if the states are sparse enough and a low number of children exist for each state. In Gnort[18], the authors present a solution based on Snort for implementing network IDS using GPUs, however the actual implementations use the default Snort approaches (full, sparse, banded, trie, etc.), still for all of them there is room for improvement in terms of space efficiency. PFAC[19] is an open-source library that implements string matching on the GPU that applies perfect hashing to the parallel failureless A-C, building one or two-level perfect hashing for efficient storage, at the expense of processing time. The final aim of PFAC is to build a perfect hash that stores only valid transitions in the automata, however the process required to compute and build the hash is laborious and quite expensive (as the authors observe, the *mod* operation is expensive to compute on the GPU), plus the actual memory requirements may be higher (authors propose an inequality for determining the upper bound for space usage, while in our approach the space usage is always constant).

ClamAV uses a depth-limited A-C tree (with a depth of 2, since the smallest virus signature contains a subexpression of just 2 bytes) along with an implementation of W-M. The W-M algorithm however uses large amounts of memory because of the shift and hash tables. In our implementation, for a block size of 3, we obtained $2^{24} = 16.7$ million elements in the hash table, or, considering that two 32/64-bit integer (we used 64-bit in our implementation) values are stored into each element of the table, a total amount of 64 MB or 128 MB memory, not counting additional structures needed.

In order for code that runs in the GPU through the CUDA architecture to access the data it needs, this needs to be transferred from the host (the RAM memory) to the device (in the GPU memory). The device to host bandwidth can reach high peaks given the PCI-E 2.0 x16 architecture, however the bandwidth is higher as the memory block being transferred is larger. That means that for small transfers, such as repeated 1 KB transfers, the transfer is highly inefficient. There is also a problem of the extra space being left-over as a gap by the library-based memory allocation routines, for small-footprint allocations. In order for the GPU to be able to take on the matching tasks from the CPU, it must have access to the memory area that needs to be scanned or passed through the A-C DFA, along with the data structures required to perform the processing. Sparse, repeated small-size memory allocations will create unwanted gaps between memory locations, which may significantly increase the memory usage, even though that may not be desired or expected. Although it may be possible that future allocations will fit the gaps left-over, there is no guarantee that it will happen eventually.

In the case of DFAs, where the number of nodes can reach several millions, this is becoming a real problem as it can create huge memory losses. The second problem with repeated allocations is that they consume a large amount of time - in our tests, copying all DFA nodes from the host device to the GPU device took over 2 hours for about 350,000 nodes. Of course, the ideal approach is to copy large areas of memory at once and benefit from the full PCI-Express bandwidth. This poses additional challenges since tree-based structures are generally comprised of unevenly distributed pointers throughout memory, simply creating a linear list from these pointers will not suffice, as all data structures must also update their internal pointers to accommodate the new locations. Our approach using multiple queues will achieve a perfectly linear, gap-free distribution of nodes, as we will show in Section III. Using our model, the performance of parsing the automata is not compromised and the space storage required is always linear in S, the total number of states.

3 Implementation

We used 62,302 signatures from the ClamAV database for our tests. We have implemented the A-C, a constraint-based A-C, C-W and W-M algorithms and we began testing on the CPU, then moved on to the GPU. The constraint-based A-C is an adaptation of the A-C algorithm to support the three types of constraints that ClamAV signatures contain. For the CPU implementations, we used path-compression and bitmapped nodes for the A-C and C-W automatons (Figure 4 shows the data structure formats used during implementation), to minimize memory usage, while for the GPU part we have built our own memory model for aligning the automaton so that we can maximize the host-to-device bandwidth when copying it into the GPU. We used a data-parallel approach for performing scanning on the GPU.

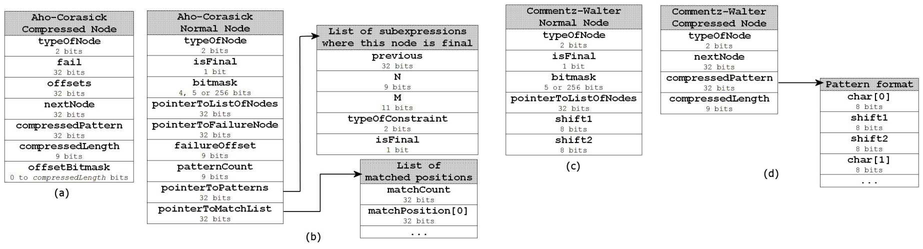


Fig. 4. a) A-C compressed nodes; b) A-C normal nodes in the constraint-based automata supporting RegEx signatures; c) C-W normal nodes; d) C-W compressed nodes

Figure 5 *b, c, d* shows results for single-core CPU benchmarks of individual algorithms, while Figure 5*f* shows the results obtained when attempting to implement a dual-scanning system (first, AC-AC used A-C in one thread for scanning RegEx signatures and again A-C in the other for scanning regular signatures, and the second, AC-CW used A-C in one thread for scanning RegEx signatures and C-W in the other for scanning regular signatures). The results obtained showed that AC-CW is performing faster than AC-AC, however given that C-W performs best when the shifting distance is higher, we have split the set of signatures into two parts: the first, containing regular expressions and patterns shorter in length than *d*, and the second containing patterns longer than *d* (where *d* was chosen as 16, 32, 64, 96 and 128) (Figure 5a shows the signature distribution after the split, showing how many signatures were scanned by the RegEx signature scanner, and how many by the regular signature scanner in the different scenarios). We also tested a two-thread scanner comprised of A-C for RegEx scanning and W-M for regular scanning (referred to as AC-WM), and results in Figure 5e show that memory usage is much lower in the AC-CW approach compared to AC-WM.

In GrAVity, the authors limit the depth of the tree in order to reduce memory usage[6]. This may produce false positives, but once a potential match is detected an extra step is employed at the end to fully verify that a full match is present. It is also stated that for a depth-limited tree of 8, the number of false positives when scanning is less than 0.0001%, which is why in our approach we have also used the same depth for our tree (however we also performed memory and bandwidth benchmarks for depths 12 and 16). Our tests were performed on 50 MB of randomly generated binary data and we have used 128 threads per block (a 66% occupancy rate offered the best results after performing initial testing with 96, 128, 160 and 192 threads) and 8,192/4,096/1,024 blocks (leading to a total of 1,048,576/524,288/131,072 threads) for each kernel running.

3.1 The Constraint-Based Aho-Corasick Automata

We have implemented the default A-C approach and also a constraint-based version, where we added basic support for the three types of constraints found in the ClamAV signatures. The process is performed as follows: a regular expression signature is being parsed and trimmed into its subexpression parts (e.g. *11AA22{8-}33FF{-12}DDEE* will insert in the tree, separately, the patterns *11AA22*, *33FF* and *DDEE*), with pointers existing from the leaf of a subexpression to the leaf of the subexpression preceding it. Whenever a leaf of any subexpression is reached, the leaf of the previous subexpression is checked - if all leaves were visited, this is considered a partial match and is checked at the end for a full match. Our constraint-based version of A-C ensures accurate matching by adding match position arrays at every leaf of every subexpression, and verifying constraints on the spot whenever a partial match is found (Figure 6b).

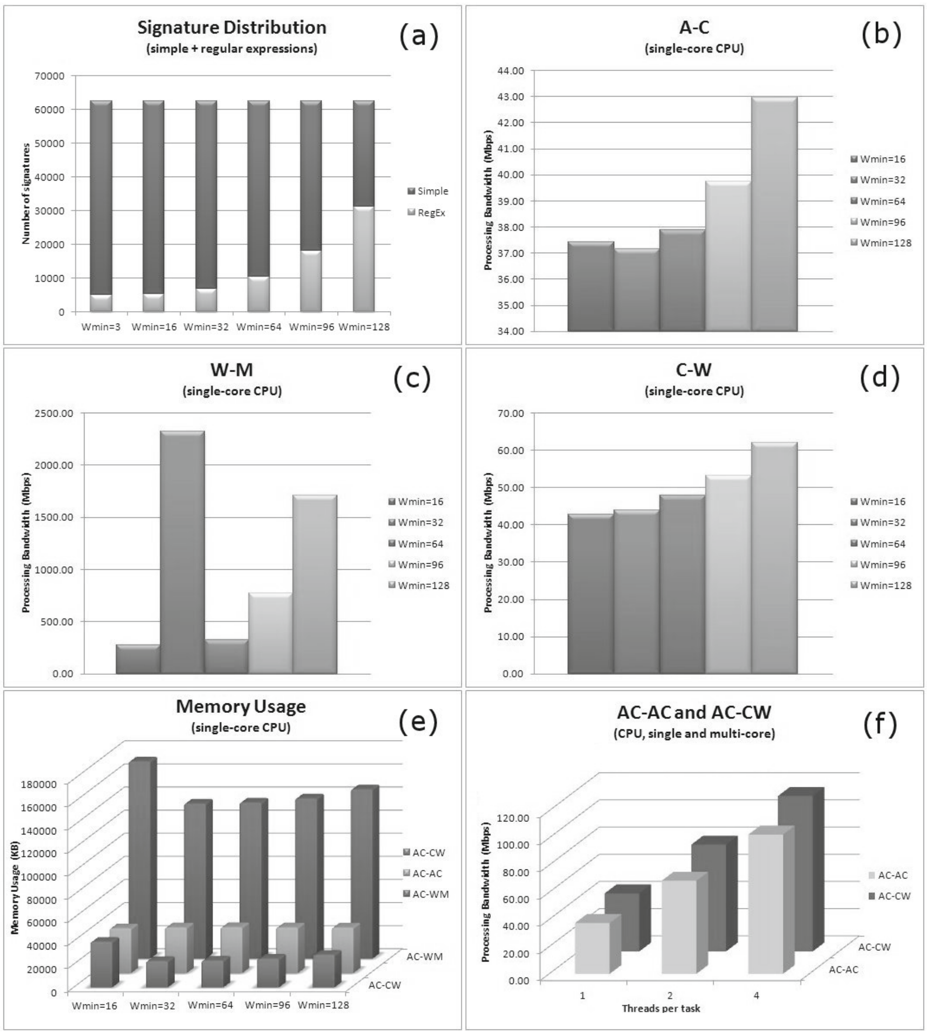


Fig. 5. Signature distribution when performing dual-scanning and single-core CPU performance results for tested algorithms

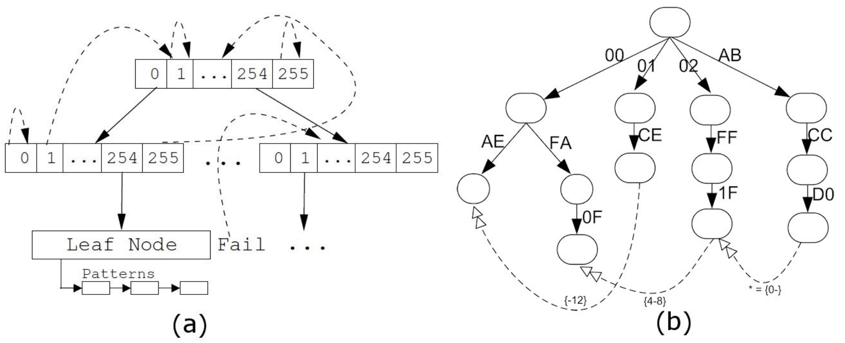


Fig. 6. a) ClamAV implementation of the A-C algorithm [13]; b) The constraint-based A-C automata for the input set {00FA0F{4-8}*02FF1F*ABCCD0, 00AE{-12}01CE}. Failure transitions are not shown.

3.2 Our Proposed Memory Compression Model

We have used stacks for rebuilding the automaton, replacing the pointers that exist in the usual data structures with offsets in the stacks pointing to the elements of interest. Basically, we are serializing the tree in a continuous, gap-free memory block, but using offsets in the stacks for locating elements, instead of pointers. For the standard A-C and C-W versions we use two stacks, *nodes*, *offsets*: the first points to the list of serialized nodes, stored as structures, while the second points to a list of offsets (Figure [7]). The *offset* member of the node structure indicates to the position where all the children of that node are stored - the number of children is given by the population count of the bitmap in the structure. The model proposed always stores all children of a node in a sequential manner in the stack, while the offset in the stack corresponding to the first child is stored in the parent node. Therefore, the child at index i in a node n is accessible through a single reference: *nodes[offsets[n->offset+i]]*.

The constraint-based automata uses a similar approach and a third stack, *links*, which holds structures that contain an offset to a node in *nodes* (this offset indicates the leaf of a previous subexpression in a regular expression pattern).

