

Abdessamad Imine · José M. Fernandez
Jean-Yves Marion · Luigi Logrippo
Joaquin Garcia-Alfaro (Eds.)

LNCS 10723

Foundations and Practice of Security

10th International Symposium, FPS 2017
Nancy, France, October 23–25, 2017
Revised Selected Papers

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, Lancaster, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Zurich, Switzerland

John C. Mitchell

Stanford University, Stanford, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

TU Dortmund University, Dortmund, Germany

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max Planck Institute for Informatics, Saarbrücken, Germany

More information about this series at <http://www.springer.com/series/7410>


Abdessamad Imine · José M. Fernandez
Jean-Yves Marion · Luigi Logrippo
Joaquin Garcia-Alfaro (Eds.)

Foundations and Practice of Security

10th International Symposium, FPS 2017
Nancy, France, October 23–25, 2017
Revised Selected Papers


Editors

Abdessamad Imine
University of Lorraine
Villers-lès-Nancy
France

José M. Fernandez 
Polytechnique de Montréal
Montreal, QC
Canada

Jean-Yves Marion
LORIA
Vandœuvre-lès-Nancy
France

Luigi Logrippo 
Université du Québec en Outaouais
Gatineau, QC
Canada

Joaquin Garcia-Alfaro 
Télécom SudParis
Evry Cedex
France

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-75649-3 ISBN 978-3-319-75650-9 (eBook)
<https://doi.org/10.1007/978-3-319-75650-9>

Library of Congress Control Number: 2018934322

LNCS Sublibrary: SL4 – Security and Cryptology

© Springer International Publishing AG, part of Springer Nature 2018

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Printed on acid-free paper

This Springer imprint is published by the registered company Springer International Publishing AG part of Springer Nature
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland



Preface

This volume contains the papers presented at the 10th International Symposium on Foundations and Practice of Security (FPS 2017), which was hosted by Lorraine Research Laboratory in Computer Science and Its Applications (LORIA), Nancy, France, during October 23–25, 2017.

FPS 2017 attracted 53 submissions. At least three reviews were made for each submitted paper. The decision on acceptance or rejection in the review process was completed after intensive discussions over a one-week period. The Program Committee accepted 17 full research papers and three short research papers for presentation. The selected papers deal with diverse research themes, ranging from classic topics, such as access control models, formal verification for secure protocols and network security to emerging issues, such as security in blockchain and encrypted databases.

The best paper award of FPS 2017 was awarded to the contribution “Defending Against Adversarial Attacks Using Statistical Hypothesis Testing” presented by Sunny Raj, Sumit Kumar Jha, Laura Pullum, and Arvind Ramanathan. The program was completed with three excellent invited talks given by Véronique Cortier (LORIA-CNRS, France), Krishna Gummedi (Max Planck Institute for Software Systems, Germany), and Florian Kerschbaum (University of Waterloo, Canada).

Many people contributed to the success of FPS 2017. First, we would like to thank all the authors who submitted their research results. The selection was a challenging task and we sincerely thank all the Program Committee members, as well as the external reviewers, who volunteered to read and discuss the papers. We greatly thank the general chairs, Luigi Logrippo (Université du Québec en Outaouais, Canada) and Jean-Yves Marion (Mines de Nancy, France), and the local organizer, Abdessamad Imine, for the great efforts to organize and perfectly manage the logistics during the symposium. Finally, we also want to express our gratitude to the publication chair, Joaquin Garcia-Alfaro (Télécom SudParis), for his work on editing the proceedings. Last but not least, thanks to all the attendees. As security becomes an essential property in the information and communication technologies, there is a growing need to develop efficient methods to analyze and design systems providing a high level of security and privacy. We hope the articles in this proceedings volume will be valuable for your professional activities in this area.

November 2017

José M. Fernandez
Abdessamad Imine
Luigi Logrippo
Jean-Yves Marion

Organization

General Chairs

Luigi Logrippo Université du Québec en Outaouais, Canada
Jean-Yves Marion Mines de Nancy, France

Program Co-chairs

José M. Fernandez Polytechnique Montréal, Canada
Abdessamad Imine Université de Lorraine, Nancy, France

Publications Chair

Joaquin Garcia-Alfaro Télécom SudParis, France

Local Organizing Committee

Abdessamad Imine Université de Lorraine, Nancy, France

Publicity Chairs

Pascal Lafourcade Université d'Auvergne, France
Raphaël Khoury Université du Québec à Chicoutimi, Canada

Program Committee

Esma Aimeur University of Montreal, Canada
Jeremy Clark Concordia University, Canada
Frédéric Cuppens IMT Atlantique, France
Nora Cuppens IMT Atlantique, France
Jean-Luc Danger Télécom Paris-Tech, France
Mourad Debbabi Concordia University, Canada
Josée Desharnais Laval University, Canada
Josep Domingo-Ferrer Universitat Rovira i Virgili, Spain
Samuel Dubus NOKIA Bell Labs, France
Sébastien Gambs Université du Québec à Montréal, Canada
Joaquin Garcia-Alfaro Telecom SudParis, France
Dieter Gollmann Hamburg University of Technology, Germany
Sushil Jajodia George Mason University, USA
Martin Johns SAP Research, Germany
Bruce Kapron University of Victoria, Canada
Nizar Kheir THALES, France

Raphaël Khoury	Université du Québec à Chicoutimi, Canada
Hyoungshick Kim	Sungkyunkwan University, Republic of Korea
Igor Kottenko	SPIIRAS, Russia
Evangelos Kranakis	Carleton University Computer Science, Canada
Pascal Lafourcade	Université d'Auvergne, France
Luigi Logrippo	Université du Québec en Outaouais, Canada
Javier Lopez	University of Malaga, Spain
Jean-Yves Marion	Mines de Nancy, France
Fabio Martinelli	National Research Council of Italy, CNR, Italy
David Mentré	Mitsubishi Electric R&D Centre Europe, France
Paliath Narendran	University at Albany, USA
Guillermo Navarro-Arribas	Universitat Autònoma de Barcelona, Spain
Jun Pang	University of Luxembourg, Luxembourg
Marie-Laure Potet	VERIMAG, France
Silvio Ranise	FBK, Security and Trust Unit, Italy
Indrakshi Ray	Colorado State University, USA
Michaël Rusinowitch	LORIA-Inria Nancy, France
Basit Shafiq	Lahore University of Management Sciences, Pakistan
Anna Squicciarini	Pennsylvania State University, USA
Natalia Stakhanova	University of New Brunswick, Canada
Chamseddine Talhi	École de Technologie Supérieure, Canada
Nadia Tawbi	Université Laval, Canada
Rakesh Verma	University of Houston, USA
Lingyu Wang	Concordia University, Canada
Edgar Weippl	SBA Research, Austria
Lena Wiese	Georg-August Universität Göttingen, Germany
Xun Yi	RMIT University, Australia
Nur Zincir-Heywood	Dalhousie University, Canada
Mohammad Zulkernine	Queen's University, Canada

Additional Reviewers

Zumrut Akcam	University at Albany, USA
Saed Alrabae	Concordia University, Canada
Carles Anglès	Universitat Rovira i Virgili, Spain
Andrew Bedford	Université Laval, Canada
Bruhadeshwar Bezawada	Colorado State University, USA
Alexander Branitskiy	SPIIRAS, Russia
Wenyaw Chan	University of Texas Health Science Center - Houston, USA
Andrey Chechulin	SPIIRAS, Russia
Vincent Cheval	LORIA-Inria Nancy, France
Mónica Del Carmen Muñoz	Universitat Rovira i Virgili, Spain

Luis Miguel Del Vasto	Universitat Rovira i Virgili, Spain
Vasily Desnitsky	SPIIRAS, Russia
Lena Doynikova	SPIIRAS, Russia
Jannik Dreier	Université de Lorraine, Nancy, France
Arthur Dunbar	University of Houston, USA
Gerardo Fernandez	University of Malaga, Spain
Carmen Fernandez-Gago	University of Malaga, Spain
David Gérard	Université d'Auvergne, France
Sanjay Goel	University at Albany, USA
Daniel Homann	Georg-August Universität Göttingen, Germany
Amrit Kumar	National University of Singapore, Singapore
Amirreza Masoumzadeh	University at Albany, USA
Maxime Puy	VERIMAG, France
Veena Ravishankar	University at Albany, USA
Sara Ricci	Universitat Rovira i Virgili, Spain
Igor Saenko	SPIIRAS, Russia
Paria Shirani	Concordia University, Canada
Wojciech Widel	IRISA, Rennes, France

Steering Committee

Frédéric Cuppens	IMT Atlantique, France
Nora Cuppens-Bouahia	IMT Atlantique, France
Mourad Debbabi	University of Concordia, Canada
Joaquin Garcia-Alfaro	Télécom SudParis, France
Evangelos Kranakis	Carleton University, Canada
Pascal Lafourcade	Université d'Auvergne, France
Jean-Yves Marion	Mines de Nancy, France
Ali Miri	Ryerson University, Canada
Rei Safavi-Naini	Calgary University, Canada
Nadia Tawbi	Université Laval, Canada

Contents

Access Control

- Attribute-Based Encryption as a Service for Access Control
in Large-Scale Organizations 3
*Johannes Blömer, Peter Günther, Volker Krummel,
and Nils Löken*
- Relationship-Based Access Control for Resharing in Decentralized
Online Social Networks 18
Richard Gay, Jinwei Hu, Heiko Mantel, and Sogol Mazaheri
- Secure Protocol of ABAC Certificates Revocation and Delegation. 35
Alexey Rabin and Ehud Gudes

Formal Verification

- Formal Analysis of Combinations of Secure Protocols 53
Elliott Blot, Jannik Dreier, and Pascal Lafourcade
- Formal Analysis of the FIDO 1.x Protocol 68
Olivier Pereira, Florentin Rochet, and Cyrille Wiedling
- A Roadmap for High Assurance Cryptography 83
Harry Halpin

Privacy

- Privacy-Preserving Equality Test Towards Big Data 95
Tushar Kanti Saha and Takeshi Koshiba
- Multi-level Access Control, Directed Graphs and Partial Orders
in Flow Control for Data Secrecy and Privacy 111
Luigi Logrippo

Physical Security

- Generation of Applicative Attacks Scenarios Against Industrial Systems 127
Maxime Puy, Marie-Laure Potet, and Abdelaziz Khaled

HuMa: A Multi-layer Framework for Threat Analysis in a Heterogeneous Log Environment 144
Julio Navarro, Véronique Legrand, Sofiane Lagraa, Jérôme François, Abdelkader Lahmadi, Giulia De Santis, Olivier Festor, Nadira Lammari, Fayçal Hamdi, Aline Deruyver, Quentin Goux, Morgan Allard, and Pierre Parrend

Monitoring of Security Properties Using BeepBeep 160
Mohamed Recem Boussaha, Raphaël Houry, and Sylvain Hallé

Network Security, Encrypted DBs and Blockchain

More Lightweight, yet Stronger 802.15.4 Security Through an Intra-layer Optimization. 173
Konrad-Felix Krentz, Christoph Meinel, and Hendrik Graupner

ObliviousDB: Practical and Efficient Searchable Encryption with Controllable Leakage 189
Shujie Cui, Muhammad Rizwan Asghar, Steven D. Galbraith, and Giovanni Russello

Ethereum: State of Knowledge and Research Perspectives 206
Sergei Tikhomirov

Vulnerability Analysis and Deception Systems

Bounding the Cache-Side-Channel Leakage of Lattice-Based Signature Schemes Using Program Semantics 225
Nina Bindel, Johannes Buchmann, Juliane Krämer, Heiko Mantel, Johannes Schickel, and Alexandra Weber

Extinguishing Ransomware - A Hybrid Approach to Android Ransomware Detection 242
Alberto Ferrante, Mirosław Malek, Fabio Martinelli, Francesco Mercaldo, and Jelena Milosevic

Deception in Information Security: Legal Considerations in the Context of German and European Law 259
Daniel Fraunholz, Christoph Lipps, Marc Zimmermann, Simon Duque Antón, Johannes Karl Martin Mueller, and Hans Dieter Schotten

Defence Against Attacks and Anonymity

SATYA: Defending Against Adversarial Attacks Using Statistical Hypothesis Testing 277
Sunny Raj, Laura Pullum, Arvind Ramanathan, and Sumit Kumar Jha

Attack Graph-Based Countermeasure Selection Using a Stateful Return on Investment Metric. 293
Gustavo Gonzalez-Granadillo, Elena Doynikova, Igor Kotenko, and Joaquin Garcia-Alfaro

Weighted Factors for Evaluating Anonymity. 303
Khalid Shahbar and A. Nur Zincir-Heywood

Author Index 319

Access Control



Attribute-Based Encryption as a Service for Access Control in Large-Scale Organizations

Johannes Blömer¹, Peter Günther², Volker Krummel², and Nils Löken¹(✉)

¹ Paderborn University, Paderborn, Germany

{johannes.bloemer, nils.loeken}@uni-paderborn.de

² Diebold Nixdorf, Paderborn, Germany

{peter.guenther, volker.krummel}@dieboldnixdorf.com

Abstract. In this work, we propose a service infrastructure that provides confidentiality of data in the cloud. It enables information sharing with fine-grained access control among multiple tenants based on attribute-based encryption. Compared to the standard approach based on access control lists, our encryption as a service approach allows us to use cheap standard cloud storage in the public cloud and to mitigate a single point of attack. We use hardware security modules to protect long-term secret keys in the cloud. Hardware security modules provide high security but only relatively low performance. Therefore, we use *attribute-based encryption with outsourcing* to integrate hardware security modules into our micro-service oriented cloud architecture. As a result, we achieve elasticity, high performance, and high security at the same time.

1 Introduction

Cloud computing [9] has become a popular computing model that allows enterprises to outsource their IT infrastructure. Resource pooling, multi-tenancy, and elasticity enable high availability of services at low costs. A major concern with cloud computing is security [4]. Particularly, a Cloud Service Provider (CSP) should not learn confidential information from the data it stores or processes. Such restrictions are imposed by data privacy laws, industry-specific regulation and standards, or by companies willing to outsource data but worried about the confidentiality of their secrets.

Many solutions for protecting data confidentiality in the cloud have emerged over time. Attribute-Based Encryption (ABE) has often been proposed as a solution, but is not deployed in practice. Solutions not based on ABE are already deployed. Cloud storage providers, e.g. Dropbox,¹ encrypt data before storage

This work was partially supported by the Federal Ministry of Education and Research (BMBF) within the collaborate research project Securing the Financial Cloud (SFC), grant 16KIS0058K, and the German Research Foundation (DFG) within the Collaborative Research Centre On-The-Fly-Computing (SFB 901).

¹ <https://www.dropbox.com>.

ensuring confidentiality from outsiders. Since the provider encrypts the data, confidentiality against insiders is not guaranteed. Protection from insiders can be achieved by client-side encryption, e.g. Cryptomator,² or encryption as a service, e.g. Ciphercloud.³

The latter two solutions are sufficient for individuals, but fall short of large organizations' needs. Typically, large organizations have members with heterogeneous rights to access data with rather complex policies describing such access rights in a fine-grained manner. Both, encryption as a service and client-side encryption do not provide fine grained access control. Additionally, with encryption as a service the encryption provider has access to plaintext data, requiring trust in the provider. Client side-encryption partially negates the benefits of resource pooling in the cloud.

Our contribution. In this paper we present a novel approach to fine-grained access control as a service. Our solution leverages the cloud's resources, while our design ensures confidentiality of data from the CSP. It is based on three components that are integrated into a system for access control as a service. Particularly, we have (1) a cloud-based service for Access Control with Encryption (ACE), (2) a cloud design that complements our ACE service, and (3) an infrastructure for identity and key management.

Our ACE service achieves fine-grained access control via Attribute-Based Encryption (ABE) with outsourcing as defined in [7]. We exploit outsourcing to design a cloud infrastructure complementing ACE such that ACE can take advantage of the cloud's resources to achieve efficiency. At the same time, our solution ensures confidentiality of data even against insider attacks. We furthermore show how to adapt the existing infrastructure for identity and key management of large-scale organizations to the specifics of access control via ABE. This enables user revocation without re-encryption through the application of standard mechanisms.

Related work and discussion. Our approach to ACE is based on ABE [1,12] with outsourcing, as introduced by Green et al. [7]. In [7] outsourcing enables mobile devices to perform ABE decryption by limiting their computations to the security critical part of the decryption, while the computationally expensive but non-critical part is performed by the cloud. We use this mechanism within the cloud to execute the security critical part of decryption on so-called hardware security modules, thus achieving high security. We use standard cloud services for non-critical computations, thus achieving efficiency.

Numerous papers identify issues with ABE that hinder its deployment in the cloud, e.g. [14–18]. Issues include that (1) data integrity is not protected, (2) fine-grained access control for write access is typically ignored, and (3) user's access cannot be revoked once granted. Zhao et al. [18] suggest attribute-based signatures to address the first two issues. This proposal is rather generic and can also be applied in our scenario. However, this is beyond the scope of this paper.

² <https://cryptomator.org>.

³ <https://ciphercloud.com>.

Means to revoke users' access rights are proposed in numerous papers [14–17]. These papers specifically suggest modifications to ABE schemes that allow user's access rights to be revoked. Particularly, they propose re-encrypting all ciphertexts that the revoked user had access to, as well as updating all the remaining users' cryptographic keys. This approach is very costly for all parties involved, especially for the party who re-encrypts the ciphertexts. Therefore, additional features are suggested to reduce the load of some parties. For example, Yang et al. [15] and Yu et al. [16] suggest to delay the re-encryption of a particular ciphertext until some user actually requests the ciphertext. Yu et al. additionally use a technique similar to outsourcing [7] in order to reduce users' loads when updating their keys: only the non-critical parts of the users' keys need to be updated, so key updates can be applied by a cloud server.

Achieving outsourced decryption [7] and user revocation, the scheme of Zhang et al. [17] at a first glance looks most similar to our approach to access control with encryption in the cloud. They consider the cloud's structure when outsourcing computations: decryption is outsourced to fog nodes, i.e. cloud resources close (e.g. in a geographical sense) to the user. Hence, Zhang et al. increase ABE's efficiency using cloud resources, but in contrast to us, do not fully deploy ABE in the cloud in a secure way.

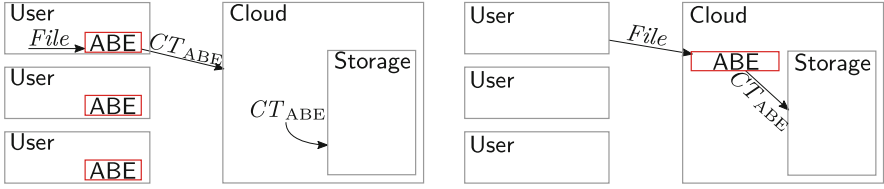
Paper organization. In Sect. 2 we discuss how ABE can be used for access control in the cloud. Section 3 gives a brief introduction to ABE and some of its properties. Section 4 presents ACE and its complementing cloud design. In Sect. 5 we present the infrastructure for identity and key management. Section 6 presents proofs of concept concerning the ABE schemes underlying ACE and the core technologies that we have used to implement ACE. Finally, the paper is concluded in Sect. 7.

2 Approaches to ABE for Cloud Infrastructures

As a starting point, we consider two basic approaches for realizing access control in the cloud via ABE. In the *security-oriented approach* (see Fig. 1a), we store ABE encrypted data at a Cloud Service Provider (CSP). Encryption and decryption of the data is handled by the user. In the *service-oriented approach* (see Fig. 1b), we also store ABE encrypted data at a CSP, but encryption and decryption of the data is handled by an encryption service in the public cloud on the user's behalf.

In the security-oriented method, users do not have to trust the CSP, because all data is encrypted by the users. Since users provide own resources for ABE, this approach does not benefit from resource pooling and elasticity in the cloud. This method also prohibits ABE's tight integration into complex applications where many services process the decrypted data.

The service-oriented approach offers resource pooling and elasticity. Furthermore, it can be coupled with services for data processing. However, this method requires users to ultimately trust the CSP.



(a) Security-oriented approach: ABE performed by users. (b) Service-oriented approach: ABE performed in the cloud.

Fig. 1. ABE performed in a cloud environment.

In this work, we enhance the service-oriented approach with mechanisms to obtain security similar to the security-oriented method while preserving the elasticity of the service-oriented approach. To achieve this, we partition the computations of ABE encryption and decryption into sub-services according to their security and elasticity requirements.

3 Attribute Based Encryption with Outsourcing

We describe ABE-OS-KEM, a primitive that underlies the architecture for our ACE service. We also cover security guarantees and requirements.

3.1 ABE-OS-KEM

Access control can be cryptographically enforced by Attribute-Based Encryption (ABE) [1]. In ABE, ciphertexts are associated with access structures. Access structures represent Boolean formulas consisting only of AND and OR operators. Users hold attributes, e.g. role descriptions, represented by secret keys. Attribute sets represent interpretations of the variables in Boolean formulas, and satisfy an access structure if their interpretations satisfy the policy’s formula. Only attributes satisfying a ciphertext’s policy are able to decrypt that ciphertext.

Our approach to protect the confidentiality of outsourced data follows the KEM/DEM paradigm [8, Chap.11.3]. The Data Encapsulation Mechanism (DEM) with algorithms (Encrypt , Decrypt) takes keys from some key space \mathcal{K} . Our Key Encapsulation Mechanism (KEM) is based on ABE:

Definition 1. *An ABE-KEM consists of algorithms*

Setup: *given a security parameter Λ , output public parameters pub_{ABE} and a master secret msk_{ABE} .*

Keygen: *given pub_{ABE} , msk_{ABE} and an attribute set A_{ID} , output a user key $\text{sk}_{ABE, ID}$.*

Encaps: *given pub_{ABE} and a policy \mathbb{A} , output a symmetric key $k \in \mathcal{K}$ and a ciphertext CT_{ABE} .*

Decaps: *given pub_{ABE} , $\text{sk}_{ABE, ID}$ and ciphertext CT_{ABE} , output a symmetric key k .*

We require for all correctly set up systems, policies \mathbb{A} , tuples $(k, CT_{ABE}) \leftarrow \text{Encaps}(pub_{ABE}, \mathbb{A})$, and user keys $sk_{ABE, ID}$, if the attributes in $sk_{ABE, ID}$ satisfy \mathbb{A} then $\text{Decaps}(pub_{ABE}, sk_{ABE, ID}, CT_{ABE}) = k$.

Typically, the Decaps algorithm is computationally expensive [7]. Therefore, [7] splits the Decaps algorithm into a computationally expensive part that is not security critical, and a security critical part that is rather efficient. Splitting Decaps yields a variant of ABE-KEM that we call ABE-OS-KEM, as it allows to partially outsource computations to the cloud.

Definition 2. An ABE-OS-KEM consists of the following algorithms

Setup, Keygen and Encaps, are as in Definition 1.

TransKey: given pub_{ABE} and $sk_{ABE, ID}$, output a transformation key tk_{ID} and a decapsulation key $sk_{PK, ID}$.

Transform: given pub_{ABE} , tk_{ID} and CT_{ABE} , output partially decrypted ciphertext CT_{PK} .

Decaps: given pub_{ABE} , $sk_{PK, ID}$ and CT_{PK} , output a symmetric key k .

We require for all correctly set up systems, policies \mathbb{A} , tuples $(k, CT_{ABE}) \leftarrow \text{Encaps}(pub_{ABE}, \mathbb{A})$, user keys $sk_{ABE, ID}$, and key pairs $(tk_{ID}, sk_{PK, ID}) \leftarrow \text{TransKey}(pub_{ABE}, sk_{ABE, ID})$ if $sk_{ABE, ID}$ satisfies \mathbb{A} then

$$\text{Decaps}(pub_{ABE}, sk_{PK, ID}, \text{Transform}(pub_{ABE}, tk_{ID}, CT_{ABE})) = k.$$

The original Decaps algorithm is split into two algorithms: Transform and Decaps. The additional algorithm TransKey is required to produce separate keys required by algorithms Transform and Decaps.

For our proof of concept implementation, we constructed an ABE-OS-KEM based on the ciphertext-policy ABE scheme of Rouselakis and Waters [11] and the outsourcing technique of Green et al. [7]. We present our ABE-OS-KEM in Sect. 6.1.

3.2 Security of ABE-OS-KEM and Protection of Keys

For the security of our ABE-OS-KEM, we adopt the notion of security against replayable chosen-ciphertext attacks (RCCA, see [7]): no reasonable adversary learns any information about the key encapsulated in a certain ciphertext, nor can that key be modified unnoticeably. These properties even hold for adversaries with access to arbitrary transformation keys.

In an ABE-OS-KEM, we consider several types of keys for decryption. Keys are categorized as critical and non-critical for security. Transformation keys (tk_{ID}) are non-critical. Critical keys are further classified as user-specific and file-specific. Keys $sk_{ABE, ID}$ and $sk_{PK, ID}$ are user-specific, while the data encapsulation mechanism's symmetric keys are file-specific.

Knowledge of a *user key* allows decryption of ciphertexts with policies satisfied by the user key. Hence, such a key is security critical. An ABE-OS-KEM

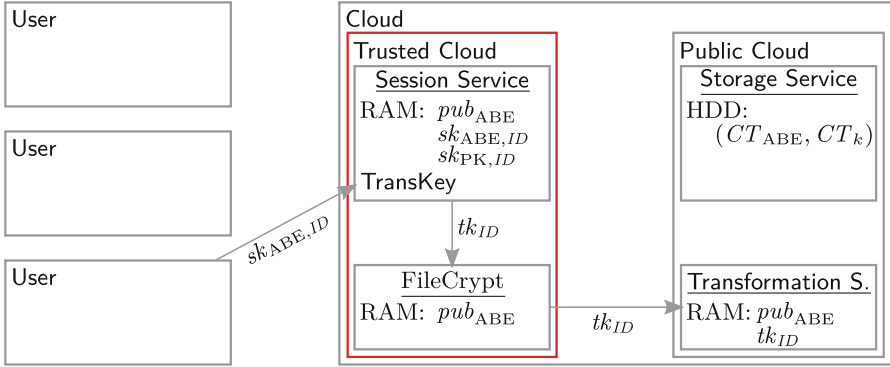


Fig. 2. The interaction during session initialization and the sub-services’ internal states after initialization. The user sets up the session with the session service that computes a transformation key given to FileCrypt, which relays the transformation key to the transformation service.

being a KEM, each file is encrypted under an individual *symmetric key* k . Thus, if a symmetric key leaks, the corresponding file leaks. Thus, also the symmetric keys have to be considered as critical to security. In an ABE-OS-KEM, *transformation keys* tk_{ID} are meaningless without their respective *decapsulation keys* $sk_{PK, ID}$. Together the transformation keys and decapsulation keys have the same decapsulation capabilities as the user keys they are derived from. The order in which operation Transform and Decaps are performed then give the classification of transformation keys to be non-critical and decapsulation keys to be critical and user-specific.