Building the A-C/C-W automata and preprocessing it is not part of the virus scanner itself, since database signatures are usually delivered (through the update mechanism) in a ready-to-be-used form. Given that our automata is one continuous block of memory, the transfer between the host and the device is the fastest possible. The algorithm we used to build the node stack for our model is presented as follows:

– *Initialization*

```
1: $topOfNodeStack \leftarrow 1$ (top of node stack)
2: $currentPosition \leftarrow 0$ (position in the stack)
3: $topOfOffsetStack \leftarrow 0$ (top of the stack of offsets)
4: $node$ (the currently processed node)
```

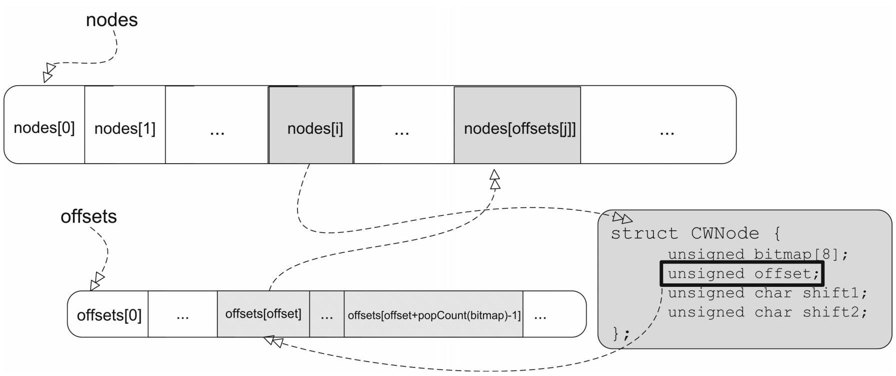


Fig. 7. The stack-based memory model used for efficient storage of the C-W automata in the GPU. A similar model is used for the basic A-C implementation.

– *function addNode(node, currentPosition)*

```
1: $nodeStack[currentPosition] \leftarrow node$
2: $nodeStack[currentPosition].offset \leftarrow topOfOffsetStack$
3: add to hash ($key \leftarrow node$, $value \leftarrow currentPosition$)
4: $pc \leftarrow popCount(node.bitmap)$
5: **for** $i \leftarrow 0$ to $pc - 1$ **do**
6:     $offsetsStack[topOfOffsetStack] \leftarrow topOfNodeStack + i$
7:     $topOfOffsetStack \leftarrow topOfOffsetStack + 1$
8: **end for**
9: $old \leftarrow topOfNodeStack$
10: $topOfNodeStack \leftarrow topOfNodeStack + pc$
11: **for** $i \leftarrow 0$ to $pc - 1$ **do**
12:     addNode($node.child[i]$, $old + i$)
13: **end for**
```

To further reduce memory usage, the *offsets* stack can be completely discarded, assuming that offsets are stored in the node itself. The code for serializing the automata while using a single stack of nodes is the following:

– *Initialization*

```
1: $topOfNodeStack \leftarrow 1$ (top of node stack)
2: $currentPosition \leftarrow 0$ (position in the stack)
3: $node$ (the currently processed node)
```

– *function addNode(node, currentPosition)*

```
1: $nodeStack[currentPosition] \leftarrow node$
2: $nodeStack[currentPosition].offset \leftarrow topOfNodeStack$
3: add in hash ($key \leftarrow node$, $value \leftarrow currentPosition$)
4: $pc \leftarrow popCount(node.bitmap)$
5: $old \leftarrow currentPosition$
```

```
6: $topOfNodeStack \leftarrow topOfNodeStack + pc$
7: for $i \leftarrow 0$ to $pc - 1$ do
8:     addNode($node.child[i]$, $old + i$)
9: end for
```

In order to efficiently restore the pointers in the new structures, a hash is created in the first step that holds a list of the nodes in the automaton, together with their new corresponding offset in the GPU memory-representation. A second processing step is then employed, which parses the GPU tree in pre-order and restores pointers by locating them in the hash and replacing them with the offset. While path-compression and bitmapped nodes may be applied to the automata at this stage to further reduce the memory usage on the GPU, there are a few elements that suggest this option should be avoided in general: first, memory rellocation in the GPU is a time-consuming process (memory manipulation routines, such as *memmove()* and *realloc()* in C, have significant impact over the overall performance of the running thread) and second, coding complexity greatly reduces, therefore reducing performance penalties in the GPU.

A simple theoretical analysis shows that in order to implement, using our model and bitmapped nodes, a basic Aho-Corasick pattern matching automata with an alphabet of size Σ and a total of N nodes/states, the total space required for storage is $N \times (2log_2N + \Sigma)$ bits (for each node, the following was considered: one failure offset to indicate the new state in case of a mismatch, using log_2N bits, the offset pointer to indicate to the list of children for the current node which is also occupying log_2N bits, and a bitmapped representation of the alphabet occupying Σ bits). Comparing this result to PFAC[19], where the AC-Compact approach used by the authors for 1,703,023 nodes occupied 24.18 MB (about 15 bytes per node, assuming an alphabet size of $\Sigma = 32$), our approach (without any performance penalty, unlike AC-Compact) would require a storage space of only $2 \times 21 + 32 = 47$ bits per node, or 10 bytes per node and a total of 15.02 MB of memory (1.6 times less memory) to store the complete automata in memory.

4 Experimental Results

We have conducted a series of tests for evaluating the performance of our approach on the GPU using two types of tests: single-scanning, where the whole input data set was scanned using the A-C and constraint-based A-C machine (Figure 8), and dual-scanning, where we have used the A-C and AC-CW approaches for performing scanning in parallel with regular expressions (in the A-C machine) and simple patterns (through C-W, see Figure 9).

In Figure 8a, the total number of nodes obtained after applying the depth-limited implementation of our model (we used three different depths of 8, 12 and 16 for testing various scenarios) is directly proportional to the amount of memory used by the A-C automata in Figure 8b. The GPU throughput of both the standard A-C and the constraint-based A-C used in RegEx scanning (in Figure

8 c, d), for the three different depths shows that the default A-C works about twice as fast as the constraint-based A-C. RegEx scanning is mainly limited by the large number of small signatures that get matched partially, since ClamAV's smallest signature is comprised of only 2 bytes. In particular, for a depth of 8, when using the basic A-C machine, our automata was using only 18.63 MB of memory for about 352,921 nodes, compared to the approach used in GrAVity where the amount was about 345 MB (almost 19 times lower memory usage). However, given that the A-C approach we had used in this test was affected by the existence of many nodes that included redundant information (such as the one used for leaves in subexpressions of regular expressions), the AC-CW approach (which was using even less memory for the C-W automata than the equivalent A-C) consumed only 14.75 MB of memory (lowest amount in our test) for w_{min}=32, compared to almost 315 MB in the previous solution, leading to almost 22 times lower memory usage in this scenario.

The last step of our experiment involved implementing the dual-scanning system on the GPU, based on the observations made *a-priori* that using a combination of the Aho-Corasick and Commentz-Walter automata produces the lowest memory usage, while maintaining the throughput similar to that of the Aho-Corasick RegEx scanning approach alone. In our experiment, in one kernel ran the A-C algorithm and in the other the C-W machine. We limited the depth of the automata at a distance of 8, as discussed before, and the number of total nodes for A-C (running in kernel 1) is shown in Figure 9a (we used different test scenarios involving different values for W_{min} for determining the best compromise between memory usage and throughput performance), while the remaining

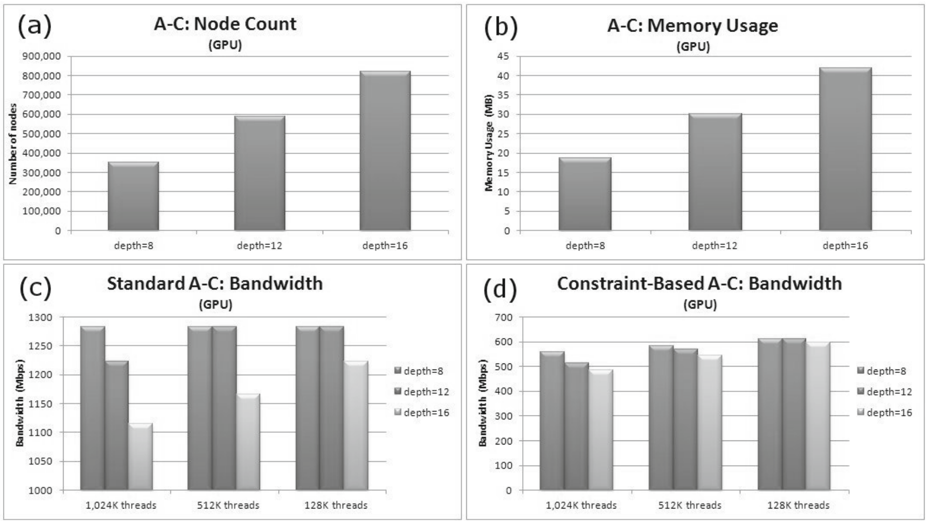


Fig. 8. Results obtained during the experimental phase when using a single A-C kernel and the entire experimental set of ClamAV signatures

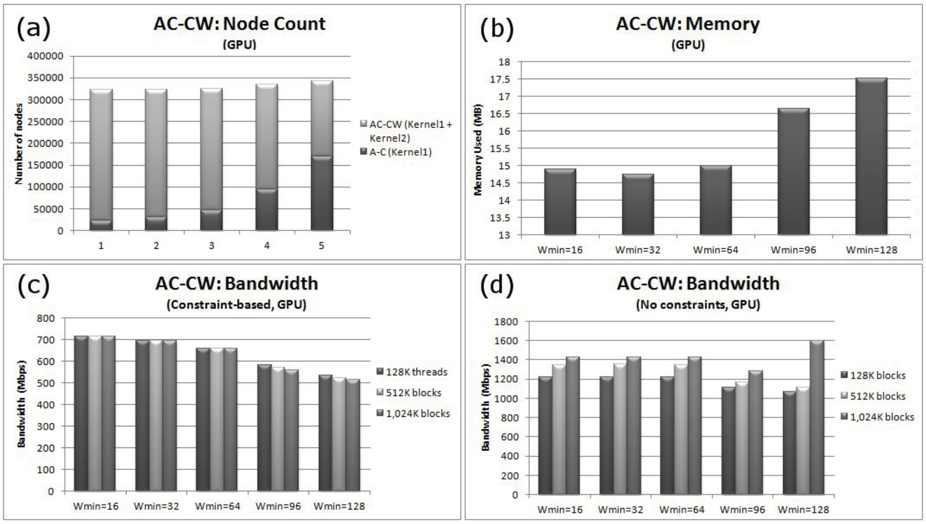


Fig. 9. Results obtained during the experimental phase when using dual-scanning with one A-C kernel scanning for regular expressions and an AC-CW kernel for scanning simple patterns

nodes were being used by regular pattern scanning in kernel 2. The memory used by the GPU implementation is shown in Figure 9b and one can see that while W_{min} increases, the total storage required also increases, since the number of RegEx signatures increases (the data structures used in RegEx scanning occupy of course more space than those used in regular signature matching). Figure 9c shows performance obtained when using RegEx scanning, while Figure 9d shows performance obtained when dropping the RegEx scanning entirely and only performing regular A-C matching in kernel 1. The bandwidth reached 1,420 Mbps when using 1 million threads (almost 34 times faster than the AC-CW throughput on the CPU), for $W_{min} = 128$, but at the expense of memory usage (about 17.5MB required). The best compromise in terms of the $\frac{Bandwidth}{Memory}$ ratio is offered by the AC-CW implementation for $W_{min} = 16$ and $W_{min} = 32$, both having a ratio value of 48.

5 Conclusion

We have proposed a highly-efficient memory compression model for implementing GPU-accelerated virus signature matching and have tested it on the GPU, in single and dual-scanning modes. Experimental results have shown that our model is ideal for implementation in real-time monitoring systems based on GPU hardware acceleration. Tests performed have revealed high performance improvements over previous known attempts, using almost 22 times less memory than related implementations used in GrAVity, while achieving up to 38 times higher bandwidths than single-core CPU implementations. In our experiment, the most memory-efficient approach for implementing virus signature matching relied on dual-threaded scanning, where one thread was scanning regular expressions through the Aho-Corasick machine, and the other thread was scanning regular patterns using the Commentz-Walter algorithm.

Future work includes a GPU-based, WDDM TDR-compliant driver model for performing real-time virus scanning as part of the Windows operating system.

Acknowledgements. This work was partially supported by the grant of the European Commission FP7-REGPOT-CT-2011-284595 (HOST), and Romanian national grant PN-II-ID-PCE-2011-3-0260 (AMICAS). Special thanks go to Sean Baxter and Tim Murray at NVIDIA Corp. for their valuable help and insight on the CUDA architecture.

References

1. Aho, A., Corasick, M.: Efficient string matching: An Aid to blbiographic search. CACM 18(6), 333–340 (1975)
2. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Communications of the ACM 20, 762–772 (1977)
3. Commentz-Walter, B.: A String Matching Algorithm Fast on the Average. In: Maurer, H.A. (ed.) ICALP 1979. LNCS, vol. 71, pp. 118–132. Springer, Heidelberg (1979)
4. Wu, S., Manber, U.: A fast algorithm for multi-pattern searching. Technical Report TR-94-17, University of Arizona (1994)
5. Clam AntiVirus, http://www.clamav.net
6. Vasiliadis, G., Ioannidis, S.: GrAVity: A Massively Parallel Antivirus Engine. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID 2010. LNCS, vol. 6307, pp. 79–96. Springer, Heidelberg (2010)
7. NVIDIA: NVIDIA CUDA Compute Unified Device Architecture Programming Guide, version 4.1, http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf
8. Lee, T.H.: Generalized Aho-Corasick Algorithm for Signature Based Anti-Virus Applications. In: Proceedings of 16th International Conference on Computer Communications and Networks, ICCN (2007)
9. Pungila, C.: A Bray-Curtis Weighted Automaton for Detecting Malicious Code Through System-Call Analysis. In: 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC, pp. 392–400 (2009)
10. Erdogan, O.: Hash-AV: fast virus signature scanning by cache-resident filters. International Journal of Security and Networks 2(1/2) (2007)
11. Lin, P.C., Lin, Y.D., Lai, Y.C.: A Hybrid Algorithm of Backward Hashing and Automaton Tracking for Virus Scanning. IEEE Transactions on Computers 60(4), 594–601 (2011)
12. Cha, S.K., Moraru, I., Jang, J., Truelove, J., Brumley, D., Andersen, D.G.: Split Screen: Enabling Efficient, Distributed Malware Detection. In: Proc. 7th USENIX NSDI (2010)
13. Miretskiy, Y., Das, A., Wright, C.P., Zadok, E.: Avfs: An On-Access Anti-Virus File System. In: Proceedings of the 13th USENIX Security Symposium (2004)

14. Tuck, N., Sherwood, T., Calder, B., Varghese, G.: Deterministic memory-efficient string matching algorithms for intrusion detection. In: 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, vol. 4, pp. 2628–2639 (2004)
15. Snort, http://www.snort.org/
16. Zha, X., Sahni, S.: Highly Compressed Aho-Corasick Automata For Efficient Intrusion Detection. In: IEEE Symposium on Computers and Communications, ISCC, pp. 298–303 (2008)
17. Vasiliadis, G., Polychronakis, M., Ioannidis, S.: MIDeA: A Multi-Parallel Intrusion Detection Architecture. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS, pp. 297–308 (2011)
18. Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos, E.P., Ioannidis, S.: Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In: Lippmann, R., Kirda, E., Trachtenberg, A. (eds.) RAID 2008. LNCS, vol. 5230, pp. 116–134. Springer, Heidelberg (2008)
19. Liu, C.H., Chien, L.S., Chang, S.C., Hon, W.K.: PFAC Library: GPU-based string matching algorithm. In: PU Technology Conference, GTC (2012)

Intended Actions: Risk Is Conflicting Incentives

Lisa Rajbhandari and Einar Snekkenes

Norwegian Information Security Laboratory, Gjøvik University College, Norway
{lisa.rajbhandari,einar.snekkenes}@hig.no

Abstract. Most methods for risk analysis take the view that risk is a combination of consequence and likelihood. Often, this is translated to an expert elicitation activity where likelihood is interpreted as (qualitative/ subjective) probabilities or rates. However, for cases where there is little data to validate probability or rate claims, this approach breaks down. In our Conflicting Incentives Risk Analysis (CIRA) method, we model risks in terms of conflicting incentives where risk analyst subjective probabilities are traded for stakeholder perceived incentives. The objective of CIRA is to provide an approach in which the input parameters can be audited more easily. The main contribution of this paper is to show how ideas from game theory, economics, psychology, and decision theory can be combined to yield a risk analysis process. In CIRA, risk magnitude is related to the magnitude of changes to perceived utility caused by potential state changes. This setting can be modeled by a one shot game where we investigate the degree of desirability the players perceive potential changes to have.

Keywords: Game theory, Risk analysis, risk, conflicting incentives, intended actions.

1 Introduction

One of the key objectives of risk analysis is to provide insight suitable for deciding if risk exposure needs to be changed. That is, if a mitigation action is needed, or risk exposure may be increased. Most methods for risk analysis (including the ISO standard 27005 [1], NIST 800-30 [2], COBIT [3], CORAS [4]), take the view that risk is a combination of consequence and likelihood. Often, this is translated to an expert elicitation activity where likelihood is interpreted as (qualitative/ subjective) probabilities or rates. However, for cases where there is little data to validate probability or rate claims, this approach breaks down. Besides the use of subjective judgment, the other challenge to Probabilistic Risk Analysis (PRA) is handling of human performance and error [5]. Studies have shown that experts rely on heuristics in making judgments (decision) which might result in biases and errors [6, 7]. In addition, people are generally not well calibrated at estimating probabilities [7]. Taleb [8] has provided examples of incidents (called Black Swans) that cannot be accurately predicted based on the historical data. Thus, the questions is: What information relating to uncertainty can one reasonably expect to be able to collect reliably, and how should this information be framed so that it is 'auditable'? Clearly, one approach could be to explicitly capture the uncertainty using techniques such as interval analysis, p-boxes or second order (subjective) probabilities. In this paper, we take a different view.

The objective of our Conflicting Incentives Risk Analysis (CIRA) method is to provide an approach where 'probability affecting' input parameters that can cause risks to go from 'acceptable' to 'unacceptable' can be audited more easily. In CIRA, we model risks in terms of conflicting incentives where risk analyst subjective probabilities are traded for stakeholder perceived incentives. These incentives are by necessity subjective. After all, it is these subjective perceptions that will influence the stakeholder actions. However, the risk analyst should aim to collect these incentives objectively without being influenced by his own subjective judgment. The key contribution of this paper is to show how ideas from game theory, economics, psychology, and decision theory can be combined to yield a risk analysis method.

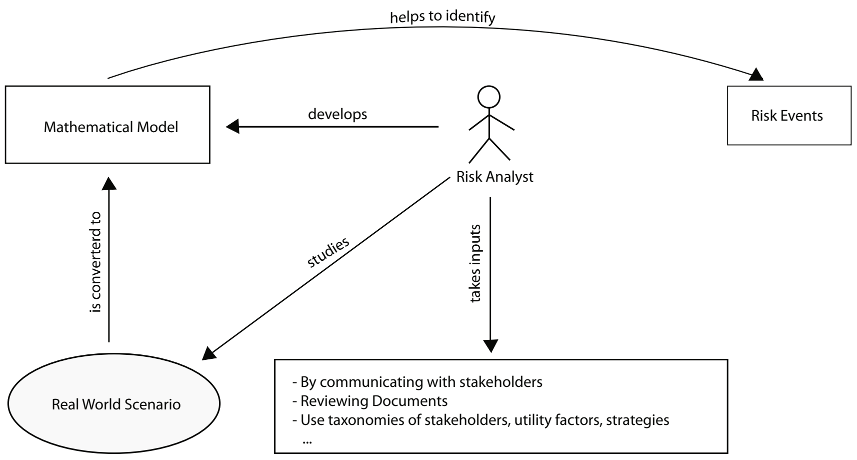


Fig. 1. The Risk Analysis Process

When doing risk analysis, the risk analyst studies the real world scenario and makes the abstraction 'a mathematical model' as depicted in Fig. 1. In reference to this model, the analyst then comes up with a 'hypothetical scenario' which he communicates with the stakeholders during workshops, interviews and/or surveys. He then collects data by gathering responses from the stakeholders, reviewing documents, using existing taxonomies of stakeholders, strategies, etc. Finally, he makes an estimation based on the model and identifies the risk events. However, the analyst needs to consider the validity of the model. That is, the abstraction should capture the key features of the real world such that deductions made from the abstraction provide a close approximation of what also will hold in the real world.

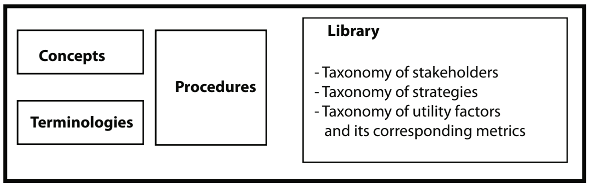

Concepts
Terminologies
Procedures
Library
- Taxonomy of stakeholders
- Taxonomy of strategies
- Taxonomy of utility factors and its corresponding metrics

Fig. 2. The CIRA Method

The CIRA method includes various concepts that are used to build the method, terminologies for understanding the method and procedures for implementing the method. The library includes the taxonomy of stakeholders, taxonomy of strategies, and taxonomy of utility factors and its corresponding metrics. Thus, the library can help the analyst to identify the relevant stakeholders, and for each stakeholder, identify their utility factors and its corresponding metrics, and strategies. The content of the library is dynamic, i.e. existing content can be updated and new content can be added. For instance, the choice of the metric for a utility factor such as privacy might differ depending on stakeholders and application. Thus, with the library, the risk analyst can be provided with a selection of possible definitions of metrics capturing privacy that he can select from or modify.

In CIRA, the risk owner is the stakeholder whose perspective we take when doing our analysis. We focus on risks at the managerial level rather than the technical level. We use aspects of Multi Criteria Decision Analysis (MCDA) and Multi-attribute Utility Theory (MAUT) [9, 10, 11] to estimate stakeholder utility. We model stakeholder incentives (i.e. changes in utilities) as the sum of weighted values for utility factors such as privacy, satisfaction etc. One important motivation for the work reported in this paper is that in classical risk analysis, risk having high consequence and low probability is problematic unless there is strong evidence that the likelihood is small. Verification of low probability is difficult and requires patience (i.e. waiting a long time for nothing to happen). On the other hand, utility factors can be identified from past actions, surveys/ interviews and research in psychology. It is well known that many of the incidents are caused by psychological motives. Thus, we want to include psychological motives like revenge or desire for recognition in our methodology. However, documenting that a stakeholder perceives that he has little to gain by forcing an incident will often require less patience than verifying that the probability of the incident is small.

In our model, a strategy is an action that may modify the value of the utility factors. The strategy owner is the stakeholder that is in the position to trigger the execution of the strategy. Figure 3 depicts a scatter plot of the utilities of the outcome of some example strategies, where the strategy owner (risk owner) utility is plotted on the x (y) axis. Each quadrant represents various situations for the stakeholders: (1) 'cooperation', where both the stakeholders have the incentive to cooperate as they both gain positive utility, (2) 'desirable' situation for the risk owner as he gets positive utility and the converse for the strategy

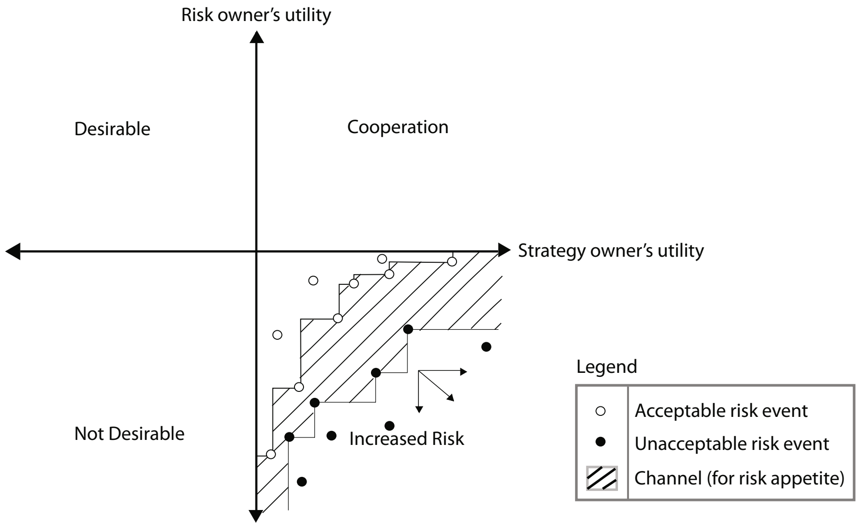


Fig. 3. Incentive graph

owner, (3) 'not desirable' situation for the stakeholders- as they both get negative utility, they both want to move out of this quadrant and (4) risky situation for the risk owner as he gets negative utility while the strategy owner gets positive utility.

In this paper, we focus on the fourth quadrant where each dot represents a particular *risk event*, caused by the 'intentional' execution of a strategy (intended action) by the strategy owner. Intended actions include all the adversarial activities, not just internal issues within the organization. Other risk events caused by unintended/ accidental behavior of the strategy owner, environmental risks, technical failures, etc will be investigated in future work. We argue that the underlying cause of risk is in fact the 'conflicting incentives' situation itself. In the case of intentional behavior of the strategy owner, we have a conflicting incentive situation when a strategy owner can cause a transition that will yield a gain for himself and a loss for the risk owner. This setting can be modeled by a one shot game where we investigate the degree of desirability the stakeholders perceive potential changes to have. The degree of desirability (undesirability) can be expressed in terms of positive (negative) change in utility. That is the greater the positive change in utility, the greater the degree of desirability. The idea being that risk is the combination of the degree of desirability that motivates the strategy owner to send the risk owner to an undesirable state and the magnitude of this undesirability. The risk for the risk owner increases when there is a decrease in his utility or an increase in the utility of the strategy owner or any combination of the former two. Other details regarding risk appetite will be explained later in Sect. 4.

The remainder of this paper is structured as follows. Related work is given in Sect. 2. In Sect. 3, we first give a brief overview of the steps of the CIRA method, then we explain the SNS scenario and finally our method by means of a running example of the interaction between two stakeholders in Social Networking Services (SNS): the data subject and the data controller. Even though CIRA can address multiple players game, for the purpose of explaining our approach we chose a simplified example including two players. In Sect. 4 and Sect. 5, we discuss our findings and future work respectively. Conclusion is given in Sect. 6.

2 Related Work

Researchers have put forward various definitions of risk. According to Slovic et al. [12], risk can be considered in three basic ways: risk as feelings (proposed by Lowenstein et al. [13]), risk as analysis and risk as politics. Risk as feeling consists of individuals' "fast, instinctive, and intuitive reactions to danger"; risk as analysis covers the scientific approach to risk such as risk assessment approaches; risk as politics comes into effect when there is a clash between the former two [12].