4 Access Control with Encryption and Cloud Design

In this section, we describe our service for access control with encryption in the cloud (ACE). Our goal is a service that provides the security of the security-oriented approach from Sect. 2 and the elasticity of the service-oriented approach. We first split ACE into sub-services based on our ABE-OS-KEM and its keys. Then we present our model of a cloud. Subsequently, we match ACE sub-services to the components of our cloud design. Finally, we discuss our approach in terms of security and elasticity.

4.1 Access Control with Encryption via ABE

We aim to realize ACE via ABE using our ABE-OS-KEM. As discussed in Sect. 3.2, several categories of keys exist in an ABE-OS-KEM. For each of the categories, we establish a separate sub-service: a *session service* handling user-specific keys, a *FileCrypt* service for file-specific keys, and a *transformation service* handling non-critical keys. Hence, the session service implements algorithms

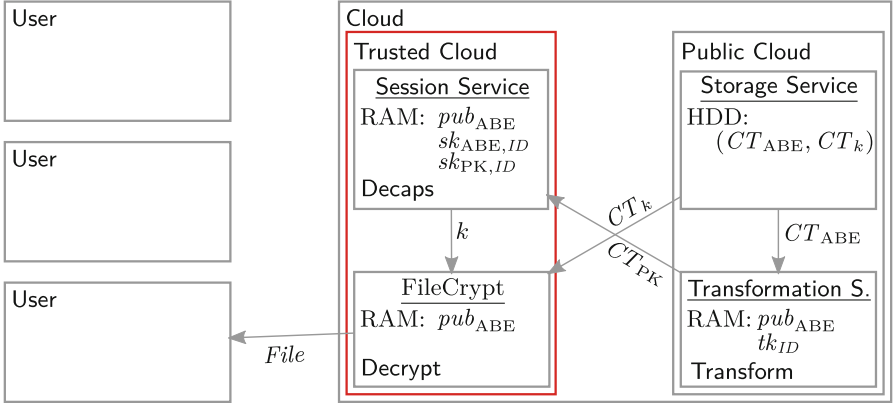


Fig. 3. Interactions during file access. ABE ciphertext CT_{ABE} is transformed into partially decrypted ciphertext CT_{PK} by the transformation service using the transformation key. The session service performs a **Decaps** operation on CT_{PK} using the decapsulation key $sk_{PK,ID}$ and obtains k which is used by **FileCrypt** to decrypt CT_k . The user receives the result.

TransKey and **Decaps**, the transformation service implements **Transform** and **FileCrypt** implements **Encaps** as well as algorithms **Encrypt** and **Decrypt** of the data encapsulation mechanism.

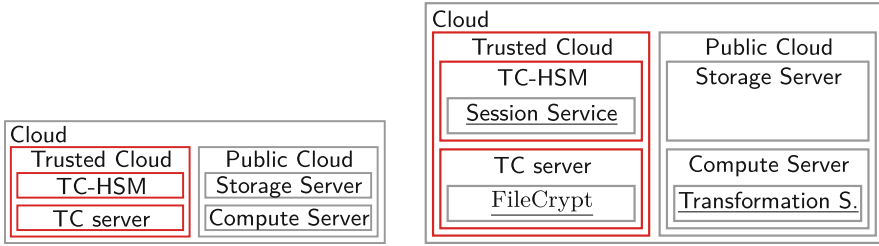
We further explore our services' interactions. Particularly, we discuss how users set up a session with ACE and access files. We omit file uploads, because they only involve the **FileCrypt** service.

The interactions for setting up a session of ACE are shown in Fig. 2. The session service uses the user's key $sk_{ABE,ID}$ to compute transformation key tk_{ID} and decapsulation key $sk_{PK,ID}$ via algorithm **TransKey**. While the service keeps the $sk_{PK,ID}$ secret, tk_{ID} is handed over to **FileCrypt** and forwarded to the transformation service in the public cloud.

For decryption, the sub-services interact as shown in Fig. 3. When the user requests an encrypted file, stored as (CT_{ABE}, CT_k) in cloud storage, the transformation service receives a copy of CT_{ABE} and applies algorithm **Transform**. The resulting CT_{PK} is given to the session service. The session service applies the **Decaps** algorithm and gives the obtained symmetric key k to **FileCrypt**. Applying the data encapsulation mechanism's **Decrypt** algorithm to k and CT_k received from the cloud's storage yields the plaintext data *File* that is given to the user.

4.2 Cloud Design for ACE

As described in Sect. 2, our architecture for access control in the cloud aims at realizing strong security as if access control and decryption were performed at the user while leveraging the cloud's computational resources for those tasks. In order to achieve this goal, our cloud model must reflect the security requirements



(a) The layout of our cloud: a trusted (b) The mapping of ACE sub-services to the components of our cloud based on security considerations.

Fig. 4. The components of our cloud and their relation to ACE.

imposed by the sub-services of ACE. Figure 4a presents our cloud model. The design considers storage and computational resources of the public cloud and complements them with a trusted cloud, which consists of two components, a TC server and a TC-HSM.

A *TC server* is a server with dedicated technical and organizational measures like remote attestation and memory encryption to protect the integrity of executed software and to protect the confidentiality of processed data. A *TC-HSM* is a tamper-resistant Hardware Security Module (HSM) that provides a restricted and well-defined Application Programming Interface (API). An HSM protects integrity and confidentiality of stored data also against attackers with physical access to the device. Furthermore, the TC-HSM API supports run-time initialization with user keys.

On the one hand, the TC server provides high computational power and is able to dynamically assign its resources to executed services based on their workload. On the other hand, the TC server is unable to protect the confidentiality of data against adversaries with physical access. This protection is provided by a TC-HSM at the cost of restricted flexibility and power. In consequence, the components of our trusted cloud achieve security by different means and thus establish different levels of security. Both the TC-HSM and the TC server are secure. However, its stronger guarantees make the TC-HSM fit to serve as a trust anchor for our cloud.

4.3 Security, Execution of Services

Our division of ABE-OS-KEM's algorithms into services reflects the security levels required by the algorithms based on the threat that exposing their key inputs poses. Thus, the assignment of services to the components of our cloud model must also reflect the expected levels of security. As a result, we map our ACE sub-services, which implement the ABE-OS-KEM, to the components of our cloud as shown in Fig. 4b.

As discussed in Sect. 4.2, the TC-HSM provides the highest level of security. It is thus fit to run the session service that works on security critical user specific secrets. The TC server provides a level of security that is sufficient to run the FileCrypt service that operates on file-specific secrets. The public cloud’s compute servers provide no security guarantees, so they may only operate on non-critical keys. The transformation service can be run on such servers. For this assignment, the security needed by the three categories of ABE-OS-KEM keys, and thus our services, match the three levels of security provided by the components of our cloud model.

4.4 Elasticity

The partitioning of our ACE service into sub-services does not only reflect the sensitivity of keys. It also supports elasticity because each sub-service can be scaled individually. This provides four dimensions of elasticity: elasticity with respect to data storage, simultaneous access to multiple files, the complexity of access policies, and the number of active users.

Data is always stored encrypted at the storage server in the public cloud. Hence, storage can be dynamically (de-)provisioned based on the needed amount, the expected reliability, and the acceptable latency.

For simultaneous processing of multiple files, we distinguish between encryption (write access) and decryption (read access). Encryption of data does not involve user keys and is performed on the TC server by the FileCrypt service. Depending on the number of files to encrypt, the TC server provisions resources for the FileCrypt service based on standard load balancing mechanisms. Decryption of files additionally involves the transformation service and the session service. The transformation service is hosted in the public cloud with high elasticity. The session service is executed on the TC-HSM with limited resources, but due to the initialization (see Sect. 4.1), additional TC-HSMs can temporarily be initialized.

For decrypting files with complex access policies, we benefit from the ABE-OS-KEM with a separated transformation and decapsulation step. The complexity of Decaps at the constrained TC-HSM is independent of the policy. Hence, additional resources for complex policies only need to be provisioned for Transform that is executed in the public cloud.

Finally, the number of active users determines the amount of provisioned HSMs for executing the session service. Resources for the FileCrypt and transformation service are provisioned during active encryption or decryption and are freed while no data is being accessed.

5 Infrastructure for Identity and Key Management

In this section, we describe identity management, rights management, and key management for our ABE-based ACE service. Identity management provides entities like users or hardware components with an identity and revokes

identities. Rights management is the assignment of access rights, respectively attributes, to identities according to their roles. Key management is the technical task of enforcing those rights using ABE and includes key generation, key storage, and key revocation.

5.1 Tasks

Identity and rights management. As a building block, our system uses a classical Public Key Infrastructure (PKI) with a root certificate and a corresponding Certification Authority (CA). In a technical sense, an identity ID is a public key with a certificate. The CA provides an entity with an identity by issuing a certificate for the entity’s public key. Standard mechanisms like revocation lists (see [10, Chap. 13]) are used to revoke an identity. An entity can prove its identity or establish a secure channel with another entity based on its certificate.

An identity possesses a set of rights according to its role. In our case, these rights correspond to a set of ABE-attributes (see Sect. 3). We encode these attributes into the certificate of the identity during certificate generation. This allows entities to check the rights of an identity.

ABE key storage. For our system, we apply client-side key management to put the individual organizations in control of decryption keys (see [4]). Therefore, in our system, each organization operates a dedicated service for ABE key storage. The ABE decryption key $sk_{\text{ABE},ID}$ of the user with identity ID is then stored at the key storage of the user’s organization.

Based on the key storage, we modify the initialization of our ACE service from Sect. 4.1 (see Fig. 5, compare Fig. 2). No user can have direct access to her key. Instead, the user with identity ID authenticates at the key storage to obtain a ticket for her key. The ticket is only granted to the user if she has not been revoked by the CA. The user forwards the ticket to the session service, who, via an authenticated channel, presents the ticket to the key store and obtains the user’s key $sk_{\text{ABE},ID}$ in return.

ABE key generation. For the generation of ABE user keys, we operate a global key generation service that has access to the ABE master secret key. Take note that the creation of user secret keys via algorithm KeyGen is independent of any user identifier. Thus, organizations can request user keys from the key generation service without providing user identifiers. Organizations can store the obtained user secret keys in their respective key storages and bind the keys to users later on. Note that the key generation service itself is beyond our considerations.

Our system explicitly supports sharing of data between identities of different organizations. Therefore, the ABE master secret key msk_{ABE} and the corresponding public parameters pub_{ABE} (see Definitions 1 and 2) are used globally for all participating organizations.

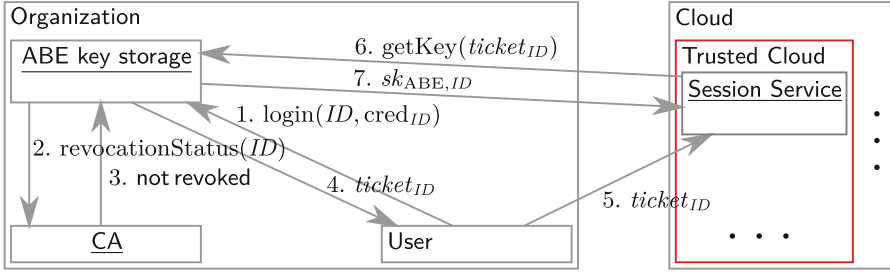


Fig. 5. ACE initialization with key storage: The user ID uses her credentials (e.g. password) to log in at the key storage, which checks the user’s revocation status. If the user is not revoked, she is given a ticket $ticket_{ID}$ that she presents to the ACE session service. The session service uses the ticket to obtain the user’s ABE key from the key store.

5.2 Discussion

Our system is *dynamic*, i.e. it is possible to add and revoke entities, as well as changing entities’ access rights. Our system also provides *separation of duties* among multiple parties. It distinguishes the technical aspects of key management from the administrative task of rights management, and it splits responsibilities between participating organizations.

Dynamic. To *add users* to the system, we generate an identity ID consisting of a public/private key pair with a PKI-based certificate. New identities are provided with secret ABE decryption keys according to their role and corresponding attributes. To *revoke/remove users* from the system we revoke their ID based on classical revocation mechanisms of the PKI. This effectively revokes the user from the ACE service because it prevents access to the user’s secret key $sk_{ABE,ID}$ at the key storage. Hence, we implicitly add a key revocation mechanism to ABE by combining ABE with classical PKI and a dedicated ABE key storage. *Changing the access rights* of an entity can be realized by first revoking its identity and then providing it with a new certificate that reflects the updated rights.

Separation of duties. Our setup supports *distinguishing administrative from technical duties* by providing distinct services for identity and rights management on the one side, and key storage and ABE key generation on the other side. To *separate responsibilities between organizations*, each organization implements its own identity management and operates its own service for key storage. Nevertheless, an organization could operate the key management service at a trusted third party or in the cloud.

The CA and the ABE key generation service are operated by a trusted global provider. To split responsibilities between organizations, we propose a hierarchical PKI with an inter-organizational master CA and additional organization-specific CAs. ABE key generation with access to the master secret key msk_{ABE} is a single point of attack. To mitigate this risk, we propose to apply techniques for distributing msk_{ABE} as in [6].

6 Proofs of Concept—ABE-OS-KEM and Infrastructure

In this section we present proofs of concept for our definition of ABE-OS-KEM and our cloud architecture. Particularly, we provide a description of an ABE-OS-KEM and complement it a description of how to implement our cloud architecture using cloud technology.

6.1 Our Construction of RCCA Secure ABE-OS-KEM

Our ABE-OS-KEM applies the ideas of outsourced decryption from [7] to the Rouselakis/Waters ciphertext-policy ABE scheme [11]. As in [7], we achieve security against replayable chosen-ciphertext attacks (RCCA, [7]) via the Fujisaki-Okamoto transform [5]. Our architecture from Sect. 4.1, and especially the separation between the session and FileCrypt services relies on the properties of a key encapsulation mechanism. Therefore, we have modified the original encryption scheme [7] to adhere to the definition of a key encapsulation mechanism. Furthermore, we explicitly describe our scheme in the efficient type-III setting of bilinear groups [3].

For convenience, we assume the data encapsulation mechanism used in conjunction with our ABE-OS-KEM to use keys from $\{0, 1\}^A$. The scheme is defined as follows:

Setup(1^λ): compute RW master secret $msk_{\text{ABE}}^{\text{RW}} = \alpha$ and public parameters $pub_{\text{ABE}}^{\text{RW}} = ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \{g_i, u_i, h_i, v_i, w_i\}_{i \in \{1,2\}}, e(g_1, g_2)^\alpha)$, where p is a λ -bit prime, $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order p , $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerate bilinear map and generator $g_1 \in \mathbb{G}_1$, generator $g_2 \in \mathbb{G}_2$ and $\alpha \in \mathbb{Z}_p$ are chosen uniformly at random. Parameters $a, b, c, d \in \mathbb{Z}_p$ are chosen uniformly at random and are used to compute $\forall i \in \{1, 2\} : u_i = g_i^a, h_i = g_i^b, v_i = g_i^c, w_i = g_i^d$. Choose hash functions $F : \{0, 1\}^* \rightarrow \mathbb{Z}_p, H_1 : \mathbb{G}_T \times \{0, 1\}^A \rightarrow \mathbb{Z}_p$ and $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^A$. Output $pub_{\text{ABE}}^{\text{RW}} = (pub_{\text{ABE}}^{\text{RW}}, F, H_1, H_2)$ and $msk_{\text{ABE}}^{\text{RW}} = msk_{\text{ABE}}^{\text{RW}}$.

Keygen($pub_{\text{ABE}}^{\text{RW}}, msk_{\text{ABE}}^{\text{RW}}, A_{ID}$): compute RW secret key for attribute set A_{ID} , i.e. pick $r_{ID}, r_{a_1}, \dots, r_{a_{|A_{ID}|}} \xleftarrow{\$} \mathbb{Z}_p$. Let $K_0 = g_1^\alpha w_1^{r_{ID}}, K_1 = g_1^{r_{ID}}$, and for all $a_i \in A_{ID} : K_{a_i,2} = g_1^{r_{a_i}}, K_{a_i,3} = (u_1^{F(a_i)} h_1)^{r_{a_i}} v_1^{-r_{ID}}$. Output $sk_{\text{ABE},ID} = (K_0, K_1, \{K_{a_i,2}, K_{a_i,3}\}_{a_i \in A_{ID}})$.

Encaps($pub_{\text{ABE}}^{\text{RW}}, \mathbb{A}$): parse \mathbb{A} as (M, ρ) with $M \in \mathbb{Z}_p^{\ell \times n}$ and row labelling $\rho : [\ell] \rightarrow \{0, 1\}^*$. Pick $R \xleftarrow{\$} \mathbb{G}_T, k \xleftarrow{\$} \{0, 1\}^A$. Set $s := H_1(R, k)$ and $r := H_2(R)$. Let $C' = k \oplus r$. Pick $y_2, \dots, y_n, t_1, \dots, t_\ell \xleftarrow{\$} \mathbb{Z}_p$. Set $\lambda := M \cdot (s, y_2, \dots, y_n)^\top$; denote by λ_i the i^{th} component of λ . Let $C := R \cdot e(g_1, g_2)^{\alpha s}, C_0 := g_2^s$ and for all $i \in [\ell] : C_{i,1} := w_2^{\lambda_i} v_2^{t_i}, C_{i,2} := (u_2^{F(\rho(i))} h_2)^{-t_i}, C_{i,3} := g_2^{t_i}$. Define $CT_{\text{ABE}} := ((M, \rho), C, C', C_0, \{C_{i,1}, C_{i,2}, C_{i,3}\}_{i \in [\ell]})$. Output (k, CT_{ABE}) .

Transkey($pub_{\text{ABE}}^{\text{RW}}, sk_{\text{ABE},ID}, ID$): pick $z \xleftarrow{\$} \mathbb{Z}_p$ and set $sk_{\text{PK},ID} := z$ and $tk_{ID} := (K'_0 = K_0^{1/z}, K'_1 = K_1^{1/z}, \{(K'_{a_i,2} = K_{a_i,2}^{1/z}, K'_{a_i,3} = K_{a_i,3}^{1/z})\})$. Output $(sk_{\text{PK},ID}, tk_{ID})$.

Transform($pub_{\text{ABE}}, tk_{ID}, CT_{\text{ABE}}$): if the ciphertext is malformed or tk_{ID} does not satisfy $\mathbb{A} = (M, \rho)$, output \perp and exit. Otherwise, let $I \subseteq [\ell]$ be a satisfying set of \mathbb{A} with respect to tk_{ID} , i.e. there are $b_i \in \mathbb{Z}_p$ such that $\sum_{i \in I} b_i M_i = (1, 0, \dots, 0)$, where M_i denotes the i^{th} row of M . Compute $B' := \prod_{i \in I} \left(e(K'_{1,i}) e(K'_{\rho(i),2}, C_{i,2}) e(K'_{\rho(i),3}, C_{i,3}) \right)^{b_i}$ and $B := e(K'_0, C_0) / B'$. Output $CT_{\text{PK}} = (C, C', B)$.

Decaps($pub_{\text{ABE}}, sk_{\text{PK}}, ID, CT_{\text{PK}}$): parse $CT_{\text{PK}} = (T_0, T_1, T_2)$ and compute $R := T_0 / T_2^z$, $k := T_1 \oplus H_2(R)$ and $s := H_2(R, k)$. Check whether $T_0 = R \cdot e(g_1, g_2)^{\alpha s}$. If the check fails, output \perp , otherwise output k .

The scheme's correctness and selective RCCA security follow from the respective properties of RW ciphertext-policy ABE [11] and the Fujisaki-Okamoto transformation [5], after applying the obvious modifications required due to outsourced decryption.

The major performance advantage of an ABE-OS-KEM in our architecture results from splitting the Decaps algorithm of an ABE-KEM into two separate Transform and Decaps steps (cf. Definitions 1 and 2). The complexity of Decaps in the ABE-OS-KEM is now independent of $sk_{\text{ABE}, ID}$ and CT_{ABE} , and hence independent of the user's permissions and of the ciphertext's policy. This allows us to handle users with a large set of permissions and complex access policies in our cloud scenario that involves resource constrained HSMs.

In concrete instantiations, arithmetic in \mathbb{G}_1 is much more efficient than in \mathbb{G}_2 [3]. In our setting, the resource constrained HSM performs TransKey, while the TC server executes Encaps. Our ABE-OS-KEM accounts for this by placing user secrets $K_{i,j}$ as arguments of TransKey in group \mathbb{G}_1 and ciphertext components of $C_{i,j}$ as arguments of Encaps in \mathbb{G}_2 .

6.2 Implementation

As a proof of concept, we have implemented our service architecture from Sect. 4 based on the following technologies:

Docker: We have implemented the sub-services of our ACE service as separate Docker⁴-based micro services. This supports elasticity and multi-tenancy because we can create and destroy service instances for each user, based on the current load situation.

Kubernetes: We use Kubernetes⁵ for orchestrating, scheduling, and monitoring the ACE sub-services. We operate a Kubernetes cluster that consists of four nodes. Each node runs with an Intel Xeon E3 at 2.3 GHz and 8 GB RAM.

RabbitMQ: We have implemented a queue between the FileCrypt service and the session service based on RabbitMQ.⁶ Then, our queuing mechanisms allows us to dynamically assign TC-HSMs to active users.

⁴ <https://docker.com>.

⁵ <https://kubernetes.io>.

⁶ <https://www.rabbitmq.com>.

Amazon S3: The FileCrypt service is compatible with the Amazon S3⁷ interface. Hence, we can use a commercial cloud storage provider to host the encrypted files.

WebDAV: We have implemented a WebDAV service as the user front-end to ACE, with the session service serving as its back-end. Our implementation extends the Go⁸ WebDAV implementation to support ABE policies. The extension passes policies as so-called WebDAV dead properties to the FileCrypt service. Since standard WebDAV clients do not support this mechanism, we have implemented a graphical user interface that allows us to define policies for file upload (encryption).

Our implementation shows that it is practical to implement an ABE service with modern cloud technology.

7 Future Work

As part of future research, we will enhance our design by various services. In particular, we want to realize a service for searchable encryption as introduced by Song et al. [13], granting authorized users the ability to efficiently search encrypted data. Another line of future research aims at including the multi-authority feature of Chase [2] into our cloud. The multi-authority feature allows the cloud resources of to be pooled among multiple instantiations of the cloud, while keeping everything beyond the hardware separate. This can help with removing the single trusted global provider for key generation that we assume in Sect. 5.1.

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
2. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_28
3. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings—the role of Ψ revisited. *Discret. Appl. Math.* **159**(13), 1311–1322 (2011)
4. Cloud Security Alliance: SecaaS implementation guidance category 8: Encryption (2012). https://downloads.cloudsecurityalliance.org/initiatives/secaas/SecaaS_Cat_8_Encryption_Implementation_Guidance.pdf. Accessed 06 July 2017
5. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. *J. Cryptol.* **26**(1), 80–101 (2013)
6. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.* **20**(1), 51–83 (2007)
7. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: 20th USENIX Security Symposium. USENIX Association (2011)

⁷ <https://aws.amazon.com/s3>.

⁸ <https://golang.org/>.

8. Katz, J., Lindell, Y.: Introduction to Modern Cryptography, 2nd edn. Chapman and Hall/CRC Press, London/Boca Raton (2015)
9. Mell, P., Grance, T.: The NIST definition of cloud computing (2011). <https://doi.org/10.6028/NIST.SP.800-145>. Accessed 06 July 2017
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1996)
11. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption. In: CCS 2013, pp. 463–474. ACM (2013)
12. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27
13. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE (2000)
14. Yang, K., Jia, X., Ren, K.: Attribute-based fine-grained access control with efficient revocation in cloud storage systems. In: ASIA CCS 2013, pp. 523–528. ACM (2013)
15. Yang, Y., Liu, J.K., Liang, K., Choo, K.-K.R., Zhou, J.: Extended proxy-assisted approach: achieving revocable fine-grained encryption of cloud data. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9327, pp. 146–166. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24177-7_8
16. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: INFOCOM 2010, pp. 534–542. IEEE (2010)
17. Zhang, P., Chen, Z., Liu, J.K., Liang, K., Liu, H.: An efficient access control scheme with outsourcing capability and attribute update for fog computing. *Future Gener. Comput. Syst.* **78**(2), 753–762 (2018)
18. Zhao, F., Nishide, T., Sakurai, K.: Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In: Bao, F., Weng, J. (eds.) ISPEC 2011. LNCS, vol. 6672, pp. 83–97. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21031-0_7



Relationship-Based Access Control for Resharing in Decentralized Online Social Networks

Richard Gay¹(✉), Jinwei Hu¹, Heiko Mantel¹, and Sogol Mazaheri²

¹ Modeling and Analysis of Information Systems, Department of Computer Science,
TU Darmstadt, Darmstadt, Germany

{gay,hu,mantel}@mais.informatik.tu-darmstadt.de

² Cryptography and Complexity Theory, Department of Computer Science,
TU Darmstadt, Darmstadt, Germany
sogol.mazaheri@cryptocomplexity.de

Abstract. Decentralized online social networks (DOSNs) have adopted quite coarse-grained policies for sharing messages with friends of friends (i.e., resharing). They either forbid it completely or allow resharing of messages only without any possibility to constrain their subsequent distribution. In this article, we present a novel enforcement mechanism for securing resharing in DOSNs by relationship-based access control and user-determined privacy policies. Our mechanism supports resharing and offers users control over their messages after resharing. Moreover, it addresses the fact that DOSNs are run by multiple providers and honors users' choices of which providers they trust. We clarify how our mechanism can be effectively implemented by a prototype for the DOSN Diaspora*. Our experimental evaluation shows that controlling privacy with our prototype causes only a rather small performance overhead.

Keywords: Decentralized online social networks · Privacy
Access control

1 Introduction

Online social networks (OSNs) are web-based services that offer users the functionality to share messages with other users. A decentralized online social network (DOSN) [12] is an OSN that is supported by multiple service providers. In a DOSN, a user can choose a provider whom she trusts most to store her profile. Typical OSNs provide an author with means for sharing a message with the set of users she categorized as 'friends', 'colleagues', etc. As of today, DOSNs allow authors to share sensitive messages with selectable sets of users but forbid resharing of sensitive messages entirely.¹

¹ Even the centralized OSN Facebook supports controlled resharing only with users with whom the message had been already shared with. The alternative in Facebook is uncontrolled sharing where users may arbitrarily reshare messages that they receive.

A better support of controlled resharing in DOSNs would be beneficial. Consider, for instance, a user who visits various US national parks. Before the trip, she had informed her friends early that she will visit the US and later informed them about the exact route and the dates of her visits. During the trip, she enjoys the landscape and sharing pictures with her friends. For privacy reasons, she wants to control spreading of this information: pictures should remain among her direct friends and dates of her visit among her friends and her friends' closest friends. Her motivation for limiting spreading of the dates of her trip could be to not provoke burglary [25]. She is less concerned about distributing the mere fact that she is visiting US national parks. This information may be distributed further, but without becoming public. This scenario illustrates the need for providing fine-grained control over sharing, resharing, and the distribution of reshared messages. We refer to the combination of these three forms of controlled information dissemination by the term *controlled resharing*.

In this article, we propose a privacy enforcement mechanism for controlled resharing in DOSNs. Our mechanism enables users to specify by privacy policies to which extent messages that they are sharing with others may be distributed further. Our mechanism provides control over the dissemination of messages inside a DOSN, ensuring that the privacy policies of all users are respected.

Conceptually, our enforcement of privacy policies is based on *relationship-based access control (ReBAC)* [13,15]. When checking authorization, we take the relationships of all users into account who were involved in delivering a message to the user who wishes to distribute this message further. Technically, we capture the relationship between users by trust values, where each user can define her personal trust values for categories of users. The decision whether a received message may be distributed to some category of users is made based on the concept of *trust concatenation* [19].

We developed an implementation of our ReBAC mechanism² for Diaspora* [18], at the time of writing the most popular DOSN [28]. To accommodate the distributed nature of DOSNs, our mechanism also has a distributed architecture. We chose a design that supports making authorization decisions in a decentralized fashion to avoid a single point of failure and performance bottlenecks. As underlying technological platform, we chose the CliSeAu tool [16]. This combination of design decisions results in a solution for enforcing controlled resharing in Diaspora* that is both, effective and efficient. In our performance evaluation, we observed an overhead of less than 2% for resharing in the domain of the same provider and of less than 4% when controlling resharing across providers.

2 Definition of Privacy Policies

After the author of a message m has shared m with a collection of categories (such as ‘friends’, ‘family’, or ‘colleagues’), some users in these categories might reshare m with others, some recipients of such a reshared message might reshare m again, and so on. To capture how a message has been delivered from its author

² Available at <http://www.mais.informatik.tu-darmstadt.de/CRDiC.html>.

u_1 to a set C_n of categories, we use lists of the form $(u_1, C_1, \dots, u_n, C_n)$, which we call *reshare paths*.

For instance, the reshare path $(Alice, \{Colleagues, Friends\}, Bob, \{Family\})$ captures the sharing of a message by Alice with her friends and colleagues and the subsequent resharing by recipient Bob with his family.

When resharing a message, the author risks that recipients might abuse sensitive content. Hence, the distribution of sensitive messages needs to be limited to receivers whom the author sufficiently trusts. The role of trust in making the decision to share or not to share a message is well captured by the notion of *decision trust*, “the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible” [22].

We model the trust of a user u in her category c by a scalar *trust value* from the interval $[0, 1]$, where greater values mean greater trust. The maximal trust value 1 means that u trusts users in c as much as herself wrt. the propagation of her messages. The minimal trust value 0 means that u does not have any trust in users in c wrt. propagation. We capture the trust values of a user, along with the user’s categories and relationships to other users in the user’s privacy policy:

Definition 1. A privacy policy of a user u is a triple $pp_u = (CAT_u, rel_u, tv_u)$, where CAT_u is a set, $rel_u \subseteq CAT_u \times USER$ is a binary relation and $tv_u : CAT_u \rightarrow [0, 1]$ is a function. A privacy policy for a set of users $U \subseteq USER$ is a family $(pp_u)_{u \in U}$ of privacy policies for each user in U .

In a privacy policy, CAT_u specifies all categories of u . The relation rel_u captures which other users are in the categories of user u . For instance, $rel_{Alice}(Friends, Bob)$ captures that Bob is a member of Alice’s *Friends* category. The function tv_u captures the trust of user u in her categories. We impose no further constraints on privacy policies. Hence, through $rel_{Alice}(Colleagues, Alice)$ and $tv_{Alice}(Colleagues) = 0.5$, Alice could specify a medium trust in herself as a colleague.

Intuitively, rel_u specifies which users from the universe $USER$ of all users may receive a message that u (re)shares with a particular category. That is, rel_u captures an expectation of u about the visibility of messages that she shares with her categories. The function tv_u captures a complementary aspect, namely to which extent u trusts users in her categories to propagate her sensitive messages.

Sensitivity of messages is not part of privacy profiles. We capture the sensitivity of messages for authors by values in $[0, 1]$ (greater values mean greater sensitivity).

We denote the trust of a message’s author in a recipient u' , who obtained a message m via a reshare path π , under a privacy policy $(pp_u)_{u \in U}$ for a set U comprising all users in π by $\mathcal{PT}((pp_u)_{u \in U}, \pi, u')$. We define $\mathcal{PT}((pp_u)_{u \in U}, \pi, u')$ recursively over the length of the reshare path:

$$\begin{aligned} \mathcal{PT}((pp_u)_{u \in U}, (u, C), u') &= \max(\{tv_u(c) \mid c \in C \wedge rel_u(c, u')\} \cup \{0\}) \\ \mathcal{PT}((pp_u)_{u \in U}, \pi.(u, C), u') &= \mathcal{PT}((pp_u)_{u \in U}, \pi, u) \cdot \mathcal{PT}((pp_u)_{u \in U}, (u, C), u') \end{aligned}$$

That is, if a user u' received a message m from m 's author u directly, then the trust of u in u' via (u, C) equals u 's maximal trust value for a category of u from C that u' is in. If there is no such category, then the trust of u in u' is 0. If u' received m via a longer path π then the trust of m 's author in u' via π is the product of the trust values for each (re)sharing of m by a user along π . We discuss the choice in Sect. 6. The product of trust values ensures that prolonged paths yield a decreased trust value.

We make the semantics of privacy policies and their impact on sharing and resharing of messages more precise by the following definitions. As a prerequisite for the definitions, we say that a reshare path (u, C) (resp. $\pi.(u, C)$) is a *reshare path to user u'* iff $rel_u(c, u')$ holds for some category $c \in C$.

Definition 2. *Sharing of a message m with sensitivity value $s \in [0, 1]$ by a user u with a set of categories C complies with a privacy policy $(pp_u)_{u \in U}$ if and only if $s \neq 1$ holds, $C \subseteq \text{CAT}_u$ holds, and (u, C) is a reshare path to u' for all users u' who receive m due to this sharing.*

Definition 3. *Let $sc \in [0, 1]$ be arbitrary. Resharing of a message m with sensitivity value $s \in [0, 1]$ which had been received via a reshare path π , by a user u with a set C of categories complies with a privacy policy $(pp_u)_{u \in U}$ if and only if $s \neq 1$ holds; $C \subseteq \text{CAT}_u$ holds; (u, C) is a reshare path to u' for all users u' who receive m due to this sharing; and $\mathcal{PT}((pp_u)_{u \in U}, \pi, u) \geq \frac{sc}{1-s}$.*

The inequality condition introduced in Definition 3 establishes a lower bound on the trust for resharing ($\frac{sc}{1-s}$) that the author of a message can raise through an increased sensitivity value. Note that Definition 3 is parametric in $sc \in [0, 1]$, where higher values of sc are more restrictive. We refer to this parameter as *sensitivity coefficient*. For instance, the sensitivity coefficient $sc = 0.35$ ensures that resharing is completely forbidden along reshare paths π with low trust ($\mathcal{PT} < 0.3$), is allowed for low-sensitivity messages ($s < 0.3$) along reshare paths π with medium trust ($0.5 \leq \mathcal{PT} \leq 0.6$), and is completely forbidden for messages with a sensitivity value above 0.65. Sensitivity coefficients have been used before in the semantics of privacy policies for controlling the direct sharing of messages between users. For instance, Kumari et al. [24] propose sensitivity coefficients for different kinds of operations. Definition 3 transfers this concept to resharing.

Our privacy policies augment what can typically be found in OSNs, namely categories of users, by trust and sensitivity. Based on these ingredients in privacy policies, we define when sharing and resharing are compliant. Since the presented semantics leaves underspecified how users' privacy policies are obtained for checking compliance, the semantics supports a decentralized storage of policies as well as dynamically changing policies.

3 The ReBAC Mechanism

We chose a distributed architecture for our mechanism to accommodate the distributed nature of DOSNs. Each service provider of a protected DOSN is

supervised by a separate controller. The controller at a provider ensures that all sharing/resharing actions of users whose profiles the provider stores comply with the privacy policies of all users, not only of users hosted at the provider.

In the design of our ReBAC mechanism, we assume that a user’s messages are only distributed to service providers who are controlled by our mechanism and who can be trusted to not circumvent our control. This could be achieved, e.g., by corresponding legal agreements between service providers: Service providers who did not sign the agreement are excluded from receiving certain messages by providers who signed the agreement.

Our mechanism provides no protection against communication outside the DOSN. For instance, a user with whom a message has been shared or reshared inside the DOSN might take this message and communicate it to others via email or might use the browser to copy the message text and paste it, possibly with modifications, into a new message in the DOSN. This risk cannot be completely mitigated by technical means. Users should take this aspect into account when specifying trust values and when admitting users to their categories. A user who is distrusted should not be given a sensitive message in the first place.

Finally, we assume that the implementation of sharing/resharing with categories in the DOSN is sound in the sense that when a user shares or reshares a message m with a category c then only users in c receive m .

3.1 Decentralized Control of Resharing

The purpose of our ReBAC mechanism is to ensure that privacy policies of all users are obeyed when sharing and resharing messages inside a DOSN. Since we assume the DOSN to soundly implement sharing and resharing of messages with categories, compliance of sharing is ensured by the DOSN (recall Definition 2). Our controllers therefore do not control the sharing of a message by a user. Compliance of resharing, however is only partially ensured by the DOSN, leaving one crucial condition to be ensured by our controllers (recall Definition 3). When a user u who obtained a message m via a reshare path π attempts to reshare m with a set C of categories, our controller therefore checks:

$$\boxed{\text{Does } \mathcal{PT}((pp_u)_{u \in U}, \pi, u) \geq \frac{sc}{1-s} \text{ hold?}} \quad (\text{C})$$

Intuitively, the check (C) ensures that the trust of m ’s author into u is sufficiently high to reshare m with other users. The check involves the privacy policy of (at least) all users in π , which are part of the users’ profiles and, hence, possibly stored at another service provider than u ’s. For enabling the controllers to perform the check, our controllers establish the availability of all relevant information by *coordination* as follows.

Sharing m of sensitivity s by u at provider sp with categories C :

When this action is performed, the controller at sp disseminates the following information to the controller of each service provider sp' with a recipient of m : the initial reshare path, the sensitivity s , and for each recipient u' the trust value $\mathcal{PT}(pp_u, (u, C), u')$. Figure 1 (left) shows the dissemination procedure

in pseudo-code. For the sharing, it is invoked with the empty path π and the value $pt = 1$ representing the trust value for the empty path. Each recipient stores the values for controlling future reshares.

Resharing m by user u at provider sp with categories C :

When this action is performed, the controller checks whether the action complies with users' privacy policies by performing check (C). If the check succeeds, then the controller disseminates the reshare path, sensitivity, and trust values to all affected service providers, as in the case of sharing. Otherwise, the controller disallows the resharing. Figure 1 (right) shows the procedure in pseudo-code. For the check, the controller uses its local privacy policies $(pp_u)_{sp(u)} = sp$ as well as the value pt_1 obtained when m was delivered to the controller's service provider. If m was never received from another service provider (i.e., π_1 is empty), then $pt_1 = 1$.

The coordination among controllers follows the propagation of messages. When a sharing/resharing causes a message to be delivered to another service provider, the information exchanged by the controllers enables the receiving controllers to perform check (C) for future reshares. No further communication among the controllers is then required for this check. That is, all coordination is decentralized.

<pre> disseminate(m, π, u, C, s, pt): Data: $sp, pp_u = (CAT_u, rel_u, tv_u)$ $U \leftarrow \{u' \mid \exists c \in C : rel_u(c, u')\}$ for $sp' \in \{sp(u') \mid u' \in U\} \setminus \{sp\}$ do $PT \leftarrow \emptyset$ for $u' \in \{u' \in U \mid sp(u') = sp'\}$ do $pt' \leftarrow pt \cdot \mathcal{PT}(pp_u, (u, C), u')$ $PT \leftarrow PT \cup \{(u', pt')\}$ end send($con@sp', (m, \pi.(u, C), s, PT)$) end </pre>	<pre> Data: $sp, u, m, C, (pp_u)_{sp(u)=sp}, sc$ $s \leftarrow$ sensitivity value for m $\pi \leftarrow$ reshare path for m to u decompose $\pi = \pi_1.\pi_2$, where π_2 is maximal with only users from sp $pt_1 \leftarrow$ trust value for π_1 to first user in π_2 $pt \leftarrow pt_1 \cdot \mathcal{PT}((pp_u)_{sp(u)=sp}, \pi_2, u')$ if $pt \geq \frac{sc}{1-s}$ then disseminate(m, π, u, C, s, pt) else disallow reshare </pre>
---	--

Fig. 1. Algorithms for decentralized coordination among controllers

3.2 Decentralized Control with Timely Policies

The approach presented in Sect. 3.1 has the virtue to require no coordination among controllers for checking whether a reshare complies with users' privacy policies. This virtue comes with a drawback, which we address in this section: When a controller checks whether a reshare of a message obtained from another service provider is compliant, it might rely on outdated privacy policies underlying the pt -value for π_1 in Fig. 1. That is, once a message has been shared/reshared with users at another provider, changes of privacy policies by users at the author's service provider cannot influence the further propagation of the message anymore.

We propose a decentralized approach for checking whether a reshare complies with timely privacy policies of users. The key idea behind our approach is to have the controllers perform a decentralized computation of the check (C) and to refrain from any proactive distribution of privacy profile information. For the decentralized computation, we transform the check (C) slightly to:

$$\boxed{\text{Does } (1 - s) \cdot \mathcal{PT}((pp_u)_{u \in U}, \pi, u) \geq sc \text{ hold?}} \quad (C')$$

The left-hand side of (C') contains all profile information (the privacy policies and the sensitivity value) and is the subject to the decentralized computation. Notably, in general none of s , $(pp_u)_{u \in U}$, and π is known to the controller performing the check. Suppose a user u wants to reshare a message m . The decentralized computation proceeds in four phases:

Phase 1: The controller at u 's service provider queries the user u' from which u received m . The controller then delegates check (C') to the controller at the service provider of u' . The latter controller queries the reshare path $\pi = (u_1, C_1, \dots, u_n, C_n)$ of m to $u' = u_n$.

Phase 2: The controller for u_n delegates the check to the controller for u_1 , along with π . The controller for u_1 queries the sensitivity value s for m and initializes result R of the decentralized computation by assigning $R \leftarrow 1 - s$.

Phase 3: The active controller (initially the controller for u_1) takes the longest prefix π_1 of π such that all users in π_1 have their profiles at the service provider of the controller. The controller then updates the result by assigning $R \leftarrow R \cdot \mathcal{PT}((pp_u)_{u \in U}, \pi_1, u_2)$, where U contains all users in π_1 and where u_2 is the first user in the remaining suffix π_2 of π . If π_2 is non-empty, the controller delegates the further computation, along with R for the result and π_2 for the reshare path, to the controller of u_2 , which then proceeds in Phase 3. If π_2 is empty, then Phase 4 is entered.

Phase 4: The active controller (for u_n) checks whether $R \geq sc$ holds. Depending on the result of the check, the controller sends the decision to allow or to disallow the reshare to the controller of u . The controller for u implements this decision and records π for future reshares.

In Phase 1, we exploit that the user u' who (re)shared message m is part of m . Moreover, the controller at u_n knows π for m , as established inductively over the length of the reshare path in Phase 4. In Phase 2, we exploit that the sensitivity value s for m is stored at the service provider of u_1 .

Figure 2 visualizes the four phases for the case that all users on the reshare path are at different service providers (SP_i), each of which having its individual controller ($\text{con}@SP_i$). The coordination is triggered by the controller at u 's service provider. The four phases then sequentially activate the remaining controllers in the ordering indicated by the arrows.

The computation we propose avoids to gather users' privacy policies at a central location. Only intermediate computation results are provided to the involved controllers and are subsequently discarded again. The computation leads the mechanism to effectively check (C) in a decentralized fashion based on timely

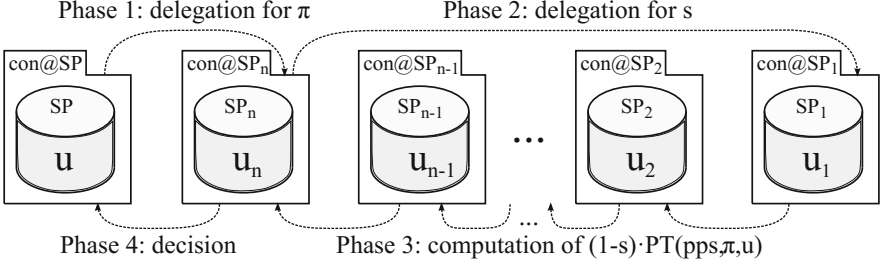


Fig. 2. Coordination for decentralized control with timely policies

privacy policies of users. The coordination among controllers in the computation follows the ordering of service providers in π , which ensures that successively involved providers are bound by contract.

3.3 Optimized Coordination

We propose two optimizations for reducing the amount of coordination in the decentralized computation presented in Sect. 3.2.

The first optimization applies when a service provider occurs more than once in a reshare path but other service providers occur in between. The optimization augments Phase 1, in which the controller for u_n additionally computes the set SP of all service providers in π and passes this set to the subsequent phases. Phase 3 is replaced by the following:

Phase 3': The controller computes $R' = R \cdot \prod_{u_i \in U} \mathcal{PT}(pp_{u_i}, (u_i, C_i), u_{i+1})$, where U is the set of all users whose profile is stored at the service provider of the controller and where $u_{n+1} = u$. The controller then removes itself from SP . If the resulting set is empty, then Phase 4 is entered. Otherwise, the controller delegates the further computation, along with R' for the result, the updated SP , and with the unchanged reshare path π , to some controller in SP who then proceeds in Phase 3'.

In Phase 3', each controller is activated at most once. The optimization is sound and precise due to the associativity and commutativity of multiplication. However, it does in general not preserve that service providers of successively involved controllers are bound by contract.

The second optimization particularly affects long reshare paths and reshare paths containing low trust values. It augments Phase 3 by the additional abort condition $\boxed{\text{Is } R' < sc?}$ that triggers the transition to Phase 4. With this condition, the computation terminates once a sufficiently low intermediate result R is encountered. The optimization is sound and precise because the product in the definition of function \mathcal{PT} is monotonically decreasing in further factors, as each of the factors equals a trust value $tv_u(c) \in [0, 1]$. Both optimizations can soundly and precisely be combined, and each maintains a decentralized computation.

4 A Prototype for Diaspora*

To demonstrate the feasibility of our ReBAC mechanism, we developed *CReDiC*, short for Controlled Resharing in Diaspora* with CliSeAu. CReDiC implements the mechanism for Diaspora*, the popular open-source DOSN. The implementation utilizes timely privacy policies (as described in Sect. 3.2) with optimized coordination (as described in Sect. 3.3). Diaspora* is a suitable candidate for CReDiC as its sharing and resharing with categories is sound. As the underlying technological platform of CReDiC, we utilize CliSeAu [16].

4.1 CliSeAu for Ruby

CliSeAu is a tool for dynamic policy enforcement in distributed programs [16]. Previously, it supported enforcement for Java programs only. We developed an extended variant of CliSeAu that supports enforcement for Ruby programs in addition. This was necessary for building CReDiC on top of CliSeAu, as Diaspora* is implemented in Ruby. Our extension utilizes Aquarium [30] for instrumenting Ruby programs. It consists of 244 lines of Java code and 38 lines of Ruby code.

Our extension of CliSeAu retains the high-level architecture and the coordination model used by CliSeAu for the mechanisms it generates. That is, the mechanisms consist of (ECs), placed at the individual components of the distributed target program. At runtime, each EC intercepts policy-relevant events of one component of the target, makes decisions for intercepted events, and enforces the decisions made. The developer of an enforcement mechanism using CliSeAu can specify the events to intercept, the decision-making, the enforcement, and the coordination among multiple ECs.

4.2 Mapping Diaspora* on Our Trust Model

We instantiate the trust model introduced in Sect. 2 for Diaspora* as follows. Diaspora* supports that users organize their acquaintances into categories (called “aspects” in Diaspora*) and that users can change the set of their categories from the default categories provided by Diaspora*. A user’s set of categories corresponds to the set CAT_u in our model. An acquaintance is either in a category of a user or not, which is captured by the relations rel_u in our model. When a user wants to share/reshare a message, she can select one or multiple of her categories to share with. This corresponds to how we model sharing and resharing with sets of categories.

Trust and sensitivity values are not supported by Diaspora*. We augment Diaspora* by trust values for categories by suffixing the category names with their trust value – e.g., “family (0.9)”. Our mechanism separates name and trust value again to allow users to change trust by renaming the category. We simulate the sensitivity value of a message by utilizing the least trust value among the categories with which the message is shared.

Diaspora* prohibits resharing of sensitive messages, i.e., of messages not classified as ‘public’. We enable our trust model for resharing by eliminating this constraint. Since Diaspora* does not allow users to specify categories for resharing and rather delivers a reshared message to all users who are related to the resharing user, we simulate the categories for a reshare by taking all categories of the resharing user. Technically, we implemented this as a patch to Diaspora* (version 0.5.3.1) consisting of 22 deleted and 20 inserted lines of code.

4.3 The Prototype

We implemented CReDiC as a policy for the CliSeAu tool. This policy specifies one EC for each service provider (called “pod” in Diaspora*). The ECs run at the respective providers and are responsible for controlling the reshares performed by users at that provider. Notably, the ECs establish the same decentralized architecture as the DOSN and neither introduce any centralized component nor impose requirements on users’ client software.

For controlling reshares, CReDiC specifies one method of Diaspora* to be intercepted. When a call to this method is intercepted by an EC at runtime, this EC extracts, from the arguments passed to the method, the user u who attempts to reshare a message as well as the message m to be reshared. With this information, the EC cooperates with other ECs of the DOSN as described in Sects. 3.2 and 3.3 for determining whether the attempted reshare complies with the users’ privacy policies.

CReDiC obtains trust values by monitoring changes of category names in profiles of users (recall that we encode the trust values in the names). It obtains sensitivity values by monitoring newly shared messages. For this monitoring, CReDiC intercepts four methods of the Diaspora* code. They allow CReDiC to keep track of dynamically changing privacy policies at the respective EC and take them into account for controlling resharing.

CReDiC is modular, consisting of eight individual components that we call “micro-policies”. Four micro-policies handle changes of users’ privacy policies and the storage of sensitivity values. The other four micro-policies handle the four phases for resharing. This separation yielded a low code complexity (each micro-policy is implemented in at most 41 lines of code). Figure 3 depicts the micro-policies (shaded boxes), their triggers (white boxes with solid arrows), and their temporal ordering (uncontinuous arrows). The trigger for a micro-policy is either an event of a service provider or a delegation received from another EC. The modularization allows a phase to take place at the same controller as the previous phase (dotted arrows) or at a different one (dashed arrows). The figure displays dash-dotted arrows where both cases are possible.

CReDiC globally fixes the sensitivity coefficient to $sc = 0.35$. Note that the particular coordination model of CliSeAu based on delegation allows CReDiC to control simultaneously occurring reshares in an interleaved fashion, without waiting for the completion of all four phases for each individual reshare. For securing the communication between the ECs, CReDiC utilizes CliSeAu’s SSL feature. With this feature, a certificate infrastructure is automatically generated for the

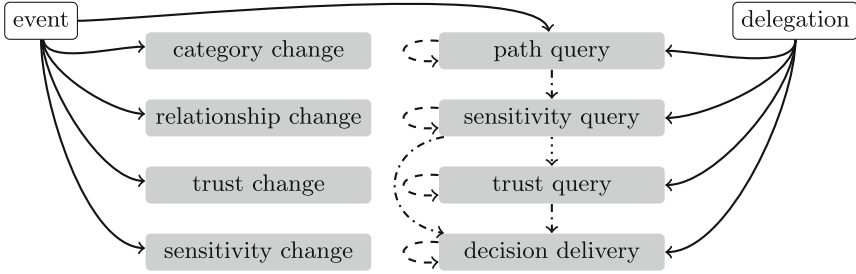


Fig. 3. Modular architecture of controllers based on eight micro-policies

individual ECs such that authenticity and confidentiality of the cooperation is maintained in presence of a network attacker. Overall, the implementation of CReDiC consists of 614 source lines of Java source code and additional 184 lines of Ruby code that realizes the interface to the Diaspora* code.

4.4 Deployment and Usage

As the provider of a Diaspora* pod, one can deploy CReDiC in two steps. First, one patches the code of the pod by applying the small patch described in Sect. 4.2. This step can be performed automatically with the GNU `patch` tool. Second, one instruments the code of the pod with CReDiC. This step is performed automatically by invoking the extended version of `CliSeAu`, described in Sect. 4.1, with CReDiC as a parameter.

Once CReDiC is deployed to a Diaspora* pod, controlled resharing is enabled in the pod. Users specify their privacy policies via the accustomed Diaspora* web interface. Concretely, a user specifies her set of categories and the users in these categories as she would in normal Diaspora*. She can specify and update her trust in categories by modifying the trust value contained in the category name (as described in Sect. 4.2). In particular, users need not use further interfaces to specify their privacy policies or benefit from CReDiC’s controlled resharing.

4.5 Analysis

We conducted several tests to verify the effectiveness of CReDiC. Concretely, we verified CReDiC for three policy-compliant cases of resharing: resharing a message from the pod on which it was initially shared, resharing from a different pod, and re-resharing a message involving three pods. We also verified for corresponding non-compliant cases that CReDiC successfully prohibits the resharing. That is, the tests confirmed that CReDiC effectively enforces users’ privacy policies. During the tests, the ECs of all pods involved in a reshare path were online during the resharing. Since DOSN pods typically aim to be available to their users, CReDiC does not implement a fallback strategy for offline pods.