As mentioned earlier there are many classical risk management approaches such as the ISO standard 27005 [1], NIST 800-30 [2], COBIT [3] and CORAS [4]. For risk assessment, apart from estimating risk as the combination of consequence and likelihood as in the methods given above, risk is also determined as the combination of threat, vulnerability and consequence (e.g. in [14]). In [15], Cox has shown the limitations of estimating risk as the combination of threat, vulnerability and consequence.

Work on risk analysis and game theory includes: [16] for estimating reliability of a system, [17, 18] for adversarial risk analysis, [19] for cybersecurity risk assessment and [20, 21] for counterterrorism. Cox [17] states that the use of game theory for risk analysis can improve the existing (adversarial) risk analysis approaches. According to Cox, this can be achieved by developing the risk models using the concept of risk analysis, then utilizing game theory for optimizing the decision of the defender in consideration to the attacker's best response. Bier et al. [18] emphasize the importance of game theory for risk analysis as it considers the actions of intelligent and adaptive adversaries. In [21], risk and reliability analysis is combined with game theory for the protection of complex systems against intelligent and adaptable adversaries. As game theoretic approach provides a way to analyze the situations of conflict between the players, it helps to understand the behavior of real world adversaries [22]. Furthermore, in [23], it was shown that game theoretically inspired risk management process can be integrated into the ISO 27005 standard.

Further, research has been done on understanding the incentives of the stakeholders which can enhance the risk analysis process. Liu et al. [24] proposed the incentive based modeling approach in order to model and understand attacker intent, objectives and strategies. Anderson et al. [25] emphasize the importance of incentives, as misaligned or bad incentives usually cause security failure, i.e. gives rise to risk.

3 Conflicting Incentives Risk Analysis (CIRA)

In this section, we first give a brief overview of the steps involved in CIRA. We then explain the social networking services (SNS) scenario. Finally, we explain our method by means of a running example of SNS.

3.1 Summary of the CIRA Method

The overview of CIRA method is as follows:

1. Identify the stakeholders.
2. For each stakeholder,
   2.1. identify the utility factors. For each identified utility factor, determine the scale, measurement procedure, explain the underlying assumptions if any and assign weights as perceived by the respective stakeholder.
   2.2. identify the strategies to be considered.
3. Model a one shot game, then
   3.1. determine the final value of the utility factors after the strategies of the players are executed.
   3.2. estimate the utility for each of the strategies for each of the players.
4. Compute the incentives (i.e. changes in utilities) for each of the strategies for each of the players.
5. Determine risk by investigating each of the strategies with respect to the sign and magnitude of the changes determined in 4.

3.2 The Social Networking Services (SNS) Scenario

To explain our method, we model a setting between two stakeholders in SNS: the data subject (i.e. user) and the data controller (i.e. SNS provider). The data controller collects personal information of the data subject to provide the service. However, it is well known that this information can be used either to provide general advertisements (ads) in compliance with the privacy policy (not exploiting the personal information of the data subject) or by facilitating targeted ads (exploiting the personal information for commercial purposes). Even though most prominent SNS allow third parties to define the types of profiles they are targeting, to which the social network then delivers targeted ads. The SNS might change their policy to sell data to third parties.

3.3 Explaining the CIRA Method

We now explain our method by means of a running example of SNS.

1. Identify the Stakeholders. The first step is to identify the stakeholders i.e. identify the risk owner along with the ‘n’ number of strategy owners. The risk owner is assumed to be player 0. In our example, the stakeholders are the data subject (risk owner) and the data controller (strategy owner).

2. Identify the Utility Factors and Strategies. For each stakeholder, we need to iteratively identify the utility factors and the strategies.

2.1. Identify the Utility Factors. For each stakeholder, identify the associated collection of utility factors that provides the player with utility.

There are many definitions of assets (a term used in classical risk analysis usually meaning anything that has value to the organization) which lead to confusion [26]. Thus, in order to avoid this confusion and in the context of our method we use the term *utility factors*. All the assets can be cast to our notion of utility factors. Utility factors depend on the perspective of the player.

In our SNS example, we assume the utility factors for the data subject to be having privacy and to be satisfied (includes availability, support: responsiveness and effectiveness, service completeness) from using the service. Similarly, we assume that the data controller is only concerned about gaining marginal net profit and having a good reputation (includes experience of others and own experience). Now, for each of the above utility factors, we determine: the scale including the semantics of values, the measurement procedure (incorporating the idea) from behavioral economics [27, 28, 29] and the underlying assumptions if any. Then the weights are assigned to the utility factors as perceived by the respective player using aspects of Multi Criteria Decision Analysis (MCDA) techniques [11]. We assume that correlation among factors are captured in the assignment of weights.

Recall that the metrics used in our example are not prescribed, but belong to the extensible part of our method. The given metrics are just examples of what a metric might look like rather than metrics that have been validated for use in a specific real life risk analysis.

Privacy : First we need to decide if we need a projected or historic privacy metric. This decides if we will count actual incidents or make use of expected/projected number of incidents based on knowledge relating to data subject, data controller and third party behavior. We decide to construct a projected privacy metric.

Scale: Privacy (scale: %) is defined to be

$$1/(1+N) \tag{1}$$

where N is the number of times that a private data object is directly or indirectly utilized (e.g. read, copied or used as input to a decision process) in a way that is in conflict with a data usage purpose statement that is understood and accepted by the data subject. These numbers come from the analysis of the scenario, directly and indirectly caused by the events triggered by the various stakeholders. Note that we make the simplifying assumption that all data objects have the same sensitivity and that all breaches of use are equally serious. The privacy incident count is not to be restricted to incidents directly caused by the data controller, but also includes breaches initiated by a third party having obtained the sensitive data as a result of a privacy incident. If required, we can easily introduce separate categories relating usage breach and data object category. The above privacy metric is relative to a single subject. However, we may modify this metric to measure the commitment of the data controller to process data in a privacy friendly way by counting the incidents relating to all data subjects. Depending on the setting, it may be relevant to limit the counting of N to a specific time period (e.g. per month, per year). In the case that N cannot be uniquely defined, e.g. because we only get to know about a subset of the privacy incidents, we end up with a lower bound on N.

Measurement procedure:

1. Identify all data objects having a privacy requirement.
2. Identify all physical/logical locations where each of these data object instances are stored over the lifetime of the data (i.e. until all copies of the data have been deleted).
3. Identify what purpose is understood and accepted by the relevant subject for each of the data objects.
4. Over the lifetime of the data, count the number of projected privacy incidents.

In many cases it might be difficult to provide the number of data locations with any precision. However, in most cases, we will be able to provide upper and lower bounds. We can then use e.g. interval arithmetic [30] when computing the metric.

Satisfaction : There are many issues relating to customer satisfaction (see e.g. work on American Customer Satisfaction Index (ACSI) [31]).

Scale: For the purpose of this metric, we model satisfaction as expectation fulfillment relating to function. We stipulate that users have expectations relating to the following issues: (1) Service availability (scale: %), (2) Support if problems including: (2.1) responsiveness (scale: %) and (2.2) effectiveness (scale: %) and (3) service completeness (scale: %). Thus, the satisfaction metric consists of four 'sub metrics'. The relative significance that a user puts on each of these factors can be established using the MCDA explained below.

Measurement procedure: For each of the above service performance issues, we can establish objective values e.g. service availability by number of interactions with a response time of less than 1 second divided by the total number of interactions (this leaves out infrastructure issues). Support really relates to two separate issues: responsiveness (scale: %), and effectiveness (scale: %) i.e. the 'extent' to which the problem is solved (e.g. completely for now and all future similar problems, or just a partial solution to this particular instance of the problem). Responsiveness is given as

$$1/(1+t) \tag{2}$$

where t is the average time in minutes required to 'solve' a problem reported by the user. Service completeness relates to the number of features that the service actually delivers divided by the number of features that the user could reasonably expect (e.g. based on similar services provided by others, or suggested in marketing material from the service provider). Here we make the simplifying assumption that all users perceive each function to be equally important.

For satisfaction, other elements may be relevant such as satisfaction from socialization and usability. However, these are not included because we use a simplified model to explain the method. In addition, other strategies of the data controller such as increasing the 'lock-in-effect' may represent risk.

Profit : The unit for profit is currency units (Euros). The weight for profit will then specify how much utility each currency unit will give.

Reputation : We interpret reputation as the data subject's expectancy relating to future behavior of the data controller. The reader should note that 'expectancy relating to future behavior' refers to the subjective, psychological expectation of the data subject and not 'expected value' in a statistical sense. For the purpose of this case study, we restrict our attention to privacy. Thus, a more appropriate name for this metric would probably be 'Privacy Reputation'.

Scale: When it comes to expectations about reputation it may be strongly influenced by past behavior i.e. our experience [32]. Thus, we model reputation relating to two issues: experience of others (scale: %) and own experience (scale: %).

Measurement Procedure: Both of the above metrics can be established by doing a survey. Clearly, we may also construct more sophisticated models taking into account incident discovery rates, and how (negative) information is spread in a human communication network. For example, we may want to establish the following: "How many privacy incidents affecting your friends (yourself) would you be willing to accept before you would stop using the service?". Then, experience of others (own) can be computed using

$$1/(1 + P) \tag{3}$$

where P is the maximum number of privacy incidents affecting your friends (yourself) that you are willing to accept. The assumption that users are more willing to accept privacy violations to others than themselves might not always hold. As mentioned earlier, we can easily address this by presenting the risk analyst with several reputations metrics so that he can choose and construct the final model using the most appropriate metric.

In our SNS example, we assume the data subject gives higher concern to his privacy than to the satisfaction he gets from using the service. On the other hand, the data controller values his profit more than reputation. In our example, we assume data subject privacy and satisfaction to be 0.6 and 0.4 respectively. Similarly, we assume the data controller assigns weights of 0.7 and 0.3 for his reputation and profit respectively. For the other elements comprising the utility factors, we make the assumption that the stakeholders perceive each of these to be equally important (see Table 1).

2.2. Identify the strategies. Determine the associated strategies of each player (except for risk owner) i.e. 'what can they do or consider doing?'. We assume the risk owner plays only the 'do nothing' option. In general, a strategy may be triggered by a planned behavior or stochastically. Note that in this paper we consider only planned or intentional execution of strategies by the strategy owner. Each strategy may modify the value of the utility factors belonging to own as well as other players' utility factors.

From the scenario, we know that the strategies of the data controller are 'do nothing' (DN'), 'exploit' (E) and 'not exploit' (NE). However, the data subject plays only the 'do nothing' (DN) option.

3. Model the Game. Model a one shot game (static and simultaneous game) between the risk owner and strategy owners. Assume the system/ environment to be in a fixed initial state and all the players are utility optimizing. By utility optimizing, we mean that they are optimizing their behavior relative to the weighted sum of the elements in their utility factor vector.

Now, we need to first determine the final value of the utility factors, then estimate the utility for each of the strategies for each of the players.

3.1. Determine the Final Value of the Utility Factors. For each of the identified utility factors, determine the final value after the strategies of the players are executed (for the utility factors' valuation, we utilize the metrics explained above). We use the additive utility function of MAUT to estimate the utility. The additive utility function for a given player is defined to be the weighted average of its individual utility functions [9] given as:

$$U = \sum_{k=1}^{m} w_k \cdot u(a_k) \ . \tag{4}$$

where
m- number of utility factors of the player,
w_k is the assigned weight of utility factor a_k and $\sum_{k=1}^{m} w_k = 1$, and
$u(a_k)$ is the utility function for the utility factor 'a_k'.

For our SNS example, Table 1 depicts the initial value (IV) of the utility factors and also its final value, if the strategies of the data controller were to be executed. For the purpose of this example, we assume the values are obtained from interviews and surveys. Note the 'do nothing' option of the data controller and data subject does not incur any changes to utility factors.

For obtaining the values of *privacy* of the data subject, we instantiate (1) with the assumption N=0 per month at the initial state and when the data controller uses option NE. However, we assume N=10 per month when the data controller uses option E. This results in the values of privacy as 100% and 9% respectively.

Note that the values for satisfaction and reputation are obtained using the techniques borrowed from MCDA and MAUT. For *support* (an element of the satisfaction utility factor), the values for the *responsiveness* are obtained by instantiating (2) with $t = 5$ at the initial state and $t = 4$ when both the strategies of the data controller are executed. Thus, responsiveness increases from the IV of

Table 1. Final Values of the Utility Factors after the Strategy of the Data Controller is Executed (an example)

				Final Values	
Stakeholders	**Utility Factors**	**Weights**	**IV**	**NE**	**E**
Data Subject	Privacy (%)	0.60	100	100	9
	Satisfaction (%)	0.40	70	74	74
	Availability (%)	0.33	80	85	85
	Support (%)	0.33	53	56	56
	Responsiveness (%)	0.50	17	20	20
	Effectiveness (%)	0.50	90	92	92
	Service Completeness(%)	0.33	80	82	82
Data Controller	Profit (Euros)	0.70	200	200	400
	Privacy Reputation (%)	0.30	42	75	38
	Experience of others(%)	0.50	33	50	25
	Own experience(%)	0.50	50	100	50

17% to 20% for both the strategies. Besides, we assume the *effectiveness* increases from 90% to 92% when both the strategies of the data controller are executed. Now, we evaluate the values for support instantiating (4) with the obtained values of responsiveness and effectiveness: for the IV as $0.50 \cdot 17 + 0.50 \cdot 90 = 53\%$. Similarly, the final values for both the NE and E are evaluated as 56%. We make the following assumptions for the other elements of satisfaction: *availability* increases from 80% to 85% and *service completeness* increases from 80% to 82% after both the strategies of the data controller are executed. Thus, using (4) and the values determined for the other elements comprising our satisfaction utility factor, the obtained IV is 70% and the final values for both the strategies are evaluated as 74%.