Our ReBAC mechanism and our prototype scale as follows. The number of controllers (ECs) is independent in the number of users in the DOSN and

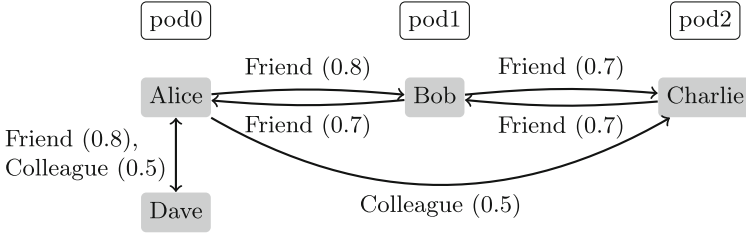


Fig. 4. Example scenario for the performance evaluation

grows linearly in the number of pods. The amount of network communication performed for the reshare of a message grows linearly in the number of pods in the reshare path of the message. This is due to our first optimization described in Sect. 3.3. In particular, no network communication takes place between the controllers when the reshare path involves only a single pod or when a user changes her privacy policy. The computational complexity grows linearly in the length of the reshare path: for each element of the path, one lookup of a trust value and one multiplication. Because our prototype duplicates users’ privacy policies in its own state and stores reshare paths, the memory required by each EC grows linearly in the number of pod users’ categories and in their number of shares and reshares.

5 Performance Evaluation

Being a mechanism that operates while the DOSN Diaspora* is running, CReDiC necessarily introduces some overhead. We evaluate how much overhead is caused with experiments in which we measure and compare the time taken by Diaspora* for resharing with and without CReDiC.

For the performance evaluation, we used three machines with Intel Quad-Core i5-4590 (3.3 GHz) CPUs and 32 GB RAM. The machines ran Ubuntu 14.04.2 with 3.13.0 kernel. We ran three patched Diaspora* pods (see Sect. 4.2) in production mode with Ruby 2.1.1, Apache 2.4.7, and a MySQL 5.5.54 database. Four user profiles were hosted by the three pods. We measured page fetch times using `curl` 7.52.1 on an Intel Quad-Core i7-6600U (2.6 GHz) with 16 GB RAM. All four machines were connected in a 1 Gbps LAN. We use a setup consisting of four users. Figure 4 displays the users (shaded boxes) and their association with the pods (white boxes). Users’ privacy policies are indicated by arrows: An arrow from user u to user u' labeled with category c and trust value t represents that $rel_u(c, u')$ holds and $tv_u(c) = t$. That is, users’ trust in categories is repeated on all arrows with the same source and same category label.

Table 1 shows our results. For each analyzed operation, the table contains a separate row. The first column shows the names of the operations, the second and third column show the durations of the operations in Diaspora* when CReDiC

Table 1. Performance evaluation results

<i>Operation</i>	<i>Duration</i>		<i>Overhead</i>	
	CReDiC	Diaspora*	Absolute	Relative
Share message	231.5 ms	230.3 ms	1.2 ms	0.52%
Reshare (intra)	294.6 ms	290.7 ms	3.9 ms	1.34%
Reshare (inter)	294.3 ms	281.8 ms	12.5 ms	4.44%
Change trust	36.0 ms	31.1 ms	4.9 ms	15.76%

is enabled and, respectively, disabled. The fourth and fifth column show the absolute and relative overhead.

Sharing a message took 231.5 ms with CReDiC enabled, compared to 230.3 ms with CReDiC disabled, which corresponds to an overhead of 1.2 ms (0.52%). For resharing, we evaluated two cases: intra-provider resharing, where Alice reshares a message by Dave with her friends and colleagues, and inter-provider resharing, where Bob reshares a message by Alice with his friends. For the two operations, the overhead of CReDiC ranges from 3.9 ms to 12.5 ms (1.34% to 4.44%). CReDiC’s overhead on dynamically changing trust between users was 4.9 ms (15.76%, due to the low baseline duration of 31.1 ms). Each duration value in Table 1 reflects the mean of the lower 90th% of 1000 measurements [27].

Overall, CReDiC maintains a rather small performance overhead. An at first sight counterintuitive result of the evaluation is that inter-provider resharing in Diaspora* is faster than intra-provider resharing. However, Diaspora* notifies remote users asynchronously about reshared messages while users at the same pod are notified synchronously. In our scenario, inter-provider resharing notifies both recipients (Alice and Charlie) asynchronously while intra-provider resharing includes one synchronous update (Dave).

6 Related Work

Underlying our model of trust, presented in Sect. 2, are two main design decisions. Firstly, we model trust as scalar values ranging from 0 to 1, which can be found also elsewhere in the literature [4, 17, 23]. Alternatives found in the literature are models of trust based on vectors of scalars (e.g., [21]). Through vectors of scalars, individual aspects of trust such as belief and disbelief in users [21] or ability, integrity, and benevolence of users [26] can be captured in a more fine-grained fashion. We build our trust model on scalar trust values rather than vectors to give users means for quantifying their relationships while taking into account that specifying trust vectors might be a burden users refrain to take.

Secondly, our model utilizes a particular notion of trust concatenation (multiplication) and selects a single path (the reshare path) for capturing trust of an author in a resharing user. Multiplication for concatenating scalar trust values has been proposed before [4, 23]. Alternative models for scalar trust values have been proposed as well. These models combine some form of multiplicative

concatenation of trust with the aggregation of trust along multiple paths, for instance via weighted sums of path trust [17] or maximal path trust [23]. Further models for trust concatenation are based on trust vectors (e.g., [19]). In defining compliance with users' privacy policies based on the reshare path and no further paths between users, we see two advantages: reduced complexity and context-dependence. By context-dependence we mean that we consider the trust along the list of users who have actually seen and reshared the message, rather than users' reputations. That is, our choice of trust value reflects the notion of decision trust, which by definition is associated with a situational context. Further validation of our model or comparison to other models, e.g., by means of user studies, are beyond the scope of this article.

In our scenario, authors of messages are the sole owners of their messages. Multiparty access control (e.g., [20]) is outside the scope of this article.

The desire to control sharing and resharing has led to the proposal of several centralized approaches. Fong et al. [6, 13, 14] propose a model of OSNs, a ReBAC model, and a language for expressing ReBAC policies. Relationships between users are modeled as binary relations on users. The policy language is a modal logic on the relations of the OSN that allows specifying constraints on resharing and subsequent distribution. The ReBAC mechanism for OSNs by Carminati et al. [8] enforces privacy policies of authors that can specify the maximum length of reshare paths, the minimal concatenated trust value, or relationship categories. The access control is shared between the requesting user, who provides a proof of being authorized to access the resource, and the resource provider, who checks the proof. Virtual Private Social Networks [3, 9] are social networks built on centralized OSNs like Facebook but achieve privacy of user information at the client-side via a browser extension. This line of work focuses on controlled sharing of messages, not controlled resharing. SCUTA [24] is a usage control mechanism for centralized control of sharing in OSNs. The mechanism controls users' client-side operations, such as viewing, saving, and printing content.

Mechanisms for DOSNs have also been proposed. Albertini et al. [1] propose an access control mechanism for cloud-based OSNs. The proposed mechanism supports resharing but introduces centralized components, KMS and RMS, for storing keys and access rules. While the mechanism utilizes encryption for users' keys and access rules transmitted to KMS and RMS, colluding KMS and RMS could reveal the plain data. Bahri et al. [2] propose a mechanism for a-posteriori access control in a DOSN, which also relies on a centralized component (called TReMa). Our mechanism, in contrast, features a fully decentralized architecture. Safebook [10, 11] and PeerSoN [5, 7] are DOSNs for protecting privacy of user data. Both DOSNs include a mechanism for controlled sharing of messages. Controlled resharing is beyond their scope. GEM [29] is a distributed goal evaluation algorithm for datalog-like policies. The goal in our trust model (compliance according to Definition 3) is of a simpler but quantitative nature that cannot be specified as a goal for GEM. D-FOAF [23] is a distributed identity management system on top of trust relationships between users in multiple OSNs. D-FOAF computes the trust between two users by gathering trust values of all paths

between requester and owner at one location. Our mechanism computes trust between two users based on a single path (the reshare path) and computes path trust in a distributed fashion to keep users' privacy policies decentralized.

7 Conclusion

We presented a novel enforcement mechanism that supports more fine-grained privacy policies for resharing of messages than popular OSNs like Facebook and DOSNs like Diaspora*. Our ReBAC mechanism enables controlled sharing and resharing of messages among users hosted at one service provider of a DOSN and also among users hosted at different providers. The mechanism enforces personal privacy policies of users inside a DOSN based on ReBAC. As usual for such access control mechanisms, malicious communication outside the DOSN is not prevented. We demonstrated that the mechanism can be effectively implemented by a prototype for Diaspora* and showed that its performance overhead is small.

Our mechanism complements mechanisms for controlled sharing in OSNs by which authors know and explicitly specify the supposed recipients of messages.

Enabling authors to better control how their messages spread after resharing shall allow them to permit resharing more often, without uncontrollable dangers to their privacy. Thus, users can securely increase their outreach in DOSNs like Diaspora* and develop new personal connections with users who have received their messages via trusted others.

Acknowledgments. We thank the anonymous reviewers for their comments and thank Sarah Ereth for her feedback at an early stage of this work. This work was partially funded by CASED (www.cased.de) and by the DFG (German research foundation) under the project FM-SecEng in the Computer Science Action Program (MA 3326/1-3).

References

1. Albertini, D.A., Carminati, B.: Relationship-based information sharing in cloud-based decentralized social networks. In: 4th Conference on Data and Application Security and Privacy, pp. 297–304 (2014)
2. Bahri, L., Carminati, B., Ferrari, E.: CARDS - Collaborative audit and report data sharing for a-posteriori access control in DOSNs. In: IEEE Conference on Collaboration and Internet Computing, pp. 36–45. IEEE Computer Society (2015)
3. Beato, F., Conti, M., Preneel, B., Vettore, D.: VirtualFriendship: hiding interactions on online social networks. In: Conference on Communications and Network Security, pp. 328–336 (2014)
4. Beth, T., Borcherdig, M., Klein, B.: Valuation of trust in open networks. In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875, pp. 1–18. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58618-0_53
5. Bodriagov, O., Kreitz, G., Buchegger, S.: Access control in decentralized online social networks: applying a policy-hiding cryptographic scheme and evaluating its performance. In: 2014 International Conference on Pervasive Computing and Communication Workshops, pp. 622–628 (2014)

6. Bruns, G., Fong, P.W.L., Siahaan, I., Huth, M.: Relationship-based access control: its expression and enforcement through hybrid logic. In: 2nd Conference on Data and Application Security and Privacy, pp. 117–124 (2012)
7. Buchegger, S., Schiöberg, D., Vu, L.-H., Datta, A.: PeerSoN: P2P social networking: early experiences and insights. In: 2nd EuroSys Workshop on Social Network Systems, pp. 46–52 (2009)
8. Carminati, B., Ferrari, E., Perego, A.: Enforcing access control in web-based social networks. *Trans. Inf. Syst. Secur.* **13**(1), 6:1–6:38 (2009)
9. Conti, M., Hasani, A., Crispo, B.: Virtual private social networks and a facebook implementation. *Trans. Web* **7**(3), 14:1–14:31 (2013)
10. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy-preserving online social network leveraging on real-life trust. *Commun. Mag.* **47**(12), 94–101 (2009)
11. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: feasibility of transitive cooperation for privacy on a decentralized social network. In: 10th International Symposium on a World of Wireless, Mobile and Multimedia Networks, pp. 1–6 (2009)
12. Datta, A., Buchegger, S., Vu, L.-H., Strufe, T., Rzdca, K.: Decentralized online social networks. In: Furht, B. (ed.) *Handbook of Social Network Technologies and Applications*, pp. 349–378. Springer, Boston (2010). https://doi.org/10.1007/978-1-4419-7142-5_17
13. Fong, P.W.L.: Relationship-based access control: protection model and policy language. In: 1st Conference on Data and Application Security and Privacy, pp. 191–202 (2011)
14. Fong, P.W.L., Anwar, M., Zhao, Z.: A privacy preservation model for facebook-style social network systems. In: Backes, M., Ning, P. (eds.) *ESORICS 2009*. LNCS, vol. 5789, pp. 303–320. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04444-1_19
15. Gates, C.E.: Access control requirements for web 2.0 security and privacy. In: *Workshop on Web 2.0 Security & Privacy* (2007)
16. Gay, R., Hu, J., Mantel, H.: CliSeAu: securing distributed Java programs by cooperative dynamic enforcement. In: Prakash, A., Shyamasundar, R. (eds.) *ICISS 2014*. LNCS, vol. 8880, pp. 378–398. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-13841-1_21
17. Golbeck, J.A.: *Computing and applying trust in web-based social networks*. Ph.D. thesis, University of Maryland (2005)
18. Grippi, D., Salzberg, M., Sofaer, R., Zhitomirskiy, I.: The Diaspora* Project, February 2016. <http://diasporafoundation.org/>
19. Hang, C., Wang, Y., Singh, M.P.: Operators for propagating trust and their evaluation in social networks. In: 8th International Joint Conference on Autonomous Agents and Multiagent Systems. vol. 2, pp. 1025–1032 (2009)
20. Hu, H., Ahn, G., Jorgensen, J.: Multiparty access control for online social networks: model and mechanisms. *IEEE Trans. Knowl. Data Eng.* **25**(7), 1614–1627 (2013)
21. Jøsang, A.: A subjective metric of authentication. In: Quisquater, J.-J., Deswarte, Y., Meadows, C., Gollmann, D. (eds.) *ESORICS 1998*. LNCS, vol. 1485, pp. 329–344. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055873>
22. Jøsang, A., Ismail, R., Boyd, C.: A survey of trust and reputation systems for online service provision. *Decis. Support Syst.* **43**(2), 618–644 (2007)
23. Kruk, S.R., Grzonkowski, S., Gzella, A., Woroniecki, T., Choi, H.-C.: D-FOAF: distributed identity management with access rights delegation. In: Mizoguchi, R., Shi, Z., Giunchiglia, F. (eds.) *ASWC 2006*. LNCS, vol. 4185, pp. 140–154. Springer, Heidelberg (2006). https://doi.org/10.1007/11836025_15

24. Kumari, P., Pretschner, A., Peschla, J., Kuhn, J.M.: Distributed data usage control for web applications: a social network implementation. In: 1st Conference on Data and Application Security and Privacy, pp. 85–96 (2011)
25. Mao, H., Shuai, X., Kapadia, A.: Loose tweets: an analysis of privacy leaks on Twitter. In: 10th Annual ACM Workshop on Privacy in the Electronic Society, pp. 1–12 (2011)
26. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An integrative model of organizational trust. *Acad. Manag. Rev.* **20**(3), 709–734 (1995)
27. Oaks, S.: *Java Performance - The Definitive Guide: Getting the Most Out of Your Code*. O'Reilly, Sebastopol (2014)
28. Paul, T., Famulari, A., Strufe, T.: A survey on decentralized online social networks. *Comput. Netw.* **75**, 437–452 (2014)
29. Trivellato, D., Zannone, N., Etalle, S.: GEM: a distributed goal evaluation algorithm for trust management. *Theory Pract. Logic Program.* **14**(3), 293–337 (2014)
30. Wampler, D.: Aquarium: AOP in Ruby. In: *Aspect Oriented Software Development* (2008)



Secure Protocol of ABAC Certificates Revocation and Delegation

Alexey Rabin^(✉) and Ehud Gudes

The Open University of Israel, Ra'anana, Israel
alxrabin@gmail.com

Abstract. This paper deals with the maintenance of PKI certificates for Attribute Based Access Control (ABAC). We show, that the current standard has several problems in different revocation and delegation processes. This may lead to a security hole allowing usage of ABAC certificates, when it was revoked or transferred. As a solution we suggest architecture changes, that allow to perform revocation and transfer checks in such cases, based on extensions of the validation process of the ABAC certificates. We also discuss some privacy and performance challenges that are raised as a result of our proposal.

1 Introduction

The authorization process is one of the most important issues of the access control challenge. The classical approach of authorization is based on a concept of identification. Identification is a process that defines uniquely the subject that asks for permission, to the asset that provides it. Usually, the latter isn't defined specifically for a subject, but is bound to groups. The model of defining those groups and managing the mapping of subjects and permissions to them is usually referred to as RBAC - Role Based Access Control. RBAC is a De-Facto standard of authorization, and has many important advantages. However, there are some limitations in this approach, that make it hard to implement in certain scenarios. The most important limitation is its binary approach - the subject can only belong to, or not belong to a role. In scenarios where it is desired to calculate permissions using some logic, this approach is hard to implement. Another limitation of RBAC, is that a decision to add a subject to a role has to be driven by the actual permissions of this role, and not intuitively bound to the subject itself.

Another authorization approach, that is more flexible than RBAC, is ABAC (Attribute Based Access Control). This approach, introduced in McCollum et al. (1990), suggests that the permission decisions will be taken by the asset based on the attributes' values of the subject. Unlike roles, attributes do not grant permissions directly, but try to describe the subject itself. The permission decision is based on two independent processes. The first is a description of a user by an Attribute Authority (AA) using several attributes. The second is the calculation

that the asset manager performs, based on those attributes, which results with an access decision.

Figure 1 illustrates the basic authorization process of ABAC.

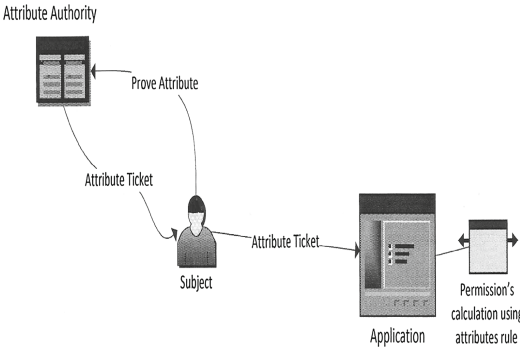


Fig. 1. Scheme of ABAC authorization

An important advantage of ABAC, which is critical to distributed environments, is that there can be a number of different AAs providing attributes. An asset can make decisions using reputation based calculations, such as introduced in Xiong and Liu (2004). For example, an asset *Bank* wants to calculate a credit rate of a subject *User*, who has asked for a loan. *User* will be asked to provide attributes of $A_{MaritalStatus}$, of $A_{ChildrenNumber}$, and of $A_{AverageAccountSum}$ with value of daily average sum of money in the account belonging to the user. The *Bank* can then assign importance of 5 to marital status, then to add to the credit rate, a 1 for each child but not to more than for 3 children, and to define thresholds for the average sum from -4 to 5. Final rank will be calculated by the sum of the values. Such attributes based calculations are an essential property of ABAC.

Modern ABAC environments also have the ability to provide the users with delegation and transfer of attributes. Li et al. (2003) defines two delegation types, which are important to a full usage of decentralized environment. The first is the delegation of an attribute authority, i.e. trust of one entity on a judgment of another. An example of such a delegation is a situation where medical qualification of a doctor is proved to the patient by Ministry of Health of one country, based on his certification done in Ministry of Health of another. The second type of delegation, which extends the previous one, is attribute based delegation. This means that some AA will trust the judgment of another AA, if the latter has some attribute. An example of such a delegation is a situation where medical qualification of a doctor is proved to the patient by Ministry of Health, based on his certification done in any organization that has an attribute of Medical School. Unlike the previous case, the MOH doesn't necessarily trust the schools specifically, but trusts their certification.

An important enabler of the ABAC approach, that actually allows its decentralization, is the concept of Attribute Certificates (AC). The concept, firstly

A subject asks the AA to prove some of its attributes. The AA returns a ticket that proves them. Now the subject can use this ticket to prove the attribute to the asset. When the asset gets the ticket's attributes, it uses the values as an input to an internal set of rules, which enable it to decide whether the asked permission should be allowed.

suggested in McCollum et al. (1990) and standardized in Housley et al. (1999), combines ABAC with PKI. In classical PKI, we use digital signatures of CA (Certificate Authority) on a certificate, as an identity proof of the certificate’s holder. In ABAC PKI, we use the digital signature of the AA on a certificate as a proof that the certificate holder has some attribute with a certain value. In addition to that, Linn and Nystrom (1999) suggested the use of anonymous certificates, i.e. ACs providing only the attributes themselves. This, in combination with a flexible protocol suggested in Blaze et al. (1998), allows ABAC to be useful in scenarios where privacy is very important.

In the field of access control, permissions revocation is as important as permissions grant. ACs provide a simple ability to treat revocation process with tools of PKI, and those tools were adopted by the AC’s standard Farrel et al. (2010). One of the differences of the latter standard from the previous ones is that it recommends not to use AC chains. Unfortunately, following this recommendation will not allow delegation process. On the other hand, as we will show in Sect. 3, when AC chains are in use, revocation mechanisms of AC standards do not cover all cases in which the certificates shall be disabled. This includes the case of cascading revocations while inference property was used and in some cases of attribute transfer. Our paper will suggest ways to widen the revocation model of ACs, so that revocation will effectively work also in those cases.

One of the strengths of decentralized ABAC approach is its ability to provide good level of privacy. The AC contains only the information the asset needs to make authorization decisions, so minimal personal data is exposed to the asset. This data isn’t sent to the attributes’ provider when it is used by the asset, so the usage is not exposed to the data provider. When we suggest to make protocol changes in the AC standard it is important to us not to weaken this ABAC property. In Sect. 5 we explore the issue and suggest different solutions to this concern.

The main contribution of this paper is a solution proposal for the risks of inconsistency of cascade revocation and cascade delegation of ACs, which can be an enabler to secure usage of AC chains. In Sect. 2 we present related work and discuss the current standards of AC. In Sect. 3 we discuss the security inconsistencies of the revocation processes and suggest the infrastructure and protocol changes to treat them. In Sect. 4 we present the needed algorithms’ changes to support the model. In Sect. 5 we present some challenges of our proposal and suggest approaches to solve them. We conclude in Sect. 6.

2 Related Work

Linn and Nystrom (1999) provides the basis of modern design for Attribute Certificates. The main issue was the creation of certificates that prove some attribute and not only user identity, so the authorization decisions could be based on them. They also proposed a basic inheritance and revocation scheme, based on the usage of classical PKI inheritance and revocation. That work also suggested the use of anonymous certificates, i.e. a certificate that does not contain identity

details, in order to improve the privacy of the holder. This approach was explored further and created a framework such as U-prove (Paquin and Zaverucha 2011).

Farrel et al. (2010) is the current standard for use of attribute certificates and it summarizes the work done in that field until 2010. It also deals with revocation and delegation, and suggests the usage of classical PKI approach to them including the use of CRLs (Certificate Revocation Lists). Due to the fact that the administration of AC chains is complex, the standard recommends not to use them. In our work we show one consequence of that complexity - the current standard does not cover some possibilities of malicious use of ACs after revocation or transfer. We also suggest a solution to that security hole.

A totally different approach to revocation, introduced by Rivest (1998), suggested avoiding at all the usage of CRL. This is achieved by making the certificate life period short enough. The idea wasn't accepted in the classical PKI, but it was adapted for ACs in Thompson et al. (2003). In our work we don't rely on that technique, since there are practical scenarios when long living certificates are needed.

Ye et al. (2006) propose a model for delegation, based on attributes, which is also important to our work. The model includes an attribute allowing to perform delegation of certain role and an attribute allowing to receive delegation of certain role. Though the work deals with the RBAC model, the approach allows decentralized delegation, based on the mentioned attributes, so the central manager, in advance, determines delegation rules and any participant can perform or receive delegation according to them. In ABAC this is a central idea.

Crampton and Khambhammettu (2008) defines several types of delegation, including strong transfer where the delegator loses all its rights after the transfer. We will show how strong transfer is incorporated in our model in the next section.

Earlier we mentioned that Linn and Nystrom (1999) proposed the usage of anonymous certificates. Those certificates have the benefit of privacy but create challenges in the revocation process, since the asset cannot connect a misbehavior to a certain user. The problem of creation of a protocol, that suggests correct balance between revocation efficiency and user privacy, is also a challenge in other fields of security research. One solution was suggested in Lou and Ren (2009), uses the idea of a Trusted Third Party (TTP), a proxy participant that masks the usage from different participants of the authorization negotiation process. Another approach is presented in Win et al. (2012). This work, that comes from the field of Digital Rights Management, proposes a scheme for revocation requests of assets, based on anonymous certificates. We will suggest to use both approaches as solutions to privacy concerns, later in our work.

Revocation check demands performance costs from the subject, and can even lead to Denial of Service attacks (Hinarejos et al. 2010). Improvements in efficiency of that check was a subject to wide research. In our work we will suggest to adopt ideas of CRL efficient structure of Naor and Nissim (2000), of PREON algorithm of Hinarejos et al. (2010) and of CREV-1 algorithm of Yap (2011) to make performance improvements of that process.

There is also a non-CRL revocation approach proposed by Boneh et al. (2001), which is based on key compromise solution of Rivest (1998). This approach allows to minimize risks of DoS attacks and improve revocation performance. We plan to incorporate some of these ideas in future work.

3 Revocation Issues

Revocation is one of the most important issues in the PKI model. It deals with situations where the issuer decides that the certificate shall become not valid before it reaches its validity date. That can happen when new information is received about the subject or when his private key is lost or stolen.

In distributed environment the issuer has no way to know where the certificate could be used. That is why all PKI standards contain a way in which the asset can check certificate's revocation status. Usually, the check is based on CRL - a list of all revoked certificates, that is published by the issuer, and can be downloaded or checked on-line during login process. As we will show in this section, in case of ACs there are revocation situations where the check cannot be done. Therefore, there is a need to extend the classical CRL protocol to treat ABAC properly.

3.1 Inference Cascade Revocation

Inference means the ability of one AA to sign an AC for a subject, based on ACs it got from other AAs. The former AC is referred to as *result certificate*. The latter as a *reason certificate*. The value of a specific field of the result AC is calculated from values of the reason ACs. The process can be repeated. The chain of ACs can be described as a reversed tree with current AC as a root, vertexes as reason ACs, and arrows describing calculations.

In Fig. 2, we see an example of such a tree. Att_p stands for some attribute. Att_1 and Att_2 are two attributes who's values' average is a value of Att_p . Att_1 value is calculated as sum of Att_{11} and Att_{12} values. And Att_{12} value is calculated from Att_{121} value by some rule.

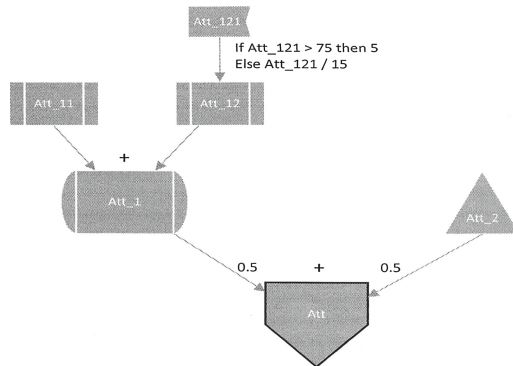


Fig. 2. Tree of attribute value calculation.

Revocation is a challenge when inference is used. In case one of the reason certificates is revoked and this certificate had proved some attribute that is critical to the issuance of the result AC, the result certificate shall also be revoked. And if there is a path in an inference tree of critical reason certificates, we expect that revocation of AC in the deepest level will lead to revocation of all the path. Unfortunately, classical CRL protocols allow assets to check issuance chain but not inference chain. AC format, described in Farrel et al. (2010) has no fields that show the inference connection. This is why the authority which issued the result certificate doesn't necessarily know about the revocation of the reason one, and therefore the derived certificates and all its descendants remain valid.

Problem Example: An attribute authority AA_{sch} gives certificates allowing work in schools to anyone who has B.Ed degree, proved by authority AA_{ed} , and has no legal violations, proved by authority AA_{police} . Some person can get the needed A_{ed} and A_{police} certificates, provide them to AA_{sch} , and get certificate A . According to current attribute certificate's standard, this certificate has no clear linkage to A_{ed} and A_{police} . It will be provided solely to assets (schools) as a permission to work. In case the holder is convicted in some criminal activity, his A_{police} certificate will be revoked. But since A has no link to its ancestor, it will remain valid.

One solution to the challenge, which follows the recommendation of Farrel et al. (2010) not to use AC chains, is to put the revocation responsibility in such case on the shoulders of AA. The AA can monitor revocation of each reason certificates it ever used for its valid issuances, and when finding a revocation, to revoke the result certificates. But because of the overhead it puts on the AA and low effectiveness of the solution, we will suggest a different approach.

Our proposal to the issue is to add an extension to the current Attribute Certificate standard and create an infrastructure to support it. This extension is based on an idea that during the process of AC creation, an AA should define which certificates were critical to the AC construction, and consequently, their revocation shall lead to AC revocation. In our paper they will be referred to as *Mandatory Certificates - MC*. And since dependency decisions are transitive, all MCs of any Mandatory Certificate of a reason AC, shall also become MCs of the AC itself, and so on, recursively.

The certificate extension will contain a list of certificates' details. For each certificate we will store its ID and link to CRL. Those fields allow an asset to check the revocation of each Mandatory Certificate of the AC. Since the process is done against each CRL server, the Certificate ID will be unique. Consequently, the scenario of a user using a certificate that is based on the revoked one, will be prevented. We will not save details of the certificate itself but only the required meta-data, in order to keep better level of privacy.

3.2 Transfer Delegation Revocation

Delegation is one of the most important properties of ABAC, since it allows decentralization of access control. Linn and Nystrom (1999) proposed a PKI

implementation to the delegation process. In their suggestion, a delegator issues a certificate to a delegatee. This certificate can contain partial or full attributes the delegator has. When the delegatee uses this certificate to get authorization from an asset, he should preserve not only his certificate, but also all the delegation chain starting from some trusted AA. The PKI delegation is combined in Farrel et al. (2010) with the idea of delegation attributes of Ye et al. (2006). When the delegation process is allowed, the AA adds to the AC an attribute that indicates that. It can also add a set of rules for delegation (for instance - maximal number of allowed delegations or delegation type). When delegation happens, the delegator shall fulfill the rules defined in his certificate, and add the same or stricter delegation rules to the delegatee certificate. The asset that checks a delegated attribute, also checks the legality of all the delegation path - signature of each certificate by its parent, its revocation status and its legality against the rules defined by the parent.

Transfer delegation. Crampton and Khambhammettu (2008) defines different types of delegation, and specifically *transfer delegation*. In transfer delegation a delegator loses the transferred attribute when the delegatee gets it. In centrally managed environment it can be done easily by indication of central authorization authority. But in PKI standards the transfer operation is not defined. AC delegation, that is implemented via processes of certificate issuance and revocation, actually allows the delegator to continue using his certificate after transfer or to transfer it to more than one delegatee. Another problem can happen when the AA wishes to demand strong transfer delegation. According to Crampton and Khambhammettu (2008) when strong transfer occurs, all attributes that were given because of the transferred attribute, will be revoked. The problem is similar to the Inference cascading revocation issue: Since the asset doesn't know that the reason certificate was transferred, it accepts the result AC that should have been revoked.

The easiest way to deal with transfer delegation problems is not to use AC Transfer delegation. In real life this type of delegation is not used widely. However, we believe that transfer extension adds important advantages in attributes' usage versatility and flexibility.

In order to solve the transfer delegation revocation issues we suggest to make two extensions to the protocol.

The first one demands creation of a suitable infrastructure. We propose here the concept of Certificate Transfer List(CTL), which is similar to Certificate Revocation List of PKI. In our proposal each Attribute Authority that issues a transferable certificate adds into it the field of CTL location. The owner of an attribute certificate gets the permission (by its attribute certificate) to add the transfer fact to the list or delete it when the transfer is stopped. The transfer list also contains the field of ID and public key of a new certificate.

When the asset gets a certificate that can be transferred, it first checks whether a transfer was already done. If yes, it checks in the CTL whether the transfer was actually done, and whether it was done to this subject. If a chain of transfers was performed, all the chain should be written to the CTL and all

CTL locations of the chain must be the same. In that way, at any moment there can be only one valid certificate with the given attribute. The transferrer can use the attribute only before he writes it to the CTL, and the transferee can only use it after. Issuance of more than one certificate can't be done, as the transferee certificate virtual identity is also part of the Transfer List.

In order to allow Strong Transfer and to allow transfer treatment in cascade inference certificates, we propose to widen the Inference extension discussed earlier. Unlike the case of regular certificates, transferred AC shall be treated as chains. That means that mandatory certificate list shall contain, in addition to regular certificates' links discussed earlier, also members of type transfer certificate. This kind of member is built of the fields of common CTL, pairs of ID and CRL, and a binary field denoting whether it is Strong Transfer. The asset that checks a transfer certificate shall scan the CTL and check that all the chain exists, all certificates are valid, and that the last ID wasn't transferred.

Figure 3 illustrates an example of a table that can be added to the AC according to our proposal. The treated scenario is of Reputation AA that provides certificates of financial risk. In this specific case, the central parameters for risk calculation were person's clearance status (proved by AC from Police Department AA in the first row), his education level (proved by AC from Ministry of Education AA in the second row, that was given based on his degree proved by AC from the Open University AA in the third row), real estate he owns (proved by strong transferable AC from Land Authority AA in the fourth row), the car he owns (proved by non-strong transferable AC from Ministry of Transport AA that was issued to someone else and transferred to him in the fifth row), and details of the bank account he uses (proved by strong transferable AC from Free Bank AA, that was issued to one person, transferred to another and then transferred to our subject). As we suggested, the rows have only IDs and links to relevant lists, so no actual value of any attribute is exposed.

The proposed changes are incorporated in the algorithms which are described and explained in the next section.

4 The New Algorithms

Our proposal leads to changes of four algorithms in the AC model.

Certificate creation should be changed to allow addition of mandatory certificates table to the AC. **Certificate delegation** process should be changed to

ID	CRL	CTL	Transfer List	Strong
4568625	https://police.gov.il/clearance/crl			
7853264	https://edu.gov.us/highschool/degrees/crl			
8924684	https://openu.ac.il/staff/crl			
9256246	https://land.gov.il/owners/crl	https://land.gov.il/owners/ctl	-	TRUE
85236472	https://transport.gov.il/owners/crl	https://transport.gov.il/owners/ctl	9374903 → 85236472	FALSE
85426642	https://FreeBank.com/accounts/crl	https://FreeBank.com/accounts/ctl	8248216 → 546582 → 85426642	TRUE

Fig. 3. Table of mandatory certificates.

include registration of transfer fact in the Transfer List. The **delegation revocation** process should be changed to include transfer status change in the CTL. **AC validity check** process should be changed to include checks of correctness of all MCs. **Regular certificate revocation** process remains the same as in classical AC protocol and therefore won't be added here.

We assume here that all CTLs and CRLs are accessible. The other case shall be treated according to asset policy and can vary in different cases.

Next we present the algorithms.

4.1 Certificate Creation

This algorithm is performed by the Attribute Authority. It doesn't deal with processes of attributes calculation and specific fields values (which is unique to each AA and each type of certificate), but only with creation of fields needed for consistency checks. The algorithm builds the MC table of a new certificate. Instruction 3.1 sets the validity of AC to be minimal of all its MCs. Instruction 3.2 adds the MC table of each MC to the table of AC (so recursively all inference chains are added). The table is built of MCs and of their MC tables' members. For transferable certificates the CTL is added to the MC. Sections 3.4–3.5 treat transfer: Signs whether the transfer is strong and add the transfer chain to the certificate.

1. Create all needed attributes
2. Define Mandatory Certificates
3. For each Certificate in Mandatory Certificates:
 - 3.1 ValidUntil = min (ValidUntil, Certificate.ValidUntil)
 - 3.2 For each Line in Certificate.MandatoryCertificatesTable
 - Add Line to MandatoryCertificatesTable
 - 3.3 Add [Certificate.ID, Certificate.CRL]
 - to MandatoryCertificateTableLine
 - 3.4 If Certificate.isTransferable and TransferType = Strong
 - 3.4.1 Add Certificate.CTL to CertificateLine.CTL
 - 3.4.2 Add all certificates IDs in transfer chain
 - to CertificateLine.TransferChainList
 - 3.4.3 CertificateLine.isStrongTransfer = TRUE
 - 3.5 Else If Certificate is Transferred
 - 3.5.1 Add Certificate.CTL to CertificateLine.CTL
 - 3.5.2 Add all certificates IDs in transfer chain
 - to CertificateLine.TransferChainList
 - 3.5.3 CertificateLine.isStrongTransfer = FALSE
 - 3.6 Add CertificateLine to MandatoryCertificatesTable

Example: We will follow the example illustrated in Fig. 3. Instruction 3.2 is relevant to AC given by Ministry of Education. Since this AC has MC table of his own, with AC given by Open University there, this Open University line will be copied to current MC table. Instruction 3.3 adds CRL locations of all ACs. Instruction 3.4 is relevant to Land AC and Free Bank AC. For both the CTL

and the sign of strong transfer are written via instruction 3.4.1. For Free Bank, the attribute is itself transferred, and therefore instruction 3.4.2 copies the IDs of its Attribute Transfer chain (8248216 that issued 546582 that issued the current certificate 85426642) to the Transfer List field. Instruction 3.5 is relevant to Transport certificate, which is transferred, though it isn't strong. Instruction 3.5.1 saves its CTL, and 3.5.2 its PKI chain (9374903 that issued current 85236472). See 4th line in the figure.

4.2 Certificate Delegation

This algorithm is performed by a delegator in order to make delegation. The delegator creates and signs the delegatee certificate, and, in case of transfer delegation, adds the fact to the CTL. Re-delegation is treated in the same way.

1. Create delegateeCertificate
2. Add delegatorCertificate to delegateeCertificate.SignersChain
3. If delegation type = Transfer
 - 3.1 If delegator already transferred certificate Return Error.
 - 3.2 Else
 - 3.2.1 delegateeCertificate.CTL = delegatorCertificate.CTL
 - 3.2.2 Add [delegatorCertificate.ID, dealeateeCertificate.ID, delegateeCertificate.signature] to TransferList -> delegatorTable
4. Sign delegateeCertificate

Example: We will follow the example of FreeBank AC illustrated in Fig. 3. Free bank gave the AC of the account owning to ID 8248216. That owner transferred it to 546582, and the latter to 85426642. Each delegator constructed and signed the delegatee AC via instructions 1, 2. Since instruction 3 is relevant, instruction 3.2.1 adds link to CTL (<https://FreeBank.com/accounts/ctl>) to ACs of delegates. After the two transfers, the FreeBank CTL will have the section for 8248216 transfers. Instruction 3.2.2 will add two lines: 8248216 - > 546582 and 546582 - > 85426642. see line 6 in the figure.

4.3 Delegation Revocation

This algorithm is performed by a delegator that wants to stop the delegation. The revocation is done by deleting all transfers from the delegatee and later from the CTL. This way each certificate transferred later will be considered invalid by the CTL check.

1. Add delegatee.CertificateID to delegatee.CRL
2. If delegationType = Transfer
 - 2.1 Delete all lines in TransferList from delegatee and further down

Example: We will follow the above example of FreeBank AC illustrated in Fig. 3. If 8248216 decides to stop the transfer, both CTL lines will be deleted.

4.4 Certificate Validation During Access Check

This algorithm is part of access check algorithm. It is performed by the asset. After it is finished correctly, the asset can use the values of the attributes signed in the certificate, to calculate the subject's permissions. The algorithm checks the general certificate validity, and for certificates that were transferred to the subject, it checks whether the transfer is still in the CTL, and for strong transferable certificates, it checks in the CTL that they were not transferred elsewhere.

1. Make full X.509 validity checks. If failed return FALSE.
2. For each Certificate in Signers chain from current to root
 - 2.1 If Parent.mode = Transfer
 - 2.1.1 If Parent.Certificate.ID not in (Parent.Certificate.CTL)
 - return FALSE
 - 2.2. For each CertificateID in Mandatory Certificate Table
 - 2.2.1 If CertificateID in list CRLLocation return FALSE
 - 2.2.2 If CTL exist
 - 2.2.2.1 If the transfer order in CTL is NOT the same as in TransferChainList return FALSE
 - 2.2.2.2 If isStrongTransfer and CertificateID in CTL and not last
 - return FALSE
3. Return TRUE

Example: We will follow the example illustrated in Fig. 3, with different revocation scenarios. Let's assume that reason AC of our Reputation AC, direct or indirect, is revoked. Since all of them are in MC table constructed by algorithm 4.1, instruction 2.2 will check their CRL, and 2.2.1 return revocation error.

Let's assume that the subject transferred his land ownership to somebody else. Algorithm 4.2 will create line in Land Authority CTL for this transfer. If he tries to use our Reputation AC, instruction 2.2.2 will check the CTL of Land Authority, and instruction 2.2.2.2 will find out that current AC is a delegator, so the algorithm will return revocation error.

Let's assume that the owner of Free Bank account decides to stop its transfer. He will change the CTL, as demonstrated in Sect. 4.3. If this happened before Reputation AA started his work, instruction 2.1.1 will return validity error and the AA won't build the certificate. However, if reputation AC already exists, the asset validity check will find the revocation via instruction 2.2.2.1.

5 Challenges and Solutions

In this section we show different challenges that are raised as a result of our proposal and suggest solutions to them.

5.1 Privacy and Confidentiality Challenge

As we've mentioned before, ABAC authorization process involves three actors: the subject, the asset and the AA. Any actor is interested in minimal disclosure of its personal data to others. The subject is interested in *Privacy*. Therefore, it would like to prevent the asset from getting any of its attributes that are not explicitly needed to authorization decisions, and to prevent the AA from discovering the usage of the attributes it issued. The AA is interested in *Confidentiality*. Therefore it would like to prevent the asset from discovering the values' calculation algorithm, and specifically, which attributes were used for the calculation. The asset is interested in *Confidentiality and Privacy*. Therefore it would like to prevent the AA from discovering which attributes were used and why.

Metadata Challenge. Our suggestions contain additions to the AC format that create a Privacy challenge. Unlike in a standard AC, our AC contains fields that point to CRLs and CTLs of MCs, and this data is available to the asset. As a result, there are a few scenarios that can lead to abuse, even if the AC is anonymous:

First scenario is of a malicious asset, which can try to find more information about a subject than required. A key to such an abuse is the fact that the link to the CRL contains meaningful meta-data: Who certified the attribute. This meta-data allows to discover authorities the AA relies on. In certain cases it is easy to guess which attribute it can be, and respectively to see which attributes the subject has. Moreover, a comparison of different ACs can help the asset to guess how the inference attributes influence the result that the AA produces, and to compromise the AA confidentiality.

Another scenario is of a malicious AA. When the AA gets a request for a CRL check from the asset, it finds out what is the subject's usage for the certificate (whether direct or by inference), violating subject's privacy. In addition, the AA that is a part of an inference chain, and gets a CRL check from a latter AA in the process of construction of the final AC, can easily use it to understand the trust algorithm of the asset and of the different AAs.

Revocation Request Challenge. Win et al. (2012) exposed a built-in collision between the demand to anonymity and the ability to make an efficient revocation. In our work we assumed that the revocation situations were initiated by the AA when it got some external information about a user. However, there are common scenarios of revocation, which involve a report of some assets to the AA, that certain subject has misused the credentials he got. The same is true in case of delegation. Unfortunately, such a report can enable the AA or the delegator to understand for which purposes the AC was used (or tried to be used). This can violate the subject's privacy and the asset confidentiality.

5.2 Performance Challenge

Personal certificates have only little impact on performance problems, since they are not used very often, and usually only during the login process. Attribute certificates are different - they can be used for any authorization demand. Therefore, the performance of the certificate check process is important. Revocation checks are time consuming, especially because of network problems. Our proposal demands increase in time consumption in three ways.

The first is by the fact that MC table is added and should be treated in each check. This increase is tiny, since it is treated entirely by and in the SP.

The second is in transfer revocation process, where in case of re-delegation the full chain of transfer shall be deleted. This increase is also very small, since it is treated entirely in the CTL, happens only once and adds only one network connection.

The third and the most significant impact is caused by the CRL and CTL check. Each row of MC table invokes such a check and demands separate network connection. Although those connections can be treated parallelly, we still add some impact on the revocation process, and wait for the last answer.

In extreme cases, the check process can be even used as a mechanism for a denial of service attacks. All CRL solutions are known for being sensitive to such attacks, especially in case of a single publish server Adams and Zuccherato (1998). In the AC case, as long as there are more validations than with regular certificates, as they are more complicated (for CTL), and are done more often - the danger of DoS is larger.

5.3 Challenges Solutions

In order to address the above challenges we suggest two different approaches. One is based on Trusted Third Party and is more centralized, and another on Semi-trusted Mediator and more connected to certificates solution. We will show how they can solve the above issues and what are the advantages of each one. However, the full comparison between them, and a detailed formalization of their usage is beyond the scope of this work.

Trusted Third Party. TTP is a concept suggested by Lou and Ren (2009). It means an external revocation proxy. Its idea is to prevent the case where any single actor knows enough information to abuse it for additional knowledge extraction. In our case, during AC creation, the AA writes the real MC table to some proxy server, and put in the AC links to that server. When an asset validates the AC, it will turn to the TTP, which will look for valid CRLs and CTLs and return a result.

The TTP is also a good platform for misbehavior reports treatment. When an asset suspects a problem with the AC, it can report it to the TTP. The TTP shall save statistics of such reports, and report to the AA that created a problematic AC, when their severity, number or any other metric reaches the predefined threshold. The AA will get the suspicions but not the assets that raised them.

In order to solve the performance issues, the TTP can act as a cache proxy. Unlike the regular proxy, the cache proxy contains not only links but also the data itself. In our case the TTP will periodically sample each MC CRL and CTL and save the general AC status. During the validation process the asset will have to make only single and simple status checks against the TTP and not all the checks against all MCs' AAs.

Cache proxy has the risk of revocation time increase. Therefore it is important to reduce it. We suggest a combination of two techniques. The first is a usage of CREV-1 algorithm described in Yap (2011) to notify the TTPs with changes. The other is a stateless subscription TTPs to CRL and CTL changes, as described in Naor and Nissim (2000). If TTPs get revocation notifications, in both ways, from the AAs and other TTPs, the risk becomes lower.

Semi-trusted Mediator. The SEM concept in classical PKI, suggested by Boneh et al. (2001) is very different from the regular CRL. It suggests to prevent the need in revocation publisher via the usage of dual-stage PKI scheme - half of the keys is given to the user and half to an on-line semi-trusted server. Both halves are needed to encrypt or decrypt a message. When some certificate is revoked all such servers are instructed not to cooperate with its holder anymore.

An adaptation to the AC case will demand an AC holder to get validity tokens for the AC itself and for each MC. All those tokens will be given to the asset during authorization process, as a validity proof. To allow that, the AA will create another pair of keys for each AC and give them to SEM.

In our adaptation we would use the concept of accumulator. It was firstly introduced by Benaloh and de Mare (1993), and adapted by Camenisch and Lysyanskaya (2002) and Camenisch et al. (2009), to an efficient revocation technique for anonymous credentials. The model is based on Zero Knowledge proof of a validity token (witness). In our case each SEM will provide accumulator service, and the MCs of an AC will be treated similarly to credentials treatment in the original model.

An important advantage of the concept is that most of the validity work is done by the client. Consequently, performance issues of an asset are reduced significantly and denial of service attacks are hard to implement. Another advantage is the fact that there is no need to use the publish infrastructure. Note also that though the performance concern is in general reduced, the SEM approach still requires more computational overhead because of the cryptographic operations during validity check.

The concept also helps to reduce privacy concerns, as there is no direct connection between the assets and the AAs. Therefore scenarios of AAs that understand the usage of ACs are prevented. In order to find a complete solution to the meta-data concerns, we suggest a naming convention for semi-trusted servers that doesn't involve AA information and usage of the same SEM for different AAs.

The issue of misbehavior report is not solved natively in this approach. In order to treat it we suggest to send reports to the SEM server. We also propose that during the creation of an AC, the AA shall instruct the SEM on which misbehavior thresholds to stop cooperate with an AC.

6 Summary and Future Work

ABAC certificates enable decentralized and flexible mechanism for treatment of the authorization process. Currently their revocation and transfer processes can lead to inconsistency, and therefore should be limited. Our proposal allows to solve this inconsistency and provides a flexible protocol for revocation and delegation. On the other hand, the proposal raises privacy and performance concerns, and we presented some ideas for their treatment. It is also a base for future work that can extend the current model to become a new full secure model for ABAC certificates.

A basic further work on the suggested model is its extension to more complicated cases of dependencies between certificates. Our current model treats a basic case where any ancestor revocation shall lead to a descendant revocation. However, it is not always the case, due to the fact that ABAC allows the AA to make decisions based on a group of provided certificates. It is clear that a correct treatment of complicated dependencies will involve creation of an assertion language that shall allow the AA to create description of checks to revocation. This language will have all classical logical operators, such as OR, AND, One Of, etc. In addition, further work shall explore an issue of a treatment that will reduce the ability of a conspiracy between different actors in order to find more information. Our model prevents an ability of a single actor to abuse the data it gets, but it can and should be widened to enhance confidentiality and privacy of all the actors when different participants collude to get extra information.

In general, we believe that our suggestions and their future extensions can make AC usage more flexible and secure, and therefore allow ABAC approach to be accepted in scenarios in which it is limited today.

References

- McCollum, C.J., Messing, J.R., Notargiacomo, L.: Beyond the pale of MAC and DAC-defining new forms of access control. In: Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy, pp. 190–200. IEEE (1990)
- Adams, C., Zuccherato, R.: A general, flexible approach to certificate revocation. Entrust Technologies White Paper (1998)
- Blaze, Matt, Feigenbaum, Joan, Keromytis, Angelos D.: KeyNote: trust management for public-key infrastructures. In: Christianson, Bruce, Crispo, Bruno, Harbison, William S., Roe, Michael (eds.) Security Protocols 1998. LNCS, vol. 1550, pp. 59–63. Springer, Heidelberg (1998). https://doi.org/10.1007/3-540-49135-X_9
- Rivest, R.L.: Can we eliminate certificate revocation lists? In: Hirschfeld, R. (ed.) FC 1998. LNCS, vol. 1465, pp. 178–183. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0055482>
- Housley, R., Ford, W., Polk, W., Solo, D.: RFC 2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile (1999)
- Boneh, D., Ding, X., Tsudik, G., Wong, C.M.: A method for fast revocation of public key certificates and security capabilities. In: USENIX Security Symposium, p. 22, August 2001

- Li, N., Grosf, B.N., Feigenbaum, J.: Delegation logic: a logicbased approach to distributed authorization. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **6**(1), 128–171 (2003)
- Linn, J., Nystrom, M.: Attribute certification: an enabling technology for delegation and role-based controls in distributed environments. In: *Proceedings of the Fourth ACM Workshop on Role-Based Access Control* (1999)
- Xiong, L., Liu, L.: PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities. *IEEE Trans. Knowl. Data Eng.* **16**(7), 843–857 (2004)
- Ye, C., Wu, Z., Fu, Y.: An attribute-based delegation model and its extension. *J. Res. Pract. Inf. Technol.* **38**(1), 3–18 (2006)
- Naor, M., Nissim, K.: Certificate revocation and certificate update. *IEEE J. Sel. Areas Commun.* **18**(4), 561–570 (2000)
- Farrell, S., Housley, R., Turner, S.: RFC 5755: An Internet Attribute Certificate Profile for Authorization. IETF (2010)
- Thompson, M.R., Essiari, A., Mudumbai, S.: Certificate-based authorization policy in a PKI environment. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **6**(4), 566–588 (2003)
- Lou, W., Ren, K.: Security, privacy, and accountability in wireless access networks. *IEEE Wirel. Commun.* **16**(4), 80–87 (2009)
- Win, L.L., Thomas, T., Emmanuel, S.: Privacy enabled digital rights management without trusted third party assumption. *IEEE Trans. Multimed.* **14**(3), 546–554 (2012)
- Sufatrio, Yap, R.H.: Trusted principal-hosted certificate revocation. In: Wakeman, I., Gudes, E., Jensen, C.D., Crampton, J. (eds.) *IFIPTM 2011. IFIPAICT*, vol. 358, pp. 173–189. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22200-9_15
- Hinarejos, M.F., Munoz, J.L., Forne, J., Esparza, O.: PREON: an efficient cascade revocation mechanism for delegation paths. *Comput. Secur.* **29**(6), 697–711 (2010)
- Crampton, J., Khambhammettu, H.: Delegation in role-based access control. *Int. J. Inf. Secur.* **7**(2), 123–136 (2008)
- Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1. 1. Technical report, Microsoft Corporation (2011)
- Benaloh, J., de Mare, M.: One-way accumulators: a decentralized alternative to digital signatures. In: Helleseth, T. (ed.) *EUROCRYPT 1993. LNCS*, vol. 765, pp. 274–285. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_24
- 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). https://doi.org/10.1007/3-540-45708-9_5
- Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) *PKC 2009. LNCS*, vol. 5443, pp. 481–500. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00468-1_27

Formal Verification



Formal Analysis of Combinations of Secure Protocols

Elliott Blot¹, Jannik Dreier²(✉), and Pascal Lafourcade¹

¹ LIMOS, University Clermont Auvergne, Clermont-Ferrand, France
elliott.blot@gmail.com, pascal.lafourcade@uca.fr

² LORIA, Université de Lorraine, INRIA, CNRS, Nancy, France
jannik.dreier@loria.fr

Abstract. When trying to prove the security of a protocol, one usually analyzes the protocol in isolation, i.e., in a network with no other protocols. But in reality, there will be many protocols operating on the same network, maybe even sharing data including keys, and an intruder may use messages of one protocol to break another. We call that a *multi-protocol attack*. In this paper, we try to find such attacks using the Tamarin prover. We analyze both examples that were previously analyzed by hand or using other tools, and find novel attacks.