We assume the data controller makes an additional profit of 200 when he uses the E option rather than the NE option. For *reputation*, for the *experience of others*, the values are obtained after we instantiate the number of privacy incidents in (3) with $P = 2$, $P = 1$ and $P = 3$ in the initial state, when the data controller uses NE and E as 33%, 50% and 25% respectively. On the other hand, for *own experience* i.e. personal experience of the data subject, the values are obtained as follows after we instantiate in (3), $P = 1$ in the initial state and when the data controller uses E and $P = 0$ when the data controller uses NE. Thus, the values obtained in the initial state, when the data controller uses NE and E are 50%, 100% and 50% respectively. Thus, using (4) from the values determined for the elements comprising reputation, reputation of the data controller increases from an IV of 42% to 75% when the data controller chooses the NE option. However, it decreases to 38% when the data controller selects the E option.

Usually, the individual utility functions (i.e. utility factors in our case) are assigned values in the interval of 0 (worst) to 1 (best) when using MAUT. For instance, in our case, we can easily compress the profit/ wealth to the interval 0 to 1. However, this would not be particularly helpful as most of the values will be clustered right at the end. Thus, it is more intuitive to utilize the given scales for the utility factors' valuation.

	Utilities			Change in Utilities (Δ)	
Stakeholders	**IV**	**NE**	**E**	**NE**	**E**
Data Subject	88	89	35	89 - 88 = 1	35 - 88 = - 53
Data Controller	153	163	291	163 -153 = 10	291 - 153 = 138

Fig. 4. Matrix of Utilities and Change in Utilities w.r.t. Strategy of the Data Controller

3.2. Estimate the Utility. We again use the techniques from MAUT to estimate the utility for each of the strategies for each of the players using (4). We make the simplifying assumption that utility is linear.

For our SNS example, we use (4) to compute the utilities for the data subject and the data controller with the values given in Table 1. In the initial state, the utilities are given as follows:
For the data subject: $0.60 \cdot 100 + 0.40 \cdot 70 = 88$
For the data controller: $0.70 \cdot 200 + 0.30 \cdot 42 = 153$
Similarly, when the data controller selects NE and E, the utilities are obtained as given in Fig. 4.

4. Compute the Incentives. We need to compute the incentives (i.e. changes in utilities) for each of the strategies for each of the players. The change in utility Δ is the difference between the utility of the player in the state resulting from strategy use and the initial state.

In the SNS example, from Fig. 4, when the data controller uses the NE option, Δ for the data subject and data controller are 1 and 10 respectively. When the data controller uses the E option, the Δ for the data subject and data controller are -53 and 138 respectively.

5. Determine Risk. This can be achieved by investigating each of the strategies with respect to sign and magnitude of the changes determined in 4. The idea being that risk is the combination of the strength of the force that motivates the strategy owner to send the risk owner to an undesirable state and the magnitude of this undesirability. Risk magnitude is related to the magnitude of changes to perceived utility caused by potential state changes.

In other words, we look into how strong the players' incentives are to make the first move. Investigating the players' motivation to move first helps to understand the risks faced by the risk owner. In our model, we make the simplifying assumption that all strategy owners will need the same time to act if they have the same magnitude of incentive. Furthermore, players will move ordered by decreasing incentives and all above a certain threshold will move.

In the SNS example, when the data controller uses the NE option, it results in a positive change in utility for both the players (falls in the first quadrant in the incentive graph). Thus, we know there is no risk. However, it is clear that the data controllers' degree of desirability to play the exploit option is high as it leads him to a better position with a gain of 138. In this case, 138 is the strength of the force that motivates the data controller to send the data subject to an undesirable state and -53 is the magnitude of this undesirability and the combination of these is the risk **(-53, 138)**.

4 Discussion

In a game theoretic interpretation, if you are in equilibrium, 'forces' will pull you towards your current state. That is, there is no incentive for the players to unilaterally change state, as the state where the risk owner is heading might be worse for both himself and the strategy owner. Thus, if the strength of the 'force' to change state is negative then we are in equilibrium. We may also define game theoretic risk to be a set related to the concepts of likelihood and consequence as:

1. Likelihood(L_i): How strong is strategy owner i's incentive to make the first move or the magnitude of incentive. The theory of planned behavior provides a link [33].
2. Consequence(C): The value of the game for the risk owner.

Furthermore, risk appetite can be determined by asking questions like: 'How strong a temptation is acceptable to give a strategy owner to cause the risk owner a given loss?' In this setting, risk appetite refers to the collection of (C, L_i) pairs (referred to as SCL) for which the risk owner accepts (and prefers) that the strategy owner i makes the first move. In other words, this collection helps to decide if a risk is acceptable or not. That is, for risk acceptance criteria (C, L_i), risk (C', L_i') is acceptable, iff $C' \geq C$ and $L_i' \leq L_i$. Note that a loss (gain) will have a negative (positive) sign and we need to specify many risk pairs in order to completely specify the risk acceptance criteria. We leave the definition of a total ordering of risk for further work. If the risk owner finds himself in a situation where the current (C, L_i) is not in SCL, he will need to consider other strategies than the 'do nothing' strategy. However, this takes us from risk analysis to risk management which is outside the scope of this paper.

Recall, as shown in Fig. 3, that each dot represents a particular risk event. The quadrant is divided into three areas: the area for acceptable risk events (represented by white dots) and the area for unacceptable risk events (represented by black dots) separated by the shaded area. The shaded area represents the channel in which the analyst does not have enough information to know if a risk event is acceptable or not. The risk owner's risk appetite will be a graph somewhere in this channel which specifies a bound on the risk he currently is willing to take. However, the risk appetite will differ depending on how the event is triggered. For intended action, in many cases, it will be reasonable to make the assumption that the strategy owner will be rational in a behavioral economics sense. However, the risk appetite in the case of accidental execution may be different. Consequently, we may have multiple risk acceptance graphs.

From the strategy owner's perspective there is always the possibility that the risk owner will implement a control that reduces the incentive that the strategy owner may obtain. This concern is captured through the application of the theory of Discounted Utility (DU) ([34]). The first mover advantage is then equal to the saved opportunity cost that can be attributed to the discount factor. We make the simplifying assumption that all risk events have associated the same discount factor. However, if we have evidence suggesting that this is not the case, we can easily modify our calculations, e.g. by using interval arithmetic to capture the incertitude or by using specific discounting values if these are known. DU has received some criticism relating to the assumption that the discounting factor is assumed constant for all time intervals. However, we do not rely on this assumption as it can easily be seen that the greater the incentive, the greater the potential loss attributable to the discounting. Thus, from the assumption above, a rational strategy owner that is forced to make a choice between risk events will select the risk event with the highest value. This will (for a given strategy owner) result in risk events being sorted according to their relative urgency. However, in general we may not be able to predict the absolute time for when a risk event may occur. Thus, the relative timing of risk events triggered by different strategy owners may not necessarily be known. One way of dealing with this is to establish separate risk appetites for each of the strategy owners. This would then remove the need for assuming that all stakeholders have the same response time for a given incentive.

In addition, thinking of player strategies reveal risks in a way that typical occurrence of events in classical risk analysis does not. As we look into the strategies from the perspective of each stakeholder, we claim it provides a clearer view of the situation in terms of not only what the stakeholder can do but also what he might consider doing. Furthermore, it should be noted that quantification as such does not give objectivity. This is because there will be an element of subjectivity going into the choice and definition of the metrics capturing the utility factors. In most of the classical risk analysis approaches, the probability is obtained by asking questions such as “How likely it is that something will happen?”. When there is insufficient historical data or when we have small probabilities, it is difficult to check if the probability is in fact correct. In CIRA, we ask questions such as “How much does a person perceive to benefit from a certain incident?”. As objectivity is closely related to ‘universal agreement’, our transparent approach goes some way towards this goal by means of identifying the key issues where disagreement may exist. Once identified, one can work towards agreement and a common understanding.

5 Future Work

For future work, we will investigate to what extent CIRA scales to real world scenarios. We will consider risk from a single (i.e. ‘first’) adversarial move vs. risks from some number of adversarial moves vs. other scenarios. The idea being that we need to consider the risks posed by all the events up to the point where we have changed our exposure. Besides, we will compare CIRA with one or more of the classical risk analysis methods such as ISO 27005 which will contribute towards understanding its effectiveness.

The following issues require further work: (1) Expressing risk as a single value. (2) We will collect definitions of utility factors from the literature and put into the library. (3) Different people may value the same utility factor differently. Hence, we need to consider weights being captured as distributions. (4) In our SNS example we made the simplifying assumption that utility is linear. We need to investigate the issues relating to the non-linearity of utility factor valuation. (5) The lack of precision or incertitude can be captured using uncertainty propagation techniques such as P-box, interval arithmetic or similar. This recognition is crucial in order not to enter into a game of self deception. (6) Finally, we will investigate how risk neutral, risk seeking and risk averse behavior can be framed in CIRA.

6 Conclusion

We have presented a method for risk analysis combining ideas from game theory, economics, psychology and decision theory where the input parameters can be audited and where risk is modeled in terms of conflicting incentives. Furthermore, investigating the players' incentive to move first helps to understand the risks faced by the risk owner. Our method trades subjective probabilities for stakeholder perceived incentives. By trading subjective probabilities for stakeholder incentives, the risk analyst can focus on utility factors incorporating idea from behavioral economics during the data collection phase of a risk analysis process. Thus, CIRA is applicable in situations where the utilities for the stakeholders can be estimated reasonably well.

Acknowledgement. The work reported in this paper is part of the PETweb II project sponsored by The Research Council of Norway under grant 193030/S10. We would like to thank Dieter Gollmann and the anonymous reviewers for their valuable comments and suggestions.

References

[1] ISO: ISO/IEC 27005 Information technology -Security techniques-Information security risk management, 1st edn. (2008)
[2] Stoneburner, G., Goguen, A., Feringa, A.: NIST SP 800-30, Risk Management Guide for Information Technology. NIST (2002)
[3] IT Governance Institute: COBIT 4.1, ISA (2007)
[4] Braber, F., Hogganvik, I., Lund, M.S., Stølen, K., Vraalsen, F.: Model-based security analysis in seven steps — a guided tour to the CORAS method. BT Technology Journal 25(1), 101–117 (2007)
[5] Bier, V.M.: Challenges to the acceptance of probabilistic risk analysis. Risk Analysis 19, 703–710 (1999)
[6] Tversky, A., Kahneman, D.: Judgment under uncertainty: Heuristics and biases. Science 185(4157), 1124–1131 (1974)
[7] Shanteau, J., Stewart, T.R.: Why study expert decision making? some historical perspectives and comments. Organizational Behavior and Human Decision Processes 53(2), 95–106 (1992)

[8] Taleb, N.N.: The Black Swan: The Impact of the Highly Improbable, 2nd edn. Random House Trade Paperbacks (2010)
[9] Clemen, R.T.: Making Hard Decision: An Introduction to Decision Analysis, 2nd edn. Duxbury (1996)
[10] Wallenius, J., Dyer, J.S., Fishburn, P.C., Steuer, R.E., Zionts, S., Deb, K.: Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. Management Science 54(7), 1336–1349 (2008); INFORMS
[11] Dodgson, J.S., Spackman, M., Pearman, A., Phillips, L.D.: Multi-criteria analysis: a manual. Department for Communities and Local Government, London (2009) ISBN 9781409810230
[12] Slovic, P., Finucane, M., Peters, E., MacGregor, D.G.: Risk as analysis and risk as feelings: Some thoughts about affect, reason, risk, and rationality. Risk Analysis 24(2), 311–322 (2004)
[13] Loewenstein, G.F., Weber, E.U., Hsee, C.K., Welch, N.: Risk as feelings. Psychological Bulletin 127(2), 267–286 (2001)
[14] ASME Innovative Technologies Institute, LLC: Risk Analysis and Management for Critical Asset Protection (RAMCAP): The Framework, Version 2.0 (2006)
[15] Cox, J.L.: Some limitations of "Risk = Threat x Vulnerability x Consequence" for risk analysis of terrorist attacks. Risk Analysis 28(6), 1749–1761 (2008)
[16] Hausken, K.: Probabilistic risk analysis and game theory. Risk Analysis 22(1), 17–27 (2002)
[17] Cox Jr., L.A.T.: Game theory and risk analysis. Risk Analysis 29(8), 1062–1068 (2009)
[18] Bier, V.M., Cox Jr., L.A.T., Azaiez, M.N.: Why both game theory and reliability theory are important in defending infrastructure against intelligent attacks. In: Game Theoretic Risk Analysis of Security Threats. International Series in Operations Research & Management Science, vol. 128, pp. 1–11. Springer US (2009)
[19] Carin, L., Cybenko, G., Hughes, J.: Cybersecurity strategies: The QuERIES methodology. Computer 41, 20–26 (2008)
[20] Banks, D., Anderson, S.: Combining game theory and risk analysis in counterterrorism: A smallpox example. In: Wilson, A., Wilson, G., Olwell, D. (eds.) Statistical Methods in Counterterrorism, pp. 9–22. Springer, New York (2006)
[21] Bier, V.: Game-theoretic and relaibility methods in counterterrorism and security. In: Wilson, A., Wilson, G., Olwell, D. (eds.) Statistical Methods in Counterterrorism, pp. 23–40. Springer, New York (2006)
[22] Fricker Jr., R.D.: Game theory in an age of terrorism: How can statisticians contribute? In: Wilson, A., Wilson, G., Olwell, D. (eds.) Statistical Methods in Counterterrorism, pp. 3–7. Springer, New York (2006)
[23] Rajbhandari, L., Snekkenes, E.A.: Mapping between Classical Risk Management and Game Theoretical Approaches. In: De Decker, B., Lapon, J., Naessens, V., Uhl, A. (eds.) CMS 2011. LNCS, vol. 7025, pp. 147–154. Springer, Heidelberg (2011)
[24] Liu, P., Zang, W.: Incentive-based modeling and inference of attacker intent, objectives, and strategies. In: Proceedings of the 10th ACM Conference on Computer and Communications Security, CCS 2003, pp. 179–189. ACM, New York (2003)
[25] Anderson, R., Moore, T.: Information Security Economics – and Beyond. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 68–91. Springer, Heidelberg (2007)
[26] Kristandl, G., Bontis, N.: Constructing a definition for intangibles using the resource based view of the firm. Management Decision 45(9), 1510–1524 (2007)

[27] Mullainathan, S., Thaler, R.H.: Behavioral economics. NBER Working Paper 7948 (2000)
[28] Camerer, C.F., Lowenstein, G.: Behavioral economics: Past, present, future. In: Camerer, C.F., Loewenstein, G., Rabin, M. (eds.) Advances in Behavioral Economics, pp. 3–51. Princeton University Press (2004)
[29] Sent, E.M.: Behavioral economics: How psychology made its (limited) way back into economics. History of Political Economy 36(4), 735–760 (2004)
[30] Hayes, B.: Computing science: A lucid interval. American Scientist 91(6), 484–488 (2003)
[31] Fornell, C., Johnson, M.D., Anderson, E.W., Cha, J., Bryant, B.E.: The American Customer Satisfaction Index: Nature, purpose, and findings. Journal of Marketing 60(4), 7–18 (1996)
[32] Money, K., Hillenbrand, C.: Using reputation measurement to create value: An analysis and integration of existing measures. Journal of General Management 32(1) (2006)
[33] Ajzen, I.: The theory of planned behaviour. Organizational Behaviour and Human Decision Processes 50, 179–211 (1991)
[34] Goldin, J.: Making decisions about the future: the discounted-utility model. Mind Matters: The Wesleyan Journal of Psychology 2, 49–56 (2007)

On the Self-similarity Nature of the Revocation Data

Carlos Gañán, Jorge Mata-Díaz, Jose L. Muñoz, Oscar Esparza, and Juanjo Alins

Universitat Politècnica de Catalunya, Telematics Department, Barcelona, Spain
{carlos.ganan,jmata,jose.munoz,oscar.esparza,juanjo}@entel.upc.edu

Abstract. One of the hardest tasks of a Public Key Infrastructure (PKI) is to manage revocation. Different revocation mechanisms have been proposed to invalidate the credentials of compromised or misbehaving users. All these mechanisms aim to optimize the transmission of revocation data to avoid unnecessary network overhead. To that end, they establish release policies based on the assumption that the revocation data follows uniform or Poisson distribution. Temporal distribution of the revocation data has a significant influence on the performance and scalability of the revocation service. In this paper, we demonstrate that the temporal distribution of the daily number of revoked certificates is statistically self-similar, and that the currently assumed Poisson distribution does not capture the statistical properties of the distribution. None of the commonly used revocation models takes into account this fractal behavior, though such behavior has serious implications for the design, control, and analysis of revocation protocols such as CRL or delta-CRL.

Keywords: Self-similarity, Certification, Public Key Infrastructure, Revocation.

1 Introduction

Today we are in the midst of an electronic business revolution. It is of utmost importance that mechanisms are set up to ensure information and data security. Organizations have recognized the need to balance the concern for protecting information and data with the desire to leverage the electronic medium. Public Key Infrastructure (PKI) is a step toward providing a secure environment by using a system of digital certificates and certificate authorities (CAs). However, one of the most important aspects in the design of a PKI is certificate revocation.

Certificate revocation is the process of removing the validity of a certificate prematurely. There could be multiple reasons for revoking a certificate; such as the certificate holder leaves the organization or there is a suspicion of private key compromise. When a certificate is revoked, the information about the revoked certificate needs to be published. Some of the methods that a CA can use to revoke certificates are:

- Periodic Publication Mechanisms: Information about revoked certificates can be posted on a certificate server so that the users are warned from using those certificates. This mechanism includes the use of Certificate Revocation Lists (CRL) and Certificate Revocation Trees (CRT). A CRL is a signed list of certificates that have been revoked or suspended. CRT is a revocation technology, which is based on Merkle hash trees, where the tree represents all known certificate revocation information relevant to some known set of PKI communities.
- Online Query Mechanisms: Online Query Mechanisms comprise Online Certificate Status Protocol (OCSP) and Online Transaction Validation Protocols. OCSP is used to obtain online revocation information about certificates, and Online Transaction Validation Protocols are used for online validation, such as business transactions through credit cards.

A revocation method is selected by an organization based on the cost, infrastructure, and volumes of transactions that are expected. To gauge these costs, different revocation mechanisms are tested under the assumption that the revocation events follow a specific probability distribution. Most theoretical frameworks and simulation studies for performance evaluation assume that the temporal distribution of queries follows a Poisson distribution. Thus, organizations estimate the infrastructure needed to deploy the PKI and the associated costs. However, in this article, we demonstrate that revocation data is statistically *self-similar*, that none of the commonly used revocation models is able to capture this fractal behavior, and that such behavior has serious implications for the design, control, and analysis of revocation protocols such as CRLs.

We start by analyzing the validity of Poisson-like process assumption. We use publicly available CRLs from different certification authorities (containing more than 300,000 revoked certificates over a period of three years). Our analysis demonstrates that the Poisson distribution fails to capture the statistical properties of the actual revocation process. We also see that the Poisson distribution grossly under-estimates the bandwidth utilization of the revocation mechanism. At first glance, this might look like an obvious result, since after all as a memoryless process, Poisson distribution cannot be expected to model periodic trends like daily, weekly and monthly cycles in revocation rates. We show however that the modeling inability transcends simple cycles. In particular, we will show that self-similarity has a severe detrimental impact on the revocation service performance.

Results of our analysis, including burstiness at all scales, strongly suggest self-similar nature of revocation events. We confirm this by estimating the Hurst parameter for the observed distribution and showing that the estimates validate self-similar nature of the revocation lists. Beyond invalidating Poisson-like distributions, this proof of self-similarity has the important implications on CA utilization, throughput, and certificate stratus checking time. Intuitively, as the revocation process is bursty (non-uniformly distributed) the CA will be partially idle during low burst periods and vice versa. Thus, the revocation lists will grow non-uniformly, and current updating policies will result bandwidth inefficient.

The rest of this article is organized as follows. Section 2 gives the necessary statistical background required to understand self-similar processes and long range dependency. In Section 3, we discuss the methodology we used to collect and analyze real-world revocation data. We demonstrate self-similar nature of the revocation data, followed by a Hurst parameter estimation. In Section 4 we discuss how the observed self-similarity has crucial implications on performance of the revocation service. Next section discusses the related work in the area. Finally, we conclude in Section 6.

2 Background

2.1 Self-similar Processes

A phenomenon which is self-similar looks the same or behaves the same when viewed at different degree of magnification. Self-similarity [1] is the property of a series of data points to retain a pattern or appearance regardless of the level of granularity used and can be the result of long-range dependence (LRD) in the data series. One of the main properties of the self-similar data is burstiness [1]. Bursty data do not possess a stable mean value. Significant differences in the mean value are one of the reasons why bursty data are more difficult to control than shaped one. If a self-similar process is bursty at a wide range of timescales, it may often exhibit long-range dependence. Long-range-dependence means that all the values at any time are correlated in a positive and non-negligible way with values at all future instants.

A stochastic process $Y(t)$ is *self-similar* with Hurst parameter H if for any positive stretching factor d, the distribution of the rescaled and reindexed process $d^{-H}Y(dt)$ is equivalent to that of the original process $Y(t)$. This means for any sequence of time points $t_1, \ldots, t_n$ and any positive constant d, the collections $\{d^{-H}Y(dt_1), \ldots, d^{-H}Y(dt_n)\}$ and $\{Y(t_1), \ldots, Y(t_n)\}$ are governed by the same probability law. When the values of H are in the interval $(0.5, 1)$, the process presents LRD. A value of H equal to 0.5 indicates the absence of LRD. This means that the smoothing with aggregation is much slower for self-similar processes, the greater the degree of self-similarity, the slower will be smoothing with aggregation.

Three implications of self-similarity are:

- No natural length of bursts.
- Presence of bursts in all time scales.
- Process does not smooth out on aggregation.

2.2 Statistical Tests for Self-similarity

The practical way to estimate degree of self-similarity is to measure the values of Hurst exponent. In this paper we use five methods to test for self-similarity (details about these methods are described in [2,3]).

The first method, the variance-time plot, relies on the slowly decaying variance of a self-similar series. The variance of $Y^{(m)}$ is plotted against m on a log-log plot; a straight line with slope (β) greater than -1 is indicative of self-similarity, and the parameter H is given by $H = 1-\beta/2$. The second method, the R/S plot, uses the fact that for a self-similar dataset, the rescaled range or R/S statistic grows according to a power law with exponent H as a function of the number of points included (n). Thus the plot of R/S against n on a log-log plot has slope which is an estimate of H. The third approach, the periodogram method, uses the slope of the power spectrum of the series as frequency approaches zero. On a log-log plot, the periodogram slope is a straight line with slope close to the origin.

While the preceding three graphical methods are useful for exposing faulty assumptions (such as non-stationarity in the dataset) they do not provide confidence intervals. The fourth method, called the Whittle estimator does provide a confidence interval, but has the drawback that the form of the underlying stochastic process must be supplied. The two forms that are most commonly used are fractional Gaussian noise (FGN) with parameter $1/2 < H < 1$, and Fractional ARIMA(p,d,q) with $0 < d < 1/2$ (for details see [2])). These two models differ in their assumptions about the short-range dependences in the datasets; FGN assumes no short-range dependence while Fractional ARIMA can assume a fixed degree of short-range dependence. There are several other methods in frequency and time domain to measure the Hurst parameter.

Finally, we use the Detrended Fluctuation Analysis (DFA) [4], which aims to highlight the long-range dependence of a time series with trend. DFA method is a version for time series with trend of the method of aggregated variance used for a long-memory stationary process. It consists in aggregating the process by windows with fixed length, detrending the process from a linear regression in each window, computing the standard deviation of the residual errors (the DFA function) for all data, and finally, estimating the coefficient of the power law from a log-log regression of the DFA function on the length of the chosen window.

3 Examining the Self-similarity of the Revocation Process

3.1 Data Collection

In order to capture the temporal correlation of the revocation process, first we have to gather a large sample of revocation data. The approach we follow consists in collecting revocation data from different certification authorities using their available CRLs. In particular, we built some scripts to download and preprocess the CRLs from the following CAs[1]: VeriSign, GoDaddy, Thawte, and Comodo.

Though we concentrate our analysis on CRL because it is the most common and simplest method for certificate revocation [6], we expect the captured pattern to be extensible to any other revocation mechanism (e.g. OCSP).

[1] According to NetCraft's survey [5], using these CAs we cover most of the world market for SSL.

Table 1. Description of the collected CRLs

Issuer Name	Number of Revoked Certificates	Last Update	Next Update
GoDaddy	932,900	2012/02/01	2012/02/03
VeriSign	5,346	2012/02/02	2012/02/16
Comodo	2,727	2012/02/03	2012/02/06
GlobalSign	7,591	2012/02/02	2012/03/03
Thawte	8,061	2012/02/01	2012/02/16

Once downloaded the revocation data, we preprocess these data to remove duplicated information (e.g. certificates that are revoked due to several reasons). Note that when a revoked certificate expires, it typically remains in the CRLs for one additional publication interval, so we preprocess the CRLs to remove expired certificates too. In this sense, Thawte's and GlobalSign's CRLs may contain duplicate entries for the same certificate because of their policy statements. These policy statements impose that a certificate that is revoked by several reasons must be included in the CRL as many times as the number of revocation reasons. Thus, we remove any duplicate entry from the composite dataset, and tally the number of revocations per day. Finally, we build a dataset that covers non-expired revoked certificates from 2008 to 2012 (see Figure 1).