1 Introduction

When analyzing the security of protocols, one aims at proving specific security properties. The most common types of properties are *secrecy*, meaning that an intruder cannot know a secret value, and *authentication*, meaning that if A thinks he is talking to B , then he is really talking to B . In our digitalized world there are more and more cryptographic protocols everywhere, and we want to verify them to ensure their security.

It is not realistic to assume that a protocol is running alone in the network, and in the real world, an intruder can try to use messages of other protocols in the network to break a protocol. That is what we call a *multi-protocol attack*.

More precisely, we study the following problem of multi-protocols attacks. Given two protocols that ensure a certain security property in isolation, are they still safe for this property if we put them together in the same network? Unsurprisingly there exist many combinations of protocols where this is not the case, i.e., where we can mount multi-protocols attacks.

There are a lot of tools for automatic analysis of security properties, like ProVerif [3], AVISPA [2], Athena [27], Scyther [12], or Tamarin [23]. But they are generally used to analyze the security of a protocol executed in isolation,

This research was conducted with the support of the Indo-French Centre for the Promotion of Advanced Research (IFCPAR) and the Center Franco-Indien Pour La Promotion De La Recherche Avancée (CEFIPRA) through the project DST/CNRS 2015-03 under DST-INRIA-CNRS Targeted Programme, and by the CNRS PEPS SISC ASSI 2016/2017.

meaning that each agent only executes the analyzed protocol. In this paper, our aim is also to automatically find multi-protocols attacks using Tamarin.

Contributions: Several multi-protocols attacks have been found manually or using other tools, our aim is to find them automatically using the Tamarin prover [23]. Our contributions are the following:

- We automatically find all the manual attacks described in [22]. Moreover, we find novel different attacks on the same properties, or unknown attacks on different properties. This demonstrates the limitations of a manual approach for finding attacks. It underlines that automatic verification is a very efficient approach for analyzing the security of cryptographic protocols.
- We analyzed all the protocols given in [9], where the authors used Scyther, a different protocol verification tool. We confirm the results from Scyther using Tamarin.
- We developed an algorithm to simplify the process of creating the multi-protocol specification file in Tamarin from the individual protocol specifications. The algorithm also automatically generates necessary helping lemmas in Tamarin in order to verify the combination of the two protocols more efficiently. The algorithm is implemented in Python, and available online [15].

Related work: The existence of multi-protocol attack have been introduced by Kelsey, Schneier, and Wagner in [18]. In this paper the attacks were found manually and the authors consider protocols crafted to break other protocols.

In [22], Mathuria, Raj Singh, Venkata Sharavan, and Kirtankar found six multi-protocol attacks based on 13 protocols from literature: Denning-Sacco protocol [13], amended Woo-Lam protocol [5], ISO Five-Pass protocol [7], Abadi-Needham protocol [1], six protocols from Perrig and Song using APG [26], and three protocols from Zhou and Foley using ASPB [30]. In contrast to these works, we use an automatic verification tool to find these attacks.

Cremers found many multi-protocol attacks in [9], using the tool Scyther, with 30 protocols from literature including Needham Schroeder protocol [24], Needham Schroeder symmetric key protocol [24], Needham Schroeder symmetric key amended protocol [25], Lowe’s modified version of the Needham Schroeder protocol [19], SPLICE/AS [29], Hwang and Chen’s version of SPLICE/AS [16], Clark and Jacob’s version of SPLICE/AS [8], a basic SOPH example (Secret-Out Public-Home), Woo Lam pi f [28], Kao Chow v.1, v.2 and v.3 [17], Yahalom’s protocol [4], and Lowe’s version of Yahalom protocol [21]. Compared to this work we use the Tamarin instead of Scyther.

There is also a considerable amount of work of preventing multi-protocol attacks by construction using special composition frameworks. These frameworks exist in the computational (e.g., Universal Composability [6]) and in the symbolic setting (e.g., Protocol Composition Logic [14]).

Outline: The paper is structured as follows. In Sect. 2, we present the results we obtain and we compare them with those obtained manually in [22] or using Scyther [9]. Then, Sect. 3 discusses our workflow in Tamarin, and finally the last section concludes the paper.

2 Multi-protocol Attacks

First we define the properties that we want to verify for each protocol. We define one property for secrecy and two authentication properties.

- *Secrecy* [10]: if A claims the secrecy of a variable N_A at the end of the run of the protocol, then an intruder cannot know this variable.
- *Non-injective agreement* [11]: if a protagonist A completes a run apparently with B , then B has run the protocol with A and A agrees with all other protagonist on all values. This is not exactly the same definition as in [20], but we keep this definition because it is this one that is used by Scyther.
- *Non-injective synchronisation* [11]: if a protagonist A completes a run as the initiator apparently with B as the responder, then B has run the protocol as the responder with A , and all messages sent and received are exactly like described in the specification of the protocol, in the same order.

We call a type-flaw attack an attack where the intruder uses data of a different type than the data expected by the protocol. For example, in such an attack, the intruder could use two nonces N_1, N_2 instead of another single nonce N ($N = (N_1, N_2)$), or uses an ID as a nonce. We consider separately the case where the intruder can make type-flaw attacks (such attacks are valid) and the case where the intruder cannot (such attacks are not valid).

All our Tamarin files are available online [15].

2.1 Attacks by Cremers [9]

First we study the protocols analyzed in [9] using Scyther. We modeled all these protocols individually in Tamarin. Figure 1 presents our results using Tamarin for the properties described previously, and Fig. 2 presents our results for multi-protocols using Tamarin, where we verify the properties for the first of the two protocols. In these figures, ni-synch stands for non-injective synchronisation, sec stands for secrecy and ni-agree stands for non-injective agreement. Moreover, ✓ means that we did not find any attacks, and ✗ means there is at least one attack for the property. A yellow box means that the first protocol (the one for which we verify the security properties in the combination) is safe for this property in isolation, and red box means that both protocols are safe for this property in isolation. Empty box means that the property is not relevant for this protocol, for example the key K_{AB} does not exist in the protocol in the property secrecy $A K_{AB}$ and secrecy $B K_{AB}$, or a protagonist A never obtains a nonce N_B in the property secrecy $A N_B$.

We find the same results with Tamarin as with Scyther. We do not consider type-flaw attack for these protocols, because the number of combination with multi-protocol attack is too large (more than 100 different combinations) to model them all manually with Tamarin. All timings are calculated with 6 CPUs of 2 GHz and 32 GB of memory.

We can see in Fig. 2 that even if two protocols are safe in isolation for a property, it is not guaranteed that the combination of this two protocols is safe

name	ni-synch A	ni-synch B	sec A N_A	sec B N_A	sec A N_B	sec B N_B	sec A K_{AB}	sec B K_{AB}	ni-agree A	ni-agree B
NSS	0:07 ✓	0:06 ✓	0:01 ✗	-	0:02 ✓	0:01 ✓	0:02 ✓	0:01 ✓	0:03 ✓	0:02 ✓
NSSA	0:16 ✗	0:37 ✗	0:05 ✗	-	0:35 ✓	10:28 ✓	0:03 ✓	0:06 ✓	0:03 ✗	0:02 ✗
NS	0:28 ✓	0:05 ✗	0:01 ✓	0:01 ✗	0:08 ✓	0:02 ✗	-	-	0:12 ✓	0:04 ✗
NSL	0:22 ✓	1:07 ✓	0:01 ✓	0:04 ✓	0:07 ✓	0:02 ✓	-	-	0:08 ✓	0:17 ✓
AS	0:09 ✗	0:04 ✗	0:03 ✗	0:02 ✗	-	0:02 ✗	-	-	0:11 ✗	0:01 ✗
AShc	0:05 ✗	0:05 ✗	0:04 ✗	0:03 ✗	-	0:03 ✗	-	-	0:45 ✗	0:02 ✗
K	0:06 ✗	0:15 ✗	0:00 ✗	0:16 ✗	0:03 ✗	0:01 ✗	0:02 ✓	0:32 ✓	0:07 ✗	0:12 ✗
K2	0:05 ✗	1:40 ✗	0:04 ✗	0:30 ✗	0:04 ✗	0:30 ✗	0:04 ✓	0:21 ✓	0:04 ✗	0:40 ✗
K3	0:03 ✗	5:43 ✗	0:02 ✗	0:02 ✗	0:04 ✗	0:05 ✗	0:05 ✓	2:27 ✓	0:02 ✗	4:26 ✗
WLPif	0:00 ✗	0:01 ✗	-	-	0:00 ✗	0:00 ✗	-	-	0:00 ✗	0:01 ✗
Y	0:04 ✗	5:53 ✗	0:04 ✗	0:12 ✗	0:24 ✓	0:13 ✗	0:09 ✓	0:12 ✗	0:03 ✗	4:18 ✗
YL	0:12 ✓	0:32 ✓	0:01 ✗	0:02 ✗	0:07 ✓	0:15 ✓	0:05 ✓	0:11 ✓	0:06 ✓	0:17 ✓
AScj	0:06 ✗	0:25 ✗	0:05 ✗	0:05 ✓	-	0:01 ✗	-	-	0:02 ✗	0:06 ✗
soph	0:00 ✓	0:01 ✗	0:00 ✗	0:01 ✗	-	-	-	-	0:00 ✓	0:01 ✗

Fig. 1. Results found using Tamarin with NS = Needham Schroeder [24], NSS = Needham Schroeder Symmetric Key [24], NSSA = Needham Schroeder Symmetric Key Amended [25], NSL = Needham Schroeder Lowe [19], AS = SPLICE/AS [29], AShc = Hwang and Chen version of SPLICE/AS [16], AScj = Clark and Jacob version of SPLICE/AS [8], K = Kao Chow [17], K2 = Kao Chow v.2 [17], K3 = Kao Chow v.3 [17], WLPif = Woo Lam pi f [28], Y = Yahalom [4], YL = Yahalom Lowe [21], soph = a SOPH basic example. ni-synch denotes non-injective synchronisation, ni-agree denotes non-injective agreement, and sec $A N_A$ denotes the fact that A claims the secrecy of N_A .

too if they share keys, and multi-protocol attacks are not only due to the other protocol that is not safe for this property.

We would expect that Tamarin takes more time to analyze properties for multi-protocols than for protocols in isolation, due to the increased number of transitions and the larger number of traces with the new protocol.

name	ni-synch A	ni-synch B	sec A N_A	sec B N_A	sec A N_B	sec B N_B	sec A K_{AB}	sec B K_{AB}	ni-agree A	ni-agree B
NSS + NSSA	0:53 ✗ *	1:35 ✗ *	0:12 ✗	-	10:39 ✓	0:32 ✗ **	0:05 ✓	1:34 ✗ **	0:32 ✗ *	0:24 ✗ *
NS + AS	0:40 ✓	0:09 ✗	0:01 ✓	0:02 ✗	0:14 ✗ **	0:03 ✗	-	-	0:16 ✓	0:06 ✗
NS + AShc	0:39 ✓	0:08 ✗	0:01 ✓	0:02 ✗	0:14 ✗ **	0:03 ✗	-	-	0:18 ✓	0:07 ✗
NSL + AS	0:08 ✓	4:45 ✓	0:02 ✓	0:13 ✓	0:08 ✗ **	0:08 ✗ *	-	-	0:04 ✓	0:53 ✓
NSL + AShc	0:28 ✓	18:44 ✓	0:01 ✓	0:21 ✓	0:19 ✗ **	0:14 ✗ *	-	-	0:11 ✓	0:23 ✓
K + WLPif	0:03 ✗	0:11 ✗	0:02 ✗	0:03 ✗	0:02 ✗	0:03 ✗	0:01 ✓	0:03 ✗ **	0:03 ✗	0:06 ✗
K2 + WLPif	0:05 ✗	0:22 ✗	0:03 ✗	0:04 ✗	0:03 ✗	0:03 ✗	0:05 ✓	0:03 ✗ **	0:07 ✗	0:15 ✗
K3 + WLPif	0:03 ✗	0:23 ✗	0:02 ✗	0:32 ✗	0:02 ✗	0:03 ✗	0:11 ✓	0:03 ✗ **	0:02 ✗	0:29 ✗
YL + Y	1:52* ✗	3:12* ✗	1:03 ✗	1:01 ✗	2:21 ✓	19:37 ✓	1:15 ✓	6:48 ✓	1:01 ✗ *	6:43 ✗ *
AScj + soph	0:15 ✗	0:06 ✗	0:11 ✗	0:09 ✗ *	-	0:02 ✗	-	-	0:04 ✗	0:04 ✗

Fig. 2. Result found with Tamarin. NS = Needham Schroeder [24], NSS = Needham Schroeder Symmetric Key [24], NSSA = Needham Schroeder Symmetric Key Amended [25], NSL = Needham Schroeder Lowe [19], AS = SPLICE/AS [29], AShc = Hwang and Chen’s version of SPLICE/AS [16], AScj = Clark and Jacob’s version of SPLICE/AS [8], K = Kao Chow [17], K2 = Kao Chow v.2 [17], K3 = Kao Chow v.3 [17], WLPif = Woo Lam pi f [28], Y = Yahalom [4], YL = Yahalom Lowe [21], soph = a SOPH basic example. ni-synch denotes non-injective synchronisation, ni-agree denotes non-injective agreement, and sec A N_A denotes fact that A claims the secrecy of N_A . * = the first protocol is safe in isolation, ** = both protocol are safe in isolation (Color figure online)

But as we can see in Figs. 1 and 2, this is not always the case, like for example for the property ni-synch A for Kao Chow (K) in Fig. 1, and for Kao Chow + Woo Lam pi f (K + WLPif) in Fig. 2. This is generally due to the fact that Tamarin finds an attack more rapidly than a proof as Tamarin stops after the first attack it finds (it does not try to find all attacks).

It can also happen that Tamarin proves a property for the combination of protocols more quickly than for the protocols in isolation, like for example Needham Schroeder Lowe in Fig. 1 and Needham Schroeder Lowe + SPLICE/AS in Fig. 2 for ni-synch A . This can occur for example if the precomputations are the dominating part of the total runtime.

2.2 Attacks by Mathuria et al. [22]

We try to find the attacks described in [22] using Tamarin, to see if we find the same or different attacks if we use an automatic tool. The properties verified are not clearly defined in [22], so we keep the properties as defined previously. More precisely, we verified different authentication properties: non-injective synchronization, non-injective agreement, and a weaker agreement property. The property non-injective agreement as define previously is too strong to get comparable result with the paper, most of protocols of the paper are not safe for this property even in isolation. So we consider a weaker authentication property defined as follows:

- *weaker agreement*: if B thinks that a nonce N_A is generated by A , then A has generated N_A and B authenticates A (called Aut A in Figs. 3 and 4).

Figure 3 summarizes results that we obtain with Tamarin in isolation on protocols from [22], and Fig. 4 summarizes results we obtain for the multi-protocols. As previously, ni-synch stands for non-injective synchronization, sec stands for secrecy and ni-agree stands for non-injective agreement. Moreover ✓ means that we did not find any attacks, and ✗ means there is at least one attack for the property. A yellow box means that the first protocol (the one for which we verify security in the combination) is safe for this property in isolation, and red box means that both protocols are safe for this property in isolation. An empty box means that the property is not relevant for this protocol.

We can see in Fig. 3 in the case of APG.3 for non-injective synchronization and non-injective agreement, all attacks which we found in isolation are type-flaw attacks, and the protocol is safe if we do not consider type-flaw attacks. But attacks we found for APG.3 with APG.2 are not type flaw attacks (see 2.2), so we consider type-flaw attacks separately in this paper. But in the case of ZF.2, we have a protocol that is not safe for any property, considering type-flaws or not. So it is useless to see if ZF.2 can have a multi-protocol attack for a property in combination with an other protocol, a point that the authors of the original paper missed most likely since they searched for attacks manually.

The property weaker agreement seems to be closest to the property used in [22], because we found the same attacks for some protocols. Thus, in rest of the paper, we only present attacks on this property.

In comparison to the original paper we have found, using Tamarin, sometimes different attacks, and sometimes new attacks on the authentication of other protagonists in the same combination of protocols. This is likely due to the fact that Mathuria et al. searched for attacks manually, and were thus probably unable to analyze all combinations in detail or missed attacks in their analysis.

In all protocols, we have three participants, A the initiator, B the responder, and S the trusted server. We use symmetric encryption, so S shares the key K_{AS} (respectively K_{BS}) with A (respectively B). Moreover, K_{AB} denotes the session key between A and B , and N_A (respectively N_B) a nonce generated by A (respectively B). Then $\{M\}_K$ denotes the cipher-text obtained by encrypting a message M with the symmetric key K . As in the original paper, we assume

name	ni-synch A	ni-synch B	sec A N_A	sec B N_A	sec A N_B	sec B N_B	sec A K_{AB}	sec B K_{AB}	ni-agree A	ni-agree B	Aut A	Aut B
APG.1	0:14 ✓	0:03 ✓	0:01 ✗	0:01 ✗	0:08 ✗	0:01 ✗	-	-	0:06 ✓	0:01 ✓	0:01 ✓	0:04 ✓
APG.2	0:09 ✓	0:03 ✓	0:01 ✗	0:01 ✗	0:05 ✗	0:01 ✗	-	-	0:05 ✓	0:01 ✓	0:01 ✓	0:02 ✓
APG.3	0:06 ✗*	0:05 ✗*	0:01 ✗	0:01 ✗	0:08 ✗	0:02 ✗	-	-	0:05 ✗*	0:02 ✗*	0:01 ✓	0:06 ✓
APG.4	0:37 ✓	0:21 ✓	0:04 ✗	0:04 ✗	0:18 ✓	0:01 ✓	0:13 ✓	0:04 ✓	0:23 ✓	0:09 ✓	0:06 ✓	0:06 ✓
APG.5	0:48 ✓	25:57 ✓	0:05 ✗	0:03 ✗	0:20 ✓	2:04 ✓	0:14 ✓	6:18 ✓	0:30 ✓	11:37 ✓	0:29 ✓	0:00 ✓
APG.6	0:04 ✗	0:06 ✗	0:01 ✗	0:02 ✗	0:02 ✗	0:02 ✗	0:06 ✓	0:04 ✓	0:04 ✗	0:02 ✗	0:02 ✓	0:04 ✓
DS	0:01 ✗	0:01 ✗	-	-	-	-	0:01 ✓	0:01 ✓	0:01 ✓	0:01 ✗	0:00 ✓	-
AWL	0:01 ✗	0:01 ✗	-	-	0:00 ✗	0:00 ✗	-	-	0:00 ✗	0:00 ✗	0:00 ✗	0:00 ✓
ISO5	0:19 ✓	0:33 ✓	0:02 ✗	0:03 ✗	0:04 ✓	0:09 ✓	0:01 ✓	0:09 ✓	0:05 ✓	0:16 ✓	0:06 ✓	0:04 ✓
AN	0:01 ✗	0:01 ✗	0:00 ✗	0:01 ✗	-	0:01 ✗	0:01 ✓	0:02 ✓	0:00 ✗	0:00 ✗	0:01 ✗	0:00 ✗
ZF.1	0:16 ✗	5:01 ✗	0:05 ✗	0:24 ✗	0:06 ✗	0:18 ✗	0:13 ✗	1:08 ✓	0:02 ✗	0:39 ✓	0:11 ✓	0:06 ✗
ZF.2	0:01 ✗	0:06 ✗	0:01 ✗	0:01 ✗	0:01 ✗	0:01 ✗	0:01 ✗	0:01 ✗	0:01 ✗	0:04 ✗	0:07 ✗	0:01 ✗
ZF.3	28:25 ✓	1:08 ✗	0:39 ✗	0:12 ✗	0:33 ✗	0:11 ✗	4:50 ✓	0:57 ✗	2:10 ✓	0:22 ✗	0:47 ✓	0:35 ✓

Fig. 3. Results found with Tamarin with APG from [26], DS = Denning Sacco [13], AWL = Amended Woo Lam [5], ISO5 = ISOFive-Pass [7], AN = Abadi Needham [1], ZF from [30], * = only type-flaw attacks.

that each participant shares the same key for both protocols. In the following, when we talk about authentication, we mean non-injective agreement.

In the following we discuss our results in details. First we discuss attacks that we found and that differ from those presented in [22], then we present new attacks for properties that were not analyzed in [22].

Different Attacks

APG.4 with APG.6: The first attack is on the authentication of A . In this attack, two protagonists A and A' initiate the *APG.6* [26] protocol with B , and the intruder C pretends to be A in *APG.4* [26]. In the protocol initiated by A' , C learns $(N_{A'}, N'_B, A')$, used as a session key, and its ciphertext $\{N_{A'}, N'_B, A'\}_{K_{BS}}$. In the protocol initiated by A , C learns the nonce N_B , used to authenticate to B .

name	ni-synch A	ni-synch B	sec A N_A	sec B N_A	sec A N_B	sec B N_B	sec A K_{AB}	sec B K_{AB}	ni-agree A	ni-agree B	Aut A	Aut B
APG.1 + APG.2	0:01 ✗ **	0:08 ✗ **	0:01 ✗	0:01 ✗	0:01 ✗	0:02 ✗	-	-	0:01 ✗ **	0:03 ✗ **	0:02 ✗ **	0:01 ✗ **
APG.3 + APG.2	0:01 ✗	0:08 ✗	0:01 ✗	0:03 ✗	0:02 ✗	0:03 ✗	-	-	0:01 ✗	0:03 ✗	0:03 ✗ **	0:01 ✗ **
APG.4 + APG.6	0:18 ✗ **	1:20:54 ✗ *	0:05 ✗	0:06 ✗	4:54 ✓	53:41 ✗ *	2:09 ✓	1:27:40 ✗ **	0:15 ✗ *	1:00:48 ✗ *	4:27 ✗ **	0:28 ✗ **
APG.5 + APG.6	0:08 ✗ **	28:40 ✗ *	0:07 ✗	0:03 ✗	2:05 ✓	0:02 ✗ *	1:23 ✓	0:03 ✗ **	0:04 ✗ *	23:28 ✗ *	4:45 ✗ **	0:07 ✗ **
DS + AWL	0:01 ✗	0:01 ✗	-	-	-	-	0:01 ✗ **	0:01 ✗ **	0:01 ✗ *	0:01 ✗ *	0:01 ✗ *	-
ISO5 + AN	0:03 ✗ *	0:09 ✓	0:01 ✗	0:03 ✗	0:01 ✗ **	0:04 ✓	0:01 ✗ **	0:06 ✗ **	0:02 ✗ *	0:03 ✗ *	0:04 ✓	0:01 ✗ *
ZF.2 + ZF.1	0:01 ✗	0:39 ✗	0:07 ✗	0:12 ✗	0:06 ✗	0:12 ✗	0:05 ✗	0:12 ✗	0:01 ✗	0:33 ✗	3:52 ✗	0:11 ✗
ZF.3 + APG.2	2:48 ✗ **	2:17 ✗	0:08 ✗	1:19 ✗	4:01 ✗	1:20 ✗	0:07 ✗ **	3:52 ✗	59:21 ✗ **	1:34 ✗	1:29:39 ✓	0:38 ✗ **

Fig. 4. Results found with Tamarin with APG from [26], DS = Denning Sacco [13], AWL = Amended Woo Lam [5], ISO5 = ISOFive-Pass [7], AN = Abadi Needham [1], ZF from [30], * = the first protocol is safe in isolation, ** = both protocols are safe in isolation. (Color figure online)

In Fig. 5, steps on the left hand side are steps of *APG.4*, and steps on the right hand side are steps from *APG.6*.

This attack is a type-flaw attack, because the intruder uses $(N_{A'}, N'_{B'}, A')$ as a session key. So we blocked type-flaw attacks in Tamarin to see if there are other types of attacks, and we did not find other attacks on the authentication of A .

Denning-Sacco with Amended Woo-Lam: This attack is on the authentication of A . Again, it is a type-flaw attack, because the intruder uses (K_{AB}, T) as a nonce. In this attack, the intruder C plays the role of A and S in both protocols. First, B initiates a protocol *Woo-Lam* [5] with C who impersonates A . Then C sends the ID of A and a fake session key and a timestamp as a nonce. B encrypts that and C has now a valid message to send to B in *Denning-Sacco* [13]. This attack is described in Fig. 6.

We did not find other types of attacks for this protocol when we blocked type-flaw attacks in Tamarin using a modified model.

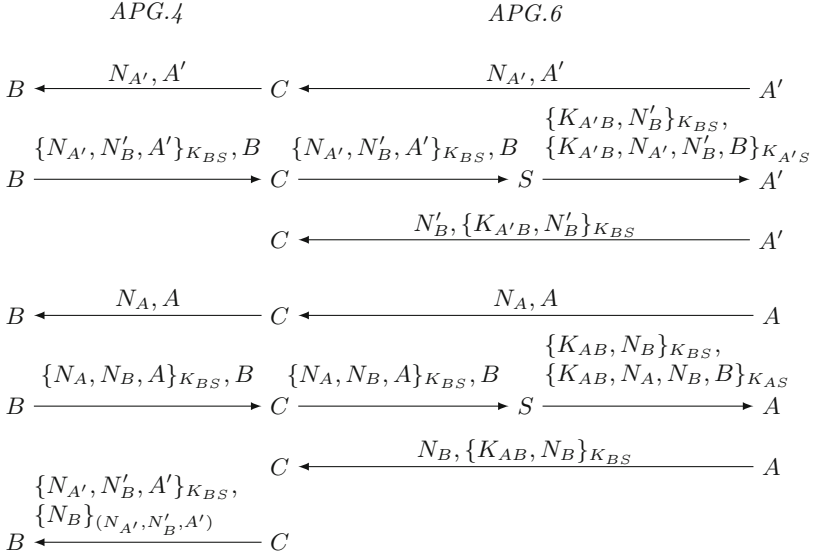


Fig. 5. Representation of the attack on APG.4 with APG.6.

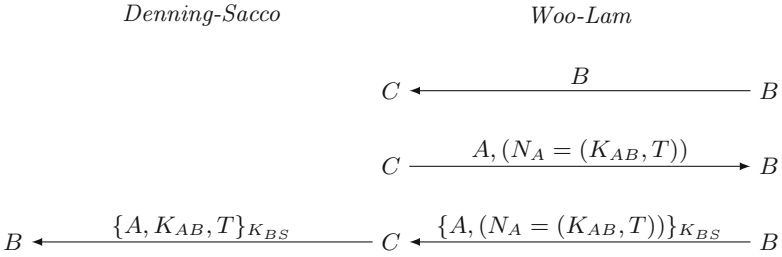


Fig. 6. Representation of the attack on Denning-Sacco with Woo-Lam.