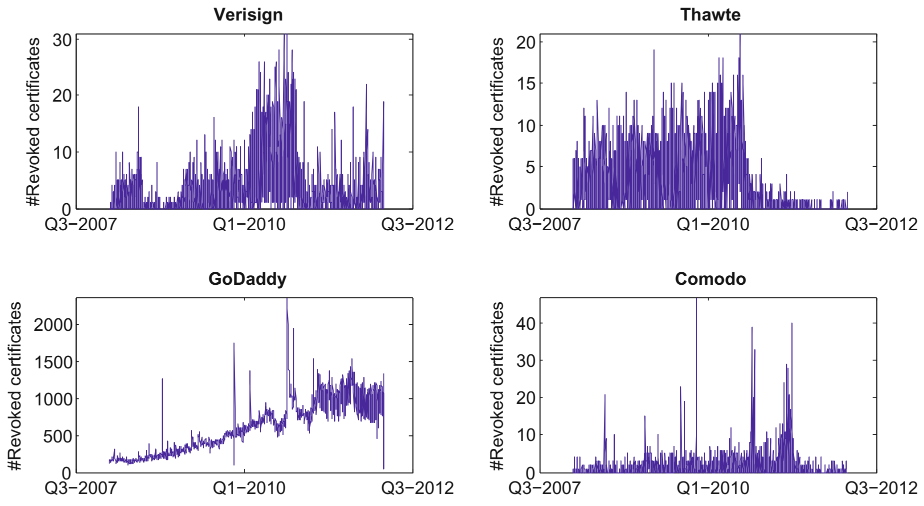


Fig. 1. Number of daily revoked certificates evolution for each CA

3.2 Evidence of Burstiness

Before providing formal estimation of self-similarity, we provide a graphical evidence of bursty nature of the revocation data at different time scales. We also show that this observed burstiness is not accounted by the Poisson distribution. In Figure 2, we show the revocation logs in four different time scales-ranging from 1 hour to 1 day. Each plot is obtained by changing the time resolution. In Figure 2 we can observe different evident trends; (i) Burstiness in all time scales: the burstiness of the revocation process does not disappear when changing the time scales. (ii) Lack of natural length of bursts: The figure shows burstiness ranging from days to months. Note that the full duration of the figure with the largest time slot is 1,000 days, and some of the bursts have many hours of duration.

In addition, it is worth noting the difference between this bursty pattern and a Poisson process. A Poisson process smooths out with large time scales and resembles a uniformly distributed white noise at higher time scales. In contrast to the revocation process, in a Poisson process the burstiness vanishes in coarse time scales, longer length bursts are absent, and bursts smooths out much faster. Thus, the trends of self-similarity present in the revocation data discussed above are totally absent for Poisson processes.

Therefore, modeling the revocation process as Poisson is clearly inadequate, and is thus likely to give unrealistic results. We will elaborate this analysis in the next section, and discuss the consequences of self-similarity in the following sections.

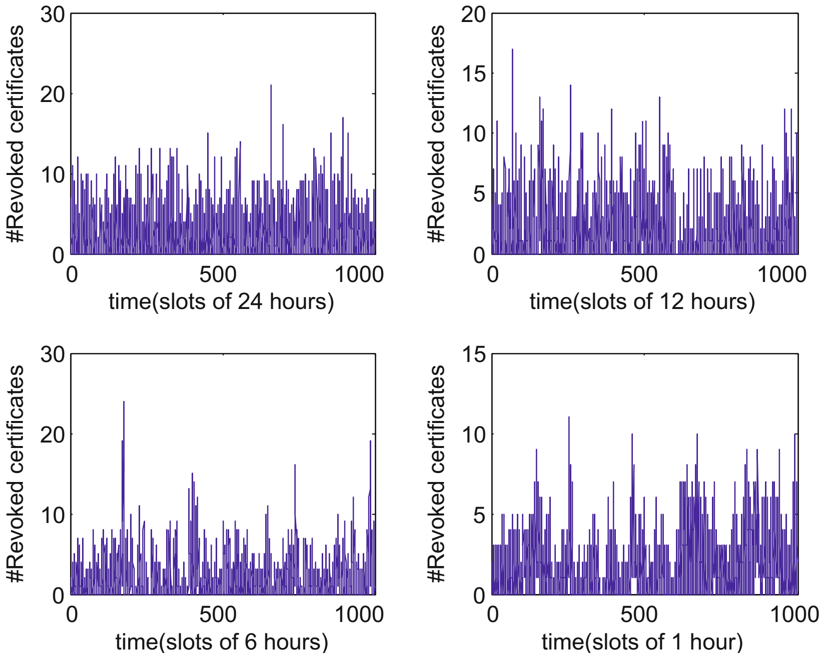


Fig. 2. Revocation Bursts over Four Orders of Magnitude

3.3 Statistical Analysis of Self-similarity

In this section, we use five different methods to estimate the Hurst parameter to demonstrate the long range dependency of the revocation events formally. Since there are different manifestations of self-similarity, different methods in time and frequency domains are used in practice for the estimation (see Sec. 2.2). Note that when using these estimators with real-life revocation data containing noise, cycles and trends, they might estimate different values of the Hurst parameter. For that reason, we use multiple methods, report the correlation coefficients and confidence intervals by different methods, and visually inspect the data for trends and cycles. The chances of estimates agreeing on real data is small [7], but if most of the estimates are above 0.5 the LRD is likely to exist.

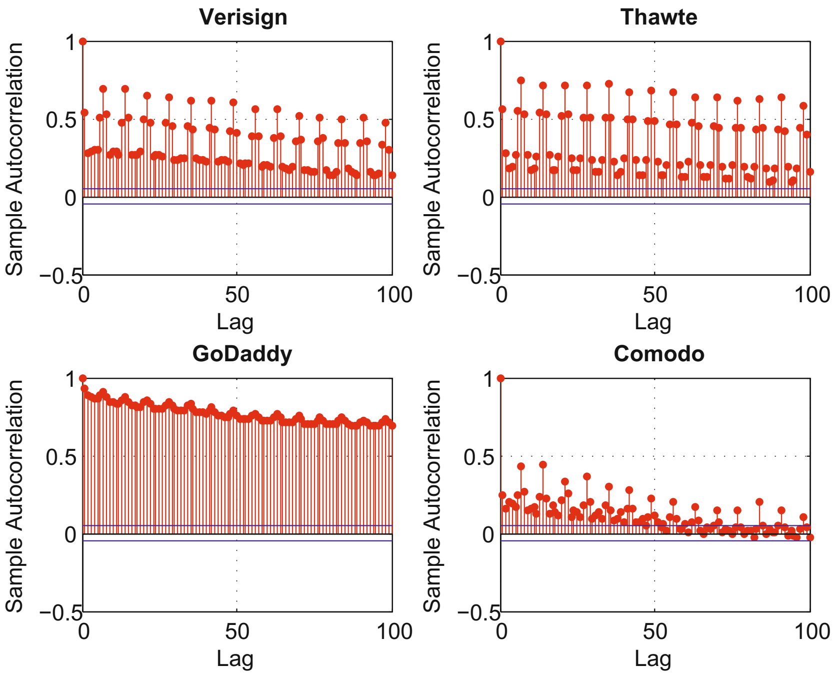


Fig. 3. Autocorrelation function of the revocation process per CA

First of all, we start analyzing the autocorrelation of the revocation data. Recall that in a self-similar process autocorrelations decay hyperbolically rather than exponentially fast, implying a nonsummable autocorrelation function $\sum_k r(k) = \infty$ (long-range dependence). For the frame data, the empirical autocorrelation functions $r(k)$ are shown in Fig. 3, with lag k ranging from 0 to 100. Notice that $r(k)$ decreases slower than exponentially no matter the CA. The curve does decay toward zero, but it does so extremely slowly. The very slowly decaying autocorrelations are indicative of LRD.

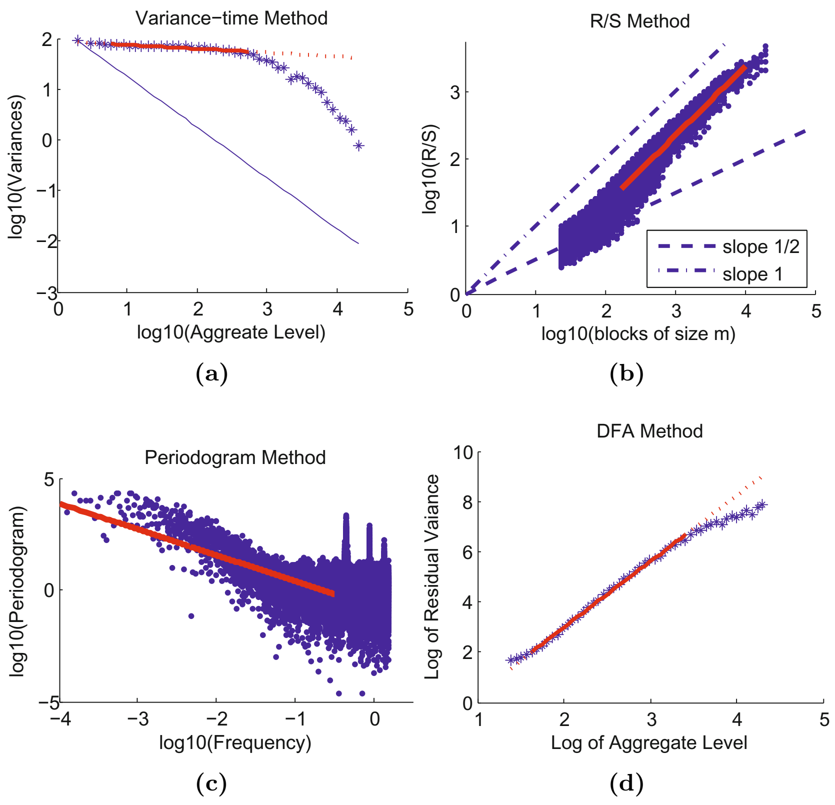


Fig. 4. Graphical methods for checking for self-similarity of the revocation process from GoDaddy (a) variance-time plot, (b) pox plot of R/S, (c) periodogram plot, and (d) DFA plot

In the following, we use five different methods for assessing self-similarity described in Section 2.2: the variance-time plot, the rescaled range (or R/S) plot, the periodogram plot, the DFA plot and the Whittle estimator. We concentrated on individual months from our revocation time series, so as to provide as nearly a stationary dataset as possible. To provide an example of these approaches, analysis of a single month from GoDaddy revocation data is shown in Figure 4. The figure shows plots for the four graphical methods: variance-time (upper left), rescaled range (upper right), periodogram (lower left) and DFA (lower right). The variance-time plot is linear and shows a slope that is distinctly different from -1 (which is shown for comparison); the slope is estimated using regression as -0.077, yielding an estimate for H of 0.96. The R/S plot shows an asymptotic slope that is different from 0.5 and from 1.0 (shown for comparison); it is estimated using regression as 0.95, which is also the corresponding estimate of H. The periodogram plot shows a slope of -0.14 (the regression line is shown), yielding an estimate of H as 0.83. Finally, the Whittle estimator for this revocation data

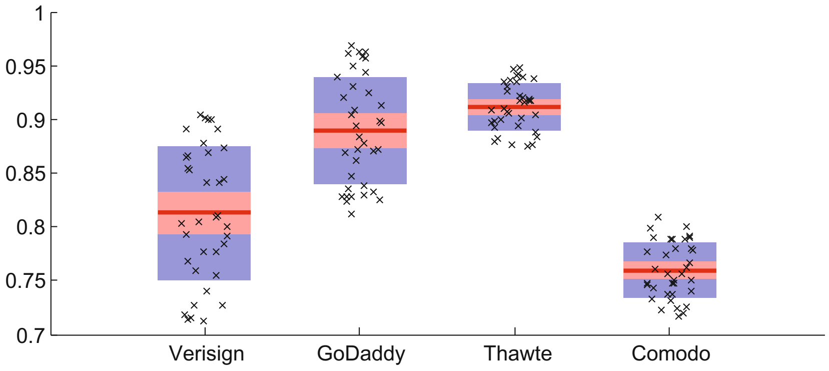


Fig. 5. Summary plot of estimates of the Hurst parameter H for all the CAs

(not a graphical method) yields an estimated Hurst value of 0,923 with a 95% confidence interval of $(0.87, 0.95)$.

Once we have seen that GoDaddy presents a significant self-similar pattern, we analyze the rest of the CAs. To that end, we use the whittle estimator to obtain the Hurst value per CA and month. We chose this estimator because it gives more refined measurement than other estimation techniques and it provides confidence levels for the Hurst parameter [8]. Note that we are not interested in estimating the exact value of the Hurst parameter but to prove the existence of self-similarity in the revocation data. Figure 5 shows the H parameter of each CA and the 95% confidence interval. It is worth noting that depending on the month there are some CAs whose H parameter varies significantly. However, no matter neither the CA nor the month, the Hurst value is always above 0.7. This means that the revocation process of any CA presents LRD.

4 Significance of Self-similarity for Revocation Data Management

Our collected data from real CAs show dramatically different statistical properties than those assumed by the stochastic models currently considered in the literature. Almost all these models are characterized by an exponentially decaying autocorrelation function. As a result, they give rise to a Hurst parameter estimate of $\hat{H} = .50$, producing variance-time curves, R/S plots, and frequency domain behavior strongly disagreeing with the self-similar behavior of actual revocation (see Section 3.3). In this section, we emphasize direct implications of the self-similar nature of the revocation data in the performance of the revocation service.