New Attacks

APG.1 with APG.2: The attack described in [22] is an attack on the authentication of B , but we also found an attack on the authentication of A . In this attack, the intruder C plays the role of A in both protocols. First, B runs the *APG.2* [26] protocol as the initiator and then the protocol *APG.1* [26] as the responder. C can pretend to be A in *APG.1* and B will accept. In Fig. 7 steps at the left are steps from *APG.1*, and the right part are steps from *APG.2*.

APG.3 with APG.2: This attack is an attack on the authentication of B . This attack is possible if A runs the *APG.3* [26] protocol as the initiator, and *APG.2* [26] as the responder. In this attack, C plays the roles of B and S in both protocols. Then C can pretend to be B in *APG.3*, and A will accept. In Fig. 8 steps at the left are steps of *APG.3*, and at the right part are steps from *APG.2*.

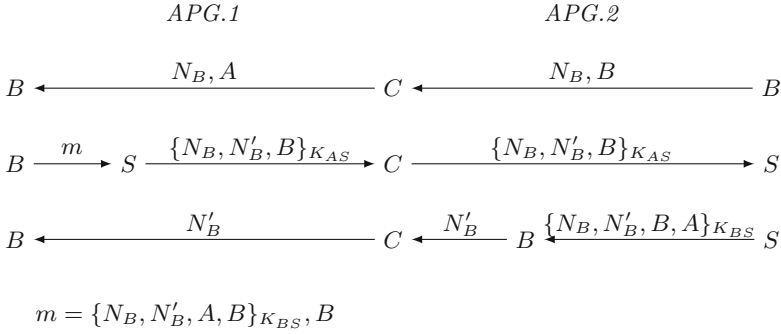


Fig. 7. Representation of the attack on APG.1 with APG.2.

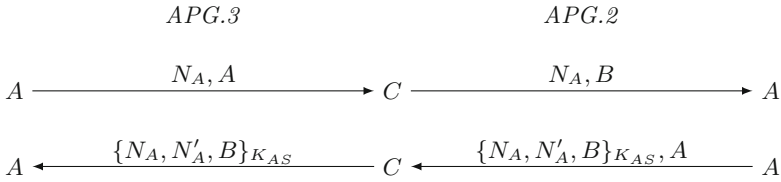


Fig. 8. Representation of the attack on APG.3 with APG.2.

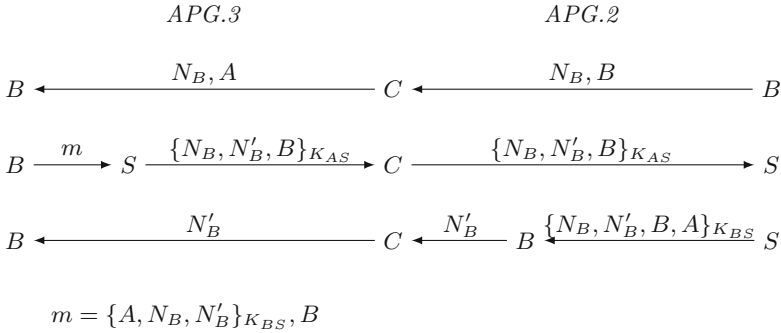
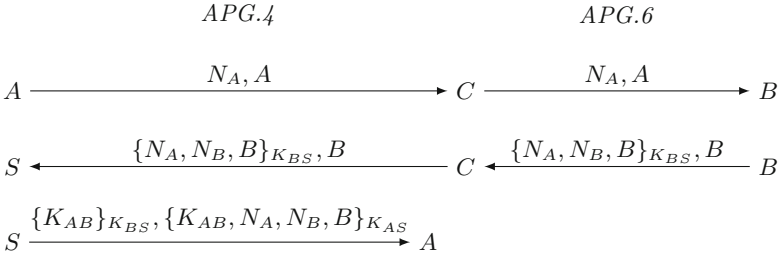
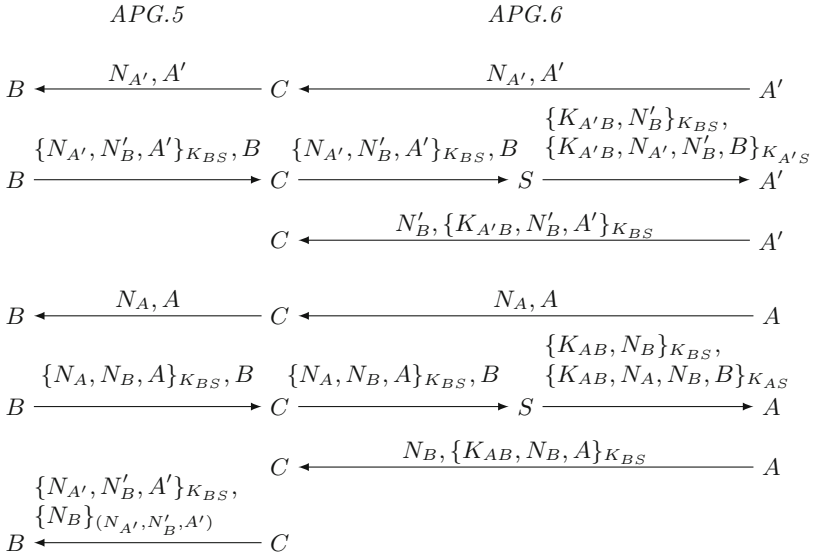


Fig. 9. Representation of the attack on APG.3 with APG.2.

We also found an attack on the authentication of *A*. In this attack, the intruder *C* plays the role of *A* in both protocols. *B* runs the *APG.2* protocol as the initiator, and *APG.3* as the responder. *C* can pretend to be *A*, and *B* in *APG.3* will accept. In Fig. 9 steps at the left are steps from *APG.3*, and steps at the right are steps from *APG.2*.

APG.4 with APG.6: We found an attack on the authentication of *B*. In this attack, *A* initiates the protocol *APG.3*, then the intruder *C* will initiate *APG.6* with *B*, using data sent by *A* in the other protocol. Finally, *C* sends the answer

**Fig. 10.** Representation of the attack on *APG.4* with *APG.6*.**Fig. 11.** Representation of the attack on *APG.5* with *APG.6*.

of B to the server in *APG.4*, and lets the protocol run. In Fig. 10, steps on the left hand side are steps from *APG.4*, and steps on the right hand side are steps from *APG.6*.

This attack is possible because the message from *APG.6* used for this attack is also used in *APG.4*, so C can get a response from B , while B does not act in *APG.4*.

APG.5 with APG.6: This attack is on the authentication of A . In this attack, two protagonists A and A' initiate the *APG.6* [26] protocol with B , and the intruder C pretends to be A in *APG.5* [26]. In the protocol initiated by A' , C learns $(N_{A'}, N'_B, A')$, used as a session key, and its encrypted version $\{N_{A'}, N'_B, A'\}_{K_{BS}}$. In the protocol initiated by A , C learns the nonce N_B , used to authenticate to B . In Fig. 11, steps at the left part are steps of *APG.5*, and steps on the right are steps from *APG.6*.

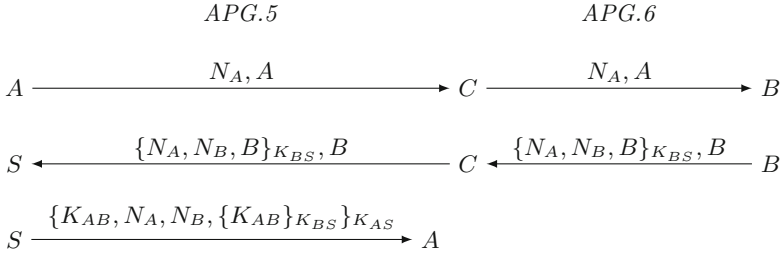


Fig. 12. Representation of the attack on APG.5 with APG.6.

This attack is a type-flaw attack. So we changed our model to disable such type-flaw attacks in Tamarin to see if there are other types of attacks, and we did not find another attack on the authentication of A .

We also found an attack on the authentication of A where the intruder uses $(N_{A'}, N'_{B'}, A')$ as a session key. In this attack, A initiates the protocol $APG.5$, then the intruder C will initiate $APG.6$ with B , using data sent by A in the other protocol. Finally, C sends the answer of B to the server in $APG.5$, and lets the protocol run. In Fig. 12, steps on the left hand side are steps from $APG.5$, and on the right hand side are steps from $APG.6$.

This attack is possible because the message from $APG.6$ used for this attack is also used in $APG.5$, so C can get a response from B , while B does not act in $APG.5$.

3 Workflow in Tamarin

As we had to write many different combinations of multiple protocols to obtain our results, we tried to simplify the process by adopting the following workflow to combine to protocols:

1. Specify each protocol individually and check the properties in isolation using Tamarin.
2. Generate the files for all the required combinations using the individual specifications.
3. Verify the combined protocols, and compare the results to known results.

To simplify the process of generating the combined specifications, we adopted certain (mostly syntactic) conventions when specifying the protocols. These mostly concern the common setup rules (key distribution etc.), the placement of labels, and the uniqueness of labels to avoid conflicts.

These conventions allowed us to develop an algorithm that can generate the input files of the composed protocols based on the individual specifications, including intermediate lemmas that simplify the analysis for Tamarin by removing undesirable cases for the subsequent analysis. The generation of these lemmas goes beyond a pure syntactical merger of the individual files. The algorithm

requires some interaction with Tamarin, but noticeably simplifies the following analysis. The main idea is that if Tamarin finds an a problem in the merged lemma, then we need to analyze the trace produced by the tool and to modify the lemma. This procedure seems to be systematic for all the examples that we have considered here.

This algorithm is implemented in Python, and works automatically on most combinations from [9, 22]. Only in a handful of cases we need to manually adapt the produced output to obtain a valid lemma that removes all undesirable cases. Note that even in these cases, the manual intervention was only necessary to create the models, the following analysis was then automatic. For more information about this algorithm, see the extended version of this paper [15]. The implementation is also available on line [15].

4 Conclusion

In this paper, we perform an automated analysis of multi-protocols in Tamarin. For this we have used the both protocols studied in [9] using Scyther and the protocol studied in [22] manually. In all cases where attacks were known previously, we also find attacks. However, the tool sometimes finds different attacks than the ones found manually or using Scyther. Moreover, we also find new and unknown attacks, underlining the advantages of an automatic analysis. We also proposed an algorithm to systematically merge two Tamarin files for our analysis.

Our future work is to see how we can integrate our algorithm for automatically merging two Tamarin files into the tools in order to facilitate the life of Tamarin users. Finally our experience also shows us that it might even be possible to propose a similar heuristic to help Tamarin users by automatically generating such helping intermediate lemmas.

References

1. Abadi, M., Needham, R.: Prudent engineering practice for cryptographic protocols. *IEEE Trans. Softw. Eng.* **22**(1), 6–15 (1996)
2. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuellar, J., Drielsma, P.H., Heám, P.C., Kouchnarenko, O., Mantovani, J., Mödersheim, S., von Oheimb, D., Rusinowitch, M., Santiago, J., Turuani, M., Viganò, L., Vigneron, L.: The AVISPA tool for the automated validation of internet security protocols and applications. In: Etessami, K., Rajamani, S.K. (eds.) *CAV 2005*. LNCS, vol. 3576, pp. 281–285. Springer, Heidelberg (2005). <https://doi.org/10.1007/11513988-27>
3. Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: *Proceedings of the 14th IEEE Workshop on Computer Security Foundations, CSFW 2001*, Washington, DC, USA, pp. 82–96. IEEE Computer Society (2001)
4. Burrows, M., Abadi, M., Needham, R.: A logic of authentication. *ACM Trans. Comput. Syst.* **8**(1), 18–36 (1990)
5. Buttyan, L., Staamann, S., Wilhelm, U.: A simple logic for authentication protocol design. In: *11th IEEE Computer Security Foundations Workshop*, pp. 153–162. IEEE Computer Society Press (1998)

6. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067 (2000). <http://eprint.iacr.org/2000/067>
7. Clark, J., Jacob, J.: A survey of authentication protocol literature: version 1.0 (1997)
8. Clark, J.A., Jacob, J.: On the security of recent protocols. *Inf. Process. Lett.* **56**(3), 151–155 (1995)
9. C. Cremers. Feasibility of multi-protocol attacks. In: Proceedings of the First International Conference on Availability, Reliability and Security (ARES), Vienna, Austria, pp. 287–294. IEEE Computer Society (2006)
10. Cremers, C., Mauw, S.: Security properties. In: Operational Semantics and Verification of Security Protocols. ISC, pp. 37–65. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-78636-8_4
11. Cremers, C., Mauw, S., de Vink, E.: Injective synchronisation: an extension of the authentication hierarchy. *Theor. Comput. Sci.* **367**(1), 139–161 (2006)
12. Cremers, C.J.: Unbounded verification, falsification, and characterization of security protocols by pattern refinement. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, CCS 2008, pp. 119–128. ACM, New York (2008)
13. Denning, D.E., Sacco, G.M.: Timestamps in key distribution protocols. *Commun. ACM* **24**(8), 533–536 (1981)
14. Durgin, N.A., Mitchell, J.C., Pavlovic, D.: A compositional logic for proving security properties of protocols. *J. Comput. Secur.* **11**(4), 677–722 (2003)
15. Elliott, B., Dreier, J., Lafourcade, P.: Formal Analysis of Combinations of Secure Protocols (Extended Version). Technical report (2017). <https://hal.archives-ouvertes.fr/hal-01558552v3>
16. Hwang, T., Chen, Y.-H.: On the security of SPLICE/AS - the authentication system in WIDE internet. *Inf. Process. Lett.* **53**(2), 97–101 (1995)
17. Kao, I.-L., Chow, R.: An efficient and secure authentication protocol using uncertified keys. *SIGOPS Oper. Syst. Rev.* **29**(3), 14–21 (1995)
18. Kelsey, J., Schneier, B., Wagner, D.: Protocol interactions and the chosen protocol attack. In: Christianson, B., Crispo, B., Lomas, M., Roe, M. (eds.) Security Protocols 1997. LNCS, vol. 1361, pp. 91–104. Springer, Heidelberg (1998). <https://doi.org/10.1007/BFb0028162>
19. Lowe, G.: An attack on the needham-schroeder public-key authentication protocol. *Inf. Process. Lett.* **56**(3), 131–133 (1995)
20. Lowe, G.: A hierarchy of authentication specification. In: 10th Computer Security Foundations Workshop (CSFW 1997), 10–12 June 1997, Rockport, Massachusetts, USA, pp. 31–44. IEEE Computer Society (1997)
21. Lowe, G.: Towards a completeness result for model checking of security protocols. *J. comput. secur.* **7**(2–3), 89–146 (1999)
22. Mathuria, A., Singh, A.R., Shravan, P.V., Kirtankar, R.: Some new multi-protocol attacks. In: Proceedings of the 15th International Conference on Advanced Computing and Communications, ADCOM 2007, Washington, DC, USA, pp. 465–471. IEEE Computer Society (2007)
23. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN prover for the symbolic analysis of security protocols. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 696–701. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_48
24. Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. *Commun. ACM* **21**(12), 993–999 (1978)

25. Needham, R.M., Schroeder, M.D.: Authentication revisited. *SIGOPS Oper. Syst. Rev.* **21**(1), 7 (1987)
26. Perrig, A., Song, D.: Looking for diamonds in the desert - extending automatic protocol generation to three-party authentication and key agreement protocols. In: *Proceedings of the 13th IEEE Workshop on Computer Security Foundations, CSFW 2000, Washington, DC, USA*, pp. 64–76. IEEE Computer Society (2000)
27. Song, D.X., Berezin, S., Perrig, A.: Athena: a novel approach to efficient automatic security protocol analysis. *J. Comput. Secur.* **9**(1–2), 47–74 (2001)
28. Woo, T.Y.C., Lam, S.S.: A lesson on authentication protocol design. *SIGOPS Oper. Syst. Rev.* **28**(3), 24–37 (1994)
29. Yamaguchi, S., Okayama, K., Miyahara, H.: The design and implementation of an authentication system for the wide area distributed environment. *IEICE Trans. Inf. Syst.* **74**(11), 3902–3909 (1991)
30. Zhou, H., Foley, S.N.: Fast automatic synthesis of security protocols using backward search. In: *Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering, FMSE 2003*, pp. 1–10. ACM, New York (2003)



Formal Analysis of the FIDO 1.x Protocol

Olivier Pereira, Florentin Rochet^(✉), and Cyrille Wiedling

UCL Crypto Group, Louvain-la-Neuve, Belgium
florentin.rochet@uclouvain.be

Abstract. This paper presents a formal analysis of FIDO, a protocol developed by the FIDO Alliance project, and which aims to provide either a passwordless experience or an extra security layer for user authentication over the Internet. We model the protocol using the applied pi-calculus and run our analysis using ProVerif. Our analysis shows that ignoring some optional steps of the standard could lead to the implementation of a flawed authentication process. On the contrary, we prove that these steps are sufficient to ensure the expected security properties.

1 Introduction

Authentication is a process arguably insecure under many forms. The most common one, password, has been criticized for years and many works have tried to propose a workable replacement [BHVOS12]. It is now widely accepted that relying on passwords only is insecure for sensitive applications.

The FIDO alliance [All] aims at establishing a standard for passwordless experience and second factor authentication. FIDO has received support from many companies around the world and has been integrated by various services such as banks (e.g. Bank of America), e-mails (e.g. Gmail), social networks (e.g. Facebook), etc. As those lines are written, the FIDO alliance claims to have brought FIDO capabilities to more than 3 billions of user accounts worldwide.

The protocol is modular and allows companies to build components in an inter-operable way. Their goal is to avoid the use of a server-side shared secret, by leveraging cryptographic capabilities of a hardware token that the user must own in order to obtain a successful authentication. The protocol works in a two-phases scenario. In the first phase, the user must register his token on the desired service. The registration process corresponds to the creation of a private/public key pair for which the public key is transmitted to the services and all the private keys are held in the hardware device. This first phase is performed for each of the services that the user wants to access to, but it must be performed once. The second phase is the authentication itself: it essentially consists in signing random challenges generated by the service, after approval by the user.

In this paper, we propose a formal analysis of the authentication methodology provided by the specifications of the U_2F standard, the “second factor experience”, in which a FIDO token is used as a complement to a password on web services. These specifications provide the description of the protocol and

some optional features. We focused our analysis on one of these features and, using the tool ProVerif [Bla01], we prove that without it, there exists an attack where a web-server could impersonate a user to a third party service, under the assumption of the use of weak passwords, which is precisely the issue that the FIDO U_2F protocol is expected to solve. We also provide a local test-bed attack scenario to illustrate our finding. On the positive side, we prove that, if this feature is properly implemented, then the expected security properties hold.

Roadmap. Sections 2 and 3 present the U_2F protocol and the applied pi-calculus used for our analysis. In Sect. 4, we define a model of the protocol and a definition of the authentication property. In Sect. 5, we present and analyze our results. Section 6 gives the related work.

2 The FIDO Protocol

The FIDO protocol aims to authenticate a user to a server, using a token (e.g. smartcard, USB token, etc.), in such a way that is not possible to impersonate a user without being in possession of his token, even if the username and the password of that user have been compromised. The protocol runs between a user, a Token, a FIDO Client embedded in the user’s web browser, and a server, after the establishment of a secure TLS between the last two entities. As described above, it is composed of two main phases: the registration and the authentication.

2.1 Registration Phase

The registration step is used to link a token to the account of a user. Figure 1 offers a high-level view of its behavior. (A full version can be found at [All].) The

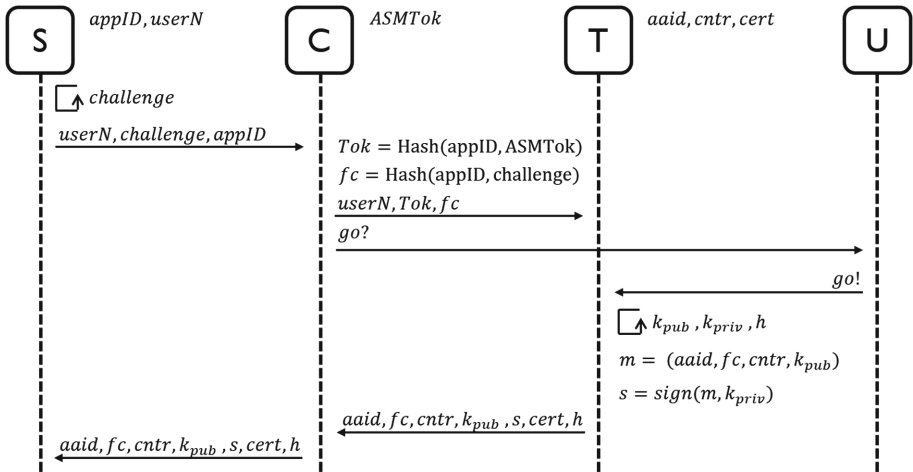


Fig. 1. Simplified registration Step. S: Server, C: Client, T: Token, U: User

first thing that happens when the FIDO Client is installed on a computer is the generation of a random *ASMTok*, which is used to provide a link between the key handle that will be created on the Token and the Client. The Token possesses different items: its unique identifier, *aaid*, a counter *cntr*, which is incremented each time it performs a signature for an authentication; and a certificate *cert*, which is delivered by the manufacturer of the Token, corresponding to the master key, which is used to sign the generated key pairs.

The registration then proceeds as follows: for a given User, identified by a username, *userN*, the Server generates a *challenge* and sends it to the Client, along with the username, and the *appID* identifying the server (e.g. its url). Then, the Client computes two values, *Tok* and *fc*, the first one links the *appID* to the secret token *ASMTok*, the second one links the *appID* to the *challenge*. Those two values are given to the Token. The latter waits for the User the permission to perform the registration step (e.g. the Client displays a message asking for the User to push a button on the Token). Once the User approves, the Token generates a key pair and stores the private key, with *Tok*, in a handle *h*. Using its identifier *aaid* and a counter *cntr*, it creates a message and signs it before sending it back to the Client, with the identifier of the handle *h* and the certificate *cert*. The Client forwards everything to the Server, which checks the certificate and the signature before adding the public key to the User's information.

2.2 Authentication Phase

The authentication step (see Fig. 2) is quite similar to the registration one. Once the User has provided his username and password, if the Server has a Token

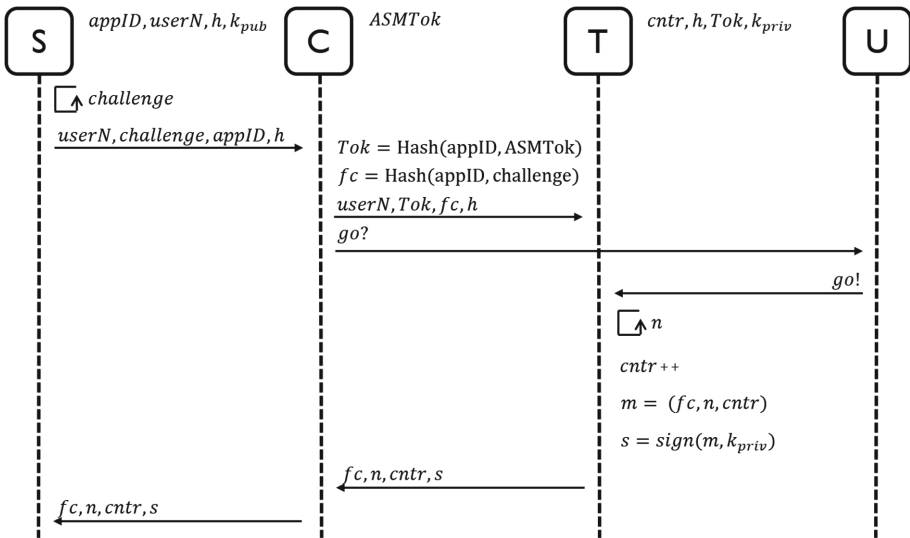


Fig. 2. Simplified authentication step.

registered to that account, it will generate a challenge and send it to the Client, along with the handle h it got from the registration step. As during the registration step, the Client will create Tok and fc , and send it to the Token before asking to the User to approve the authentication (e.g. by pushing the Token's button). Then, the Token checks if it possesses an entry h and whether its content match the Tok given by the Client. If it is, then the Token generates a nonce n , increments its counter $cntr$ and uses the secret key stored in the entry h to sign a message. This message is passed to the Client, along with n , fc and $cntr$, which transmits them to the Server. Using the public key it has registered with the account, the Server will check whether the signature is correct or not, and provide access in the former case.

3 Applied Pi-Calculus

This section briefly presents the notations of the applied pi-calculus, a process algebra introduced by Abadi and Fournet [AF01], often used to model protocols and security properties.

3.1 Terms

Messages are represented by *terms* built upon an infinite set of *names* \mathcal{N} (for communication channels or atomic data), a set of *variables* \mathcal{X} and a *signature* Σ consisting of a finite set of *function symbols* (to represent cryptographic primitives). A function symbol f is assumed to be given with its arity $\text{ar}(f)$. Then, the set of terms $T(\Sigma, \mathcal{X}, \mathcal{N})$ is formally defined by the following grammar:

$$\begin{array}{ll}
 t, t_1, t_2, \dots ::= & \\
 x & x \in \mathcal{X} \\
 n & n \in \mathcal{N} \\
 f(t_1, \dots, t_n) & f \in \Sigma, n = \text{ar}(f)
 \end{array}$$

In order to represent the properties of the primitives, the signature Σ is equipped with an *equational theory* E that is a set of equations which hold on terms built from the signature. We denote by $=_E$ the smallest equivalence relation induced by E , closed under application of function symbols, substitutions of terms for variables and bijective renaming of names. We write $M =_E N$ when the equation $M = N$ holds in the theory E .

3.2 Processes

Processes and *extended processes* are defined in Fig. 3. The process 0 represents the null process that does nothing. $P \mid Q$ denotes the parallel composition of P with Q while $!P$ denotes the unbounded replication of P (i.e. the unbounded parallel composition of P with itself). $\nu n.P$ creates a fresh name n and then behaves like P . The process $\text{if } \phi \text{ then } P \text{ else } Q$ behaves like P if ϕ holds and like

$P, Q, R ::=$	(plain) processes
0	null process
$P \mid Q$	parallel composition
$!P$	replication
$\nu n.P$	name restriction
if ϕ then P else Q	conditional
$u(x).P$	message input
$\bar{u}(M).P$	message output
$A, B, C ::=$	extended processes
P	plain process
$A \mid B$	parallel composition
$\nu n.A$	name restriction
$\nu x.A$	variable restriction
$\{^M/x\}$	active substitution

Fig. 3. Syntax for processes

Q otherwise. $u(x).P$ inputs some message in the variable x on channel u and then behaves like P while $\bar{u}(M).P$ outputs M on channel u and then behaves like P . We write $\nu \bar{u}$ for the (possibly empty) series of pairwise-distinct binders $\nu u_1 \dots \nu u_n$. The active substitution $\{^M/x\}$ can replace the variable x for the term M in every process it comes into contact with and this behavior can be controlled by restriction, in particular, the process $\nu x (\{^M/x\} \mid P)$ corresponds exactly to let $x = M$ in P .