4.1 Impact on the Revocation Mechanism

As we mentioned before, traditional mechanisms made assumptions about the revocation process to obtain efficient revocation data issuing policies. However, these assumptions neglect the self-similar nature of the revocation data. This has a direct impact due to the "burstiness" of the data and affects the congestion management of the CA/repositories.

To give an idea of the impact of self-similarity, we analyze the work of Cooper in [9] and in [10]. In these works, Cooper analyzed the best way to issue CRLs, segmented CRLs and delta-CRLs in order to decrease the request peak bandwidth. The author assumed that an average of 1,000 certificates are revoked each day and that the CRLs have a fixed validity time. By doing these assumptions, the self-similar behavior of the revocation process is neglected and the results need to be adapted to the reality.

Using the traditional approach, CRLs are published periodically. Under this assumption, CAs expect that consecutive CRLs should have similar size. However, this assumption is proven completely wrong when bursts are present. Thus, consecutive CRLs can differ significantly in the number of revoked certificates they include, and, consequently in their size. Using the data collected from Verisign[2], we studied how the size of the CRLs varies when CRLs are issued daily. As in [10], we estimate that the size of a CRL is 51 bytes plus 9 bytes for each certificate included on the CRL. If an average of r certificates are revoked each day, certificates are valid for L_c days, and a certificate, at the time of revocation, has an average of $\frac{L_c}{2}$ days until it expires, then the average size of a CRL will be [10]:

$$Size_{CRL} = 51 + 4.5 \cdot r \cdot L_c.$$

We assume that certificates have a lifetime of 365 days [11], therefore we can calculate the daily size of the Verisign CRL for 5 randomly chosen months. We execute the trial several times and check that the same depency is obtained. Figure 6 shows the results in a box-plot. Note that the CRL size has a mean size of around 150 KBytes, but it highly varies due to the revocation bursts. For instance, during March 2008, there were four CRLs that exceed the 300 KBytes. These variations are highly inefficient in terms of bandwidth, as during some days the required bandwidth double the bandwidth needed in previous days. Although this has not become a bottleneck in wired networks, novel scenarios (e.g. Vehicular Networks) cannot afford these variations.

However, the self-similarity not only affects traditional CRL issuance, but also its variants that aim to be bandwidth efficient such as delta-CRL. From [10], the bandwidth for a delta-CRL system can be computed as:

$$B = \frac{Nve^{-vt}((51 + 4.5rL_c)e^{-(w+\frac{l}{O}-l)v} + (51 + 9rw))}{(O-1)1 - e^{vl/O} + 1}, \qquad (1)$$

where N is the number of valid certificates, v is the validation rate, l is the amount of time that a delta-CRL is valid, L_c is the certificate lifetime, r is the number of certificates revoked per day, w is the window size of the delta-CRL and O is the number of delta-CRLs that are valid at any given time.

[2] Note that we use the data from VeriSign to provide a case study of variance in size of the CRLs. The same variance pattern applies to the other CAs, though it is not shown in this article.

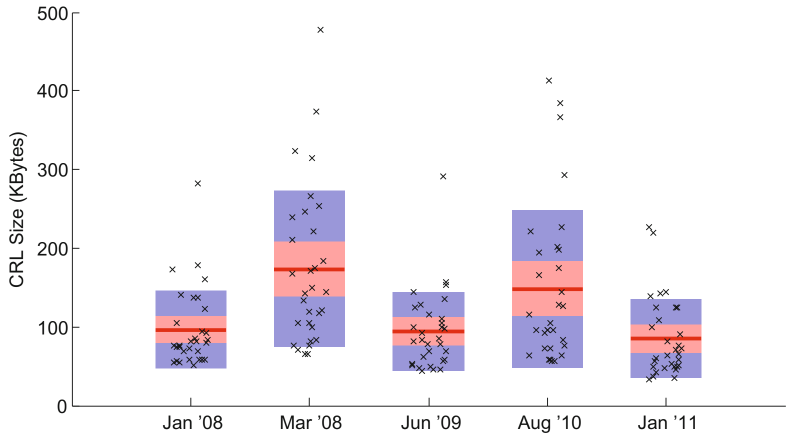


Fig. 6. Estimated daily size of Verisign's CRL

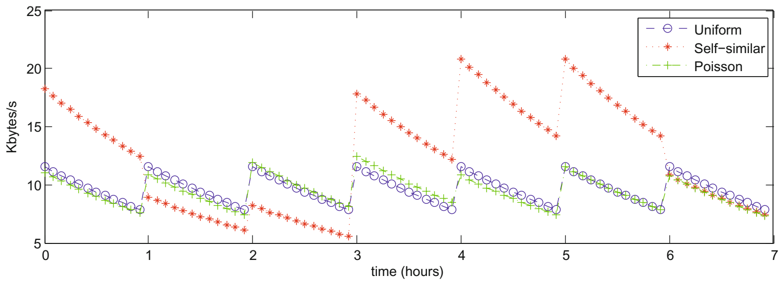


Fig. 7. Delta-CRL BW consumption

Using the bandwidth as a comparison metric, we can evaluate the impact of the self-similarity. Figure 7 shows the bandwidth necessary to download the revocation data using a sliding window delta-CRL scheme. We have assumed that there are 300,000 relying parties (N) each validating an average of 10 certificates per day (v); delta-CRLs are issued once an hour, are valid for 4 hours (O), and have a window size of 9 hours (w). We have also assumed that an average of 10 certificates are revoked each day (r) and that certificates are valid for 365 days (L_c). Note that depending on the distribution of the revocation process, the required bandwidth presents significant variations. We change the number of certificates revoked per day (r) according to three different distributions (i.e. uniform, Poisson and self-similar) and evaluate the required bandwidth of a delta-CRL system using Eq. (1). Uniform and Poisson distributions present a similar behavior. On the opposite, a self-similar process makes the delta-CRL's size to vary. Thus, the optimal window to issue delta-CRLs should be calculated taking into account the bursty pattern of the self-similar process. If this pattern is neglected, the peak bandwidth will vary with each delta-CRL issuance making the revocation service bandwidth-inefficient. When with a Poisson or uniform process the maximum peak bandwidth is of ∼12Kb/s, a burst of revocation events causes that some delta-CRL issuance require more than ∼20Kb/s. Therefore, ignoring the self-similar pattern of the revocation process leads to inaccurate network planning.

CRL releasing strategies might be optimized considering the effect of self-similarity. Periodic updates might create bottlenecks at the repositories when all users request new information at the same time. On the other hand, online checking mechanisms such as OCSP, could be computationally overloaded during bursty periods. Such mechanisms that base their efficiency on using pre-signed responses have not been conceived to work under bursty patterns. Therefore, further analysis should be conducted to establish pre-signing policies under bursty revocation periods.

5 Related Work

Most of previous studies fail to capture the characteristics of real-world revocation data; instead, they focus on theoretical aspects of certificate revocation including the model of revocation [9], the revocation cause [12], and the cost of issuing revocation information [13]. Thus, these theoretical models are not able to capture the actual pattern of the revocation data. Most recently, the statistical properties of real revocation data have been studied [14,15,16]. Nevertheless, the bursty pattern of the revocation process is neglected.

Regarding the traditional way of issuing CRLs, X.509 [17] defines one method to release CRLs. This method involves each CA periodically issuing CRLs. Using this method, the number of revoked certificates contained in each CRL varies significantly. Thus, each CRL has a different size, and the issuance of the CRLs results bandwidth inefficient. Authors in [15] already acknowledged the inefficiencies of the traditional method, and proposed releasing CRLs based on a set of economic costs. However, they assumed a Poisson process when characterizing the number of new certificate revocations, i.e., they neglected the burst pattern. Thus, the resulting CRL releasing policies could be improved by taking into account the self-similarity of the revocation process. Similarly, authors in [18] collected empirical data about the reasons and frequency of user terminations that require certificate revocations, and then model the consequences for certificate revocation. They investigate how to reduce the cost of certificate revocation by reducing the number of revoked certificates and bandwidth consumption in order to achieve better scalability.

In the same manner, authors in [14] carried out a thorough empirical analysis of the revocation data not only taking into account the number of revoked certificates, but also other factors such as geographical regions and revocation causes. They also conclude that their collected CRLs exhibit exponential distribution patterns. Though they acknowledge the existence of revocation bursts, they do not capture this behavior. On the other hand, authors in [16] suggest a functional form for the probability density function of certificate revocation requests. They choose an exponential distribution function because it adequately approximates the data they collected from a single CA. Based on this assumption, they provide an economic model based on which a CA can choose what they state to be the optimal CRL release interval. However, they do not take into account the self-similar behavior of the revocation data.

6 Conclusions

Current simulation studies for performance evaluation and revocation data release strategies most commonly assume that the temporal distribution of revocation events follows a Poisson distribution. In this paper, we questioned the assumption of Poisson distribution. Our analysis of the revocation data contained in different CRLs provides significant evidence that the real revocation events follow a self-similar distribution. In particular, our analysis showed burstiness at all time-scales, confirming scale-invariance of distribution. We also estimated and showed that Hurst parameter for the daily number of revoked certificates is above 0.5, proving the self-similarity and Long Range Dependence formally.

We then turned our attention to understanding its consequences on the performance of the revocation services. We showed that traditional revocation mechanisms, such as CRLs or delta-CRLs, do not take into account the bursty pattern of the revocation events when establishing the issuing strategies. These bursts increase the maximum peak bandwidth required to provide the revocation data timely. Thus, self-similarity has a profound effect on the engineering of traditional mechanisms and should be taking into account when designing new revocation protocols.

Acknowledgments. This work has been supported partially by the Spanish Research Council with Project TEC2011-26452 (SERVET), by Spanish Ministry of Science and Education with Project CONSOLIDER CSD2007-00004 (ARES), FPU grant AP2010-0244, by Generalitat de Catalunya with Grant 2009 SGR-1362 to consolidated research groups and with the support from the SUR (Secretaria d'Universitats i Recerca) of the DEC (Departament d'Economia i Coneixement), and by the European Social Funds.

References

1. Willinger, W., Paxson, V., Taqqu, M.S.: Self-similarity and heavy tails: structural modeling of network traffic, pp. 27–53 (1998)
2. Beran, J.: Statistics for Long-Memory Processes. Monographs on Statistics and Applied Probability. Chapman & Hall (1994)
3. Taqqu, M.S., Teverovsky, V., Willinger, W.: Estimators for long-range dependence: An empirical study. Fractals 3, 785–798 (1995)
4. Peng, C.K., Havlin, S., Stanley, H.E., Goldberger, A.L.: Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series. Chaos Woodbury Ny 5(1), 82–87 (1995)
5. Netcraft. Market share of certification authorities (2009), `https://ssl.netcraft.com/ssl-sample-report/CMatch/certs` (accessed on May 2011)

6. Jain, G.: Certificate revocation: A survey, http://csrc.nist.gov/pki/welcome.html (accessed on May 2011)
7. Karagiannis, T., Faloutsos, M., Riedi, R.H.: Long-range dependence: now you see it, now you don't. In: Proc. GLOBECOM 2002, pp. 2165–2169 (2002)
8. Leland, W.E., Taqqu, M.S., Willinger, W., Wilson, D.V.: On the self-similar nature of ethernet traffic (extended version). IEEE/ACM Trans. Netw. 2(1), 1–15 (1994)
9. Cooper, D.A.: A model of certificate revocation. In: Fifteenth Annual Computer Security Applications Conference, pp. 256–264 (1999)
10. Cooper, D.A.: A more efficient use of Delta-CRLs. In: 2000 IEEE Symposium on Security and Privacy. Computer Security Division of NIST, pp. 190–202 (2000)
11. Technological infrastructure for pki and digital certification. Computer Communications 24(14), 1460–1471 (2001)
12. Fox, B., LaMacchia, B.: Certificate Revocation: Mechanics and Meaning. In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 158–164. Springer, Heidelberg (1998)
13. Naor, M., Nissim, K.: Certificate Revocation and Certificate Update. IEEE Journal on Selected Areas in Communications 18(4), 561–570 (2000)
14. Walleck, D., Li, Y., Xu, S.: Empirical Analysis of Certificate Revocation Lists. In: Atluri, V. (ed.) DAS 2008. LNCS, vol. 5094, pp. 159–174. Springer, Heidelberg (2008)
15. Ma, C., Hu, N., Li, Y.: On the release of CRLs in public key infrastructure. In: Proceedings of the 15th Conference on USENIX Security Symposium, vol. 15, pp. 17–28 (2006)
16. Hu, N., Tayi, G.K., Ma, C., Li, Y.: Certificate revocation release policies. Journal of Computer Security 17, 127–157 (2009)
17. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection - The Directory: Autentication Frameworks, Technical Corrigendum (2000)
18. Ofigsbø, M.H., Mjølsnes, S.F., Heegaard, P., Nilsen, L.: Reducing the Cost of Certificate Revocation: A Case Study. In: Martinelli, F., Preneel, B. (eds.) EuroPKI 2009. LNCS, vol. 6391, pp. 51–66. Springer, Heidelberg (2010)

Author Index