As in [CS13], we slightly extend the applied pi-calculus by letting conditional branches now depend on formulae defined by the following grammar:

$$\phi, \psi ::= M = N \mid M \neq N \mid \phi \wedge \psi$$

If M and N are ground, we define $\llbracket M = N \rrbracket$ to be **true** if $M =_E N$ and **false** otherwise. The semantics of $\llbracket \cdot \rrbracket$ is then extended to formulae as expected.

The *scope* of names and variables is delimited by binders $u(x)$ and νu . Sets of bound names, bound variables, free names and free variables are respectively written $\text{bn}(A)$, $\text{bv}(A)$, $\text{fn}(A)$ and $\text{fv}(A)$. Occasionally, we write $\text{fn}(M)$ (resp. $\text{fv}(M)$) for the set of names (resp. variables) which appear in term M . An extended process is *closed* if all its variables are either bound or defined by an active substitution. A *context* $C[_]$ is an extended process with a hole instead of an extended process. We obtain $C[A]$ as the result of filling $C[_]$'s hole with the extended process A . An *evaluation context* is a context whose hole is not in the scope of a replication, a conditional, an input or an output. A context $C[_]$ closes A when $C[A]$ is closed. A *frame* is an extended process built up from the null process 0 and active substitutions composed by parallel composition and restriction. The *domain* of a frame φ , denoted $\text{dom}(\varphi)$ is the set of variables for which φ contains an active substitution $\{^M/x\}$ such that x is not under restriction. Every extended process A can be mapped to a frame $\varphi(A)$ by replacing every plain process in A with 0.

3.3 Operational Semantics

The operational semantics of processes in the applied pi-calculus is defined by three relations: *structural equivalence* (\equiv), *internal reduction* (\rightarrow) and *labelled reduction* ($\xrightarrow{\alpha}$).

Structural equivalence is defined in Fig. 4. It is closed by α -conversion of both bound names and bound variables, and closed under application of evaluation contexts. Structural equivalence corresponds to some structural rewriting that does not change the semantics of a process. The *internal reductions* and *labelled reductions* are defined in Fig. 5. They are closed under structural equivalence

PAR-0	$A \equiv A \mid 0$	
PAR-A	$A \mid (B \mid C) \equiv (A \mid B) \mid C$	
PAR-C	$A \mid B \equiv B \mid A$	
REPL	$!P \equiv P \mid !P$	
NEW-0	$\nu n.0 \equiv 0$	
NEW-C	$\nu u.\nu w.A \equiv \nu w.\nu u.A$	
NEW-PAR	$A \mid \nu u.B \equiv \nu u.(A \mid B)$	if $u \notin \text{fv}(A) \cup \text{fn}(A)$
ALIAS	$\nu x.\{^M/x\} \equiv 0$	
SUBST	$\{^M/x\} \mid A \equiv \{^M/x\} \mid A\{^M/x\}$	
REWRITE	$\{^M/x\} \equiv \{^N/x\}$	if $M =_E N$

Fig. 4. Structural equivalence.

(COMM)	$\bar{c}(M).P \mid c(x).Q \rightarrow P \mid Q\{^M/x\}$
(THEN)	if ϕ then P else $Q \rightarrow P$ if $\llbracket \phi \rrbracket = \text{true}$
(ELSE)	if ϕ then P else $Q \rightarrow Q$ otherwise

(IN)	$c(x).P \xrightarrow{c(M)} P\{^M/x\}$
(OUT-ATOM)	$\bar{c}(u).P \xrightarrow{\bar{c}(u)} P$
(OPEN-ATOM)	$\frac{A \xrightarrow{\bar{c}(u)} A' \quad u \neq c}{\nu u.A \xrightarrow{\nu u.\bar{c}(u)} A'}$
(SCOPE)	$\frac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$
(PAR)	$\frac{A \xrightarrow{\alpha} A' \quad \text{bv}(\alpha) \cap \text{fv}(B) = \text{bn}(\alpha) \cap \text{fn}(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$
(STRUCT)	$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$

where α is a *label* of the form $c(M)$, $\bar{c}(u)$, or $\nu u.\bar{c}(u)$ such that u is either a channel name or a variable of base type.

Fig. 5. Semantics for processes

and application of evaluation contexts. Internal reductions represent evaluation of condition and internal communication between processes. Labelled reductions represent communications with the environment.

4 Modeling FIDO

In this section, we present our model, in applied pi-calculus, of the FIDO protocol, and the definition of the authentication property it is aimed to achieve.

4.1 Settings

We focus our analysis on the FIDO protocol itself, and as such, our model starts after the establishment of the TLS channel between the Client and the Server. Thus, we consider a secure channel between these two. Although this channel may be out of reach from an attacker in a fully honest setting, we will consider different corruption scenarios that will grant the attacker access to it and therefore will not limit our analysis.

We consider the following entities:

- **User**: represents the person that is willing to connect to the Server through the Client, using the Token.
- **Token**: represents the device that stores and uses the different keys used for authentication to the Server.
- **Client**: represents the embedded Client in the browser of the User, used to established the connection to the Server.
- **Server**: represents the service the User wants to connect to, using the Client.

Let us define notations for the communication channels (see Fig. 6):

- c_I is a secure channel between the Client and the Server. It models the TLS-secured channel between the two entities.
- c_T is the secure channel between the Client and the Token. Basically, it models the USB (or NFC/Bluetooth) connection.
- c_E is the channel between the Client and the User, i.e. the screen of the computer where the Client may display information to the User.
- c_F is the channel between the User and the Token. It is just modeling the possibility, for the User, to push the Token's button to allow authentication.

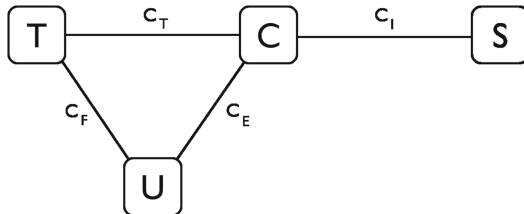


Fig. 6. Channels between the different players of the FIDO protocol.

4.2 Threat Model

In our model, like in numerous formal analyses, we consider an attacker who has control over the network. Therefore, he is able to intercept, forge, send messages. Restrictions are based on the nature of channels. An attacker can listen to, but not send messages over authenticated channels. Public channels are, by definition public, and everyone can listen to or send messages over them.

During this analysis, we consider two main scenarios:

- Everyone is honest. It is a classical scenario where the attacker only has access to the network (and non-secure channels).
- Either the Server or the Client is dishonest.

Note that we cannot corrupt both the Client and the Server at the same time, otherwise it would be impossible to guarantee anything. We also do not consider cases where the User may be corrupted, because the User is just an entity trying to connect to a service – which is meaningless to corrupt. The Token stores keys and an attacker may want to try to break it, but if such a Token is corrupted we can not guarantee any security on the second level authentication since the attacker would have access to the keys used for authentication.

4.3 Authentication Property

The FIDO protocol aims to provide authentication to the user. Once registered and linked to an account, the token is designed to be necessary in order to perform an authentication on the server. The user should provide login and password, as usual, but the server will also require a signature coming from the token, which is given if the user pushes a button on the token itself. In our study, we explore this authentication property, in order to see if there is any possibility for an attacker to impersonate a user on a server.

Definition 1. *Let us consider different predicates:*

- $\text{LoggedIn}(appID, username, pk)$ is issued by the server, identified by $appID$, when he has accepted (by finishing the protocol) a connection with the user, identified by $username$, and referenced under the public key pk .
- $\text{ExpConnection}(appID, username, s)$ is issued by the user, secretly identified by s , asserting that he expects a connection to $appID$, with his $username$.
- $\text{PushButton}(s)$ is issued by the user when he pushes the button on the token.
- $\text{TokenSign}(sk)$ is issued by the token when it signs using the secret key sk .

Using these predicates, we define the authentication property:

$$\begin{aligned} \forall appID, username, pk, \\ \text{LoggedIn}(appID, username, pk) \implies \exists s, sk \\ \text{TokenSign}(sk) \wedge (pk = \text{pk}(sk)) \wedge \\ \text{PushButton}(s) \wedge \\ \text{ExpConnection}(appID, username, s) \end{aligned}$$

This property reflects that each time a server accepts a connection, then a user first expected to initiate a connection to this server and pushed on the button of the token, which generated the expected signature.

4.4 Modeling the Protocol

Since we are focusing on the authentication property, we model the authentication part of the protocol, assuming that all the needed registrations have been already performed, with no interference of an attacker.

Server. After a registration, the Server knows, for each username registered: the public key pk , the handle h pointing to the corresponding secret key inside the Token, and the status of the counter $cntr$. During the authentication phase, when the User has provided his username and password, the Server will generate a new challenge n_s and send its $AppID$, h and n to the Client. Then, it will wait for a signature s before checking it. He also checks whether the counter sent by the token is larger than the one stored. If the verifications succeed, then the Servers accepts the login.

$$\begin{aligned}
 S(appID, username, pk, h, cntr) = & \\
 & \nu n_s; \\
 & \overline{c_I} \langle (appID, n_s, h) \rangle; \\
 & c_I \langle (x) \rangle; \\
 & \text{let } (x_1, x_2, x_3, x_4) = x \text{ in} \\
 & \text{if } \text{checksign}((x_1, x_2, x_3), x_4, pk) = \text{ok} \wedge (x_3 > cntr) \text{ then} \\
 & \text{LoggedIn}(appID, username, pk)
 \end{aligned}$$

Client. The Client waits for the inputs of the Server, then verifies that the $appID$ matches the one it is expecting (a). If the condition statement succeeds, then the Client computes hashes that will be passed to the Token for the signature, using its secret value t_c . The Client then prompts a message to the User, asking for a confirmation to the Token, and waits for a response from the latter, before passing the signature to the Server. We also consider another Client model (C_c) where the IF-condition is missing, since this step is optional in the specification (see [All16b, Sect. 5.2.1] and our discussion in Sect. 5.2).

$$\begin{array}{ll}
 C(a_c, t_c) = & C_c(t_c) = \\
 c_I \langle (x) \rangle; & c_I \langle (x) \rangle; \\
 \text{let } (x_1, x_2, x_3) = x \text{ in} & \text{let } (x_1, x_2, x_3) = x \text{ in} \\
 \text{if } (x_1 = a) \text{ then} & \text{let } KToken = \text{hash}((x_1, t_c)) \text{ in} \\
 \overline{c_E} \langle go? \rangle; & \text{let } fc = \text{hash}((x_1, x_2)) \text{ in} \\
 \text{let } KToken = \text{hash}((x_1, t_c)) \text{ in} & \overline{c_E} \langle go? \rangle; \\
 \text{let } fc = \text{hash}((x_1, x_2)) \text{ in} & \overline{c_T} \langle (x_3, KToken, fc) \rangle; \\
 \overline{c_T} \langle (x_3, KToken, fc) \rangle; & c_T \langle y \rangle; \\
 c_T \langle y \rangle; & \overline{c_I} \langle y \rangle; \\
 \overline{c_I} \langle y \rangle; &
 \end{array}$$

User. The User does not do much except for sending a go-message to the Token. We introduce a secret s_h , which captures the assumption that the button can only be pressed by the User: no one can push the button on the Token, except the User. During authentication, the User waits for the Client's prompt, and then, pushes the button of the Token to allow the authentication.

$$\begin{aligned}
 U(\text{ExpAppID}, \text{username}, s_h) = & \\
 & \text{ExpConnection}(\text{appID}, \text{username}, s_h); \\
 & c_E((x)); \\
 & \overline{c_F}(\langle \text{go!} \rangle); \\
 & \text{PushButton}(s_h)
 \end{aligned}$$

Token. During the registration step, the Token stored, under a handle h , the secret key sk , and a token $KToken$ generated by the Client. The Token itself is identified by an $aaid$ and updates a counter, $cntr$, which is used to avoid replay attacks. During the authentication step, the Token waits for inputs from the Client and the confirmation of the User (e.g. by pushing a button). It also checks whether the handle and the token are valid and provides a signature using the corresponding secret key if everything is in order.

$$\begin{aligned}
 T(\text{aaid}, \text{cntr}, h, KToken, sk) = & \\
 & c_T((x)); \\
 & \text{let } (x_1, x_2, x_3) = x \text{ in} \\
 & c_F((x)); \\
 & \text{if } (x_1, x_2) = (h, KToken) \text{ then} \\
 & \quad \nu n_t; \\
 & \quad \text{let } s_t = \text{sign}((x_3, n_t, \text{cntr}), sk) \text{ in} \\
 & \quad \text{TokenSign}(sk); \\
 & \overline{c_T}(\langle x_3, n_t, \text{cntr}, s_t \rangle)
 \end{aligned}$$

FIDO Protocol. The authentication part can now be modeled by placing all processes in parallel, which is written as follows:

$$P_a = \nu \tilde{n}. a.[S(a, p_u, pk_u, h, c) \mid C(a, s_c) \mid T(a, c, h, tok, k_u) \mid U(a, p_u, s_u) \mid \Gamma].$$

where $\tilde{n} = a, k_u, s_u, p_u, h, c, s_c, pk_u, tok$ are bound names which represent the different items created during the registration phase that are used as arguments for our processes, and $\Gamma = \{\text{pk}(k_u)/pk_u, \text{hash}((a, s_c))/tok\}$ is a frame describing the content of tok and pk_u .

We also model the case where the Client model does not verify the $appID$:

$$P_c = \nu \tilde{n}. a.[S(a, p_u, pk_u, h, c) \mid C_c(s_c) \mid T(a, c, h, tok, k_u) \mid U(a, p_u, s_u) \mid \Gamma].$$

5 ProVerif Results

In this section, we present our results, obtained using the ProVerif tool, studying the authentication property of the FIDO protocol in various scenarios.

5.1 Client Model with AppID Verification

To analyze the FIDO protocol with respect to the authentication property, we choose to use the ProVerif tool, developed by Blanchet [Bla01], which is an automated verifier that can achieve to prove injection properties (and more), for protocols. It has been used for various examples (like [KR05],[KT08]) and also has some limitations (e.g. [CW12]), especially when protocols get too complicated (e.g., involving several cryptographic primitives), but the FIDO protocol only involves signatures and hashes, and remains within ProVerif’s scope. Our ProVerif files can be found at [WRP].

Results. Our results, obtained using ProVerif, are compiled in Table 1. We show that even if we corrupt the Server, or the Client (but never both of them), there is no possible attack. The server-in-the-middle case is more interesting: we suppose that a user registered to two different servers and does not know that one of them is corrupted. We also consider that one of them is corrupted and knows the login and password of the user for the honest one. (e.g., it could be possible that the user uses the same couple login/password for the two servers.)

5.2 Client Model Without AppID Verification

In the model above, we check if the *appID* provided by the server matches the origin of the request. The *appID* verification is correctly implemented in the Chrome FIDO Client but nothing prevents an other implementation to miss this important step. Indeed, this is an ambiguous step in the FIDO specifications, where nothing is requested in the UAF protocol description. In particular, we found the following recommendation: “The FIDO client should perform the following steps: - Verify the application identity of the caller.” [All16b].

It is clearly stated as a recommendation but, for reasons that we will detail later, we believe that this should be an assertion (MUST). The optional character of this recommendation is indirectly confirmed in other official documents. For instance, the overview [All] reads: “Say a user has correctly registered a U2F device with an origin and later, a MITM on a different origin tries to intermediate the authentication. In this case, the user’s U2F device won’t even respond, since the MITM’s (different) origin name will not match the Key Handle that the MITM is relaying from the actual origin.”

The above statement ignores a possible *appID* verification since if the device does not respond, then, it must have seen the request despite the different origin. Therefore, the verification is missing, otherwise the data would not have been sent to the device. Moreover, since the token is never given the actual origin of a request, it cannot perform such a verification.

Results. As we see in Table 1, ProVerif outputs that the authentication property does not hold anymore. In practice, it corresponds to the following attack: when the User authenticates on Server A (Server ITM), the Server initiates an authentication towards Server B and forwards the challenge to the victim. Without the

Table 1. ProVerif results on our authentication property.

Corrupted Players	Without appID Verification	With appID Verification
None	✓	✓
Server	✓	✓
Client	✓	✓
Server + Client	×	×
None + Server ITM	×	✓

appID verification, the User provides a valid Server B authentication credential that Server A can use to impersonate the User on Server B.

In practice, even if the cryptographic primitives are correctly implemented, forgetting to verify the *appID* is enough to impersonate a User in a plausible attack scenario. Moreover, if FIDO is used as a second factor authentication, the Server ITM might already have access to the User password and login through its own database, since the victim might use the same password across multiple services, which is a scenario against which the Token should offer a protection.

As a result, we believe that our attack scenario violates at least the following security goal stated in the specification [All16a]:

- **SG-3** (credential disclosure resilience), in the sense that a loss of credentials is sufficient to run the man-in-the-middle attack described above and successfully bypass the two levels of authentication.

Corresponding Attack on the Real Token. We confirm the ProVerif’s results in practice by providing a working java web server implementation [Roca] that can run either as a honest server or as a corrupted server. We tested our attack with Chrome 40 [Goo] packaged for Debian and with our modified FIDO Client [Rocb]. We had to modify the FIDO Client provided by Google to remove the *appID* verification that was hopefully correctly integrated. A tutorial is provided in Appendix A to reproduce the attack.

Lesson Learned. FIDO is secure if we assume both that the implementer understood the importance of the *appID* verification despite its optional character in the specification. Moreover, we have also to assume that there is no possibility to fool the origin verification in the browser. We recommend the specification to enforce the *appID* verification: among other changes, the *appID* verification must not be written as a recommendation (using a should) but instead as an assertion (using a must). Also, these results suggest that current FIDO existing Clients should be audited to check if the *appID* verification exists and does not suffer from any weakness. Very few such clients are freely available, though.

6 Related Work

In 1996, Lowe [Low96] raised the interest in analysing authentication protocol when he presented a MITM attack against Needham and Schroeder public key protocol [NS78]. Later, the progress in web technologies drove a consortium of internet companies to unite and design a user-friendly standard identity management and an authentication protocol. This initiative was called the Liberty Alliance Project. A research group, led by Pfitzmann, laid the basis for formal analysis of such web-based identity management protocol [GPS05, PW03]. They did not designed automated formal analysis tools but they discussed formal descriptions of those protocols. Eventually, the Liberty Alliance produced SAML 2.0, an open-standard data format to exchange information between parties. These specifications lead to SSO protocols (Single-Sign-On) used for cross-authentication. The SSO protocol implemented by Google got broken with automated formal analysis [ACC+08], where researcher found how to impersonate a user when acting as a dishonest service, at another service provider. For years now, browsers have gained cryptographic capabilities and many authentication protocols appeared and gained in complexity. The need for mechanical analysis got more evident, since modelling the protocol became the time-consuming task and proofs were provided by the tool. In this line of work, Bortolozzo et al. [BCFS10] designed a tool to audit PKCS#11 tokens and found weaknesses on many of them. Some other tools have been designed to automate the process of proving security properties, such as the one we use in this paper: ProVerif [Bla01] but also APTE [Che14] and aKiss [CCK12]. Other hardware tokens, such as Yubikeys doing one-time passwords, have been analysed with other automated tools [KS12, KK16]. They are mostly chosen depending on the property we want to prove. Some are easier to use in some circumstances.

7 Conclusion

We modeled the authentication phase of the FIDO protocol in Applied Pi Calculus. We considered two different client models, based on the execution or not of a verification step by the client, step that is left optional in the specification.

Our ProVerif analysis shows that, when the verification is performed, the expected authentication properties are satisfied. However, when it is ignored, a man-in-the-middle attack becomes possible, assuming compromised (or reused) credentials, which definitely falls within the scope of the attacks that the use of a FIDO token is expected to prevent.

As a result, we recommend making this verification step mandatory, and that the authors of the 49 certified client implementations listed on the FIDO Alliance website check whether this step is actually performed.

Acknowledgement. We would like to thank the anonymous reviewers for their valuable feedback. This work was partially supported by the Innoviris C-Cure project and the Region Wallonne TrueDev project.

A Attack on a Real FIDO Authentication

We detail the steps to test the attack from Sect. 5.2 in a local test-bed.

- Clone repositories [Roca,Rocb] and download chrome [Goo].
- From chrome tab extensions, activate the developer mode and load the unpacked u2f-chrome-extension.
- Inside java-u2flib-server, do *mvn clean install* then cd inside u2f-server-demo and configure two .yml files, one for the compromised server and one for the honest server. Use the available model, you just need to provide a different port number.
- Run:
 - `java -jar target/u2flib-server-demo.jar server [your_consigDishonest_file.yml] localhost [port_honest_server] https://localhost:[port_dishonest_server] &`
 - `java -jar target/u2flib-server-demo.jar server [your_consigHonest_file.yml] http://localhost:[port_honest_server] &`
- Use chrome and your FIDO-compliant authenticator to register in both services then try to authenticate.

References

- [ACC+08] Armando, A., Carbone, R., Compagna, L., Cuellar, J., Tobarra, L.: Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for Google apps. In: Proceedings of the 6th ACM workshop on Formal Methods in Security Engineering, pp. 1–10. ACM (2008)
- [AF01] Abadi, M., Fournet, C.: Mobile values, new names, and secure communication. In: 28th ACM Symposium on Principles of Programming Languages (POPL) (2001)
- [All] FIDO Alliance: FIDO Documentation. <https://fidoalliance.org/specifications/download/>
- [All16a] FIDO Alliance: Fido security reference, September 2016. <https://fidoalliance.org/specs/fido-u2f-v1.1-id-20160915/fido-security-ref-v1.1-id-20160915.html>
- [All16b] FIDO Alliance: FIDO U2F JavaScript API, September 2016. <https://fidoalliance.org/specs/fido-u2f-v1.1-id-20160915/fido-u2f-javascript-api-v1.1-id-20160915.html>
- [BCFS10] Bortolozzo, M., Centenaro, M., Focardi, R., Steel, G.: Attacking and fixing PKCS#11 security tokens. In: ACM Conference on Computer and Communications Security (CCS) (2010)
- [BHVOS12] Bonneau, J., Herley, C., Van Oorschot, P.C., Stajano, F.: The quest to replace passwords: a framework for comparative evaluation of web authentication schemes. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 553–567. IEEE (2012)
- [Bla01] Blanchet, B.: An efficient cryptographic protocol verifier based on prolog rules. In: 14th IEEE Computer Security Foundations Workshop (CSFW) (2001)

- [CCK12] Chadha, R., Ciobăcă, Ș., Kremer, S.: Automated verification of equivalence properties of cryptographic protocols. In: Seidl, H. (ed.) ESOP 2012. LNCS, vol. 7211, pp. 108–127. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28869-2_6
- [Che14] Cheval, V.: APTE: an algorithm for proving trace equivalence. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 587–592. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_50
- [CS13] Cortier, V., Smyth, B.: Attacking and fixing Helios: an analysis of ballot secrecy. *J. Comput. Secur.* **21**(1), 89–148 (2013)
- [CW12] Cortier, V., Wiedling, C.: A formal analysis of the Norwegian E-voting protocol. In: Degano, P., Guttman, J.D. (eds.) POST 2012. LNCS, vol. 7215, pp. 109–128. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28641-4_7
- [Goo] Google: Chrome browser download. <http://google-chrome.en.uptodown.com/ubuntu/old>. Accessed 13 Jan 2016
- [GPS05] Groß, T., Pfitzmann, B., Sadeghi, A.-R.: Browser model for security analysis of browser-based protocols. In: di Vimercati, S.C., Syverson, P., Gollmann, D. (eds.) ESORICS 2005. LNCS, vol. 3679, pp. 489–508. Springer, Heidelberg (2005). https://doi.org/10.1007/11555827_28
- [KK16] Kremer, S., Künnemann, R.: Automated analysis of security protocols with global state. *J. Comput. Secur.* **24**(5), 583–616 (2016)
- [KR05] Kremer, S., Ryan, M.: Analysis of an electronic voting protocol in the applied Pi calculus. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 186–200. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31987-0_14
- [KS12] Künnemann, R., Steel, G.: YubiSecure? Formal security analysis results for the Yubikey and YubiHSM. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 257–272. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38004-4_17
- [KT08] Küsters, R., Truderung, T.: Reducing protocol analysis with XOR to the XOR-free Case in the Horn Theory based approach. *CoRR* (2008)
- [Low96] Lowe, G.: Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In: Margaria, T., Steffen, B. (eds.) TACAS 1996. LNCS, vol. 1055, pp. 147–166. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61042-1_43
- [NS78] Needham, R.M., Schroeder, M.D.: Using encryption for authentication in large networks of computers. *Commun. ACM* **21**(12), 993–999 (1978)
- [PW03] Pfitzmann, B., Waidner, M.: Federated identity-management protocols. In: Christianson, B., Crispo, B., Malcolm, J.A., Roe, M. (eds.) Security Protocols 2003. LNCS, vol. 3364, pp. 153–174. Springer, Heidelberg (2005). https://doi.org/10.1007/11542322_20
- [Roca] Rochet, F.: Fido compliant library and Java web application example. <https://github.com/frochet/java-u2f-lib-server>. Accessed 13 Jan 2016
- [Rocb] Rochet, F.: Modified fido client as a chrome extension. <https://github.com/frochet/u2f-ref-code>. Accessed 13 Jan 2016
- [WRP] Wiedling, C., Rochet, F., Pereira, O.: Proverif implementation of the fido protocol. <https://git-crypto.elen.ucl.ac.be/frochet/fido-proverif>



A Roadmap for High Assurance Cryptography

Harry Halpin^(✉)

Inria, 2 rue Simone Iff, 75012 Paris, France
harry.halpin@inria.fr

Abstract. Although an active area of research for years, formal verification has still not yet reached widespread deployment. We outline the steps needed to move from low-assurance cryptography, as given by libraries such as OpenSSL, to high assurance cryptography in deployment. In detail, we outline the need for a suite of high-assurance cryptographic software with per-microarchitecture optimizations that maintain competitive speeds with existing hand-optimized assembly and the bundling of these cryptographic primitives in a new API that prevents common developer mistakes. A new unified API with both formally verified primitives and an easy-to-use interface is needed to replace OpenSSL in future security-critical applications.

Keywords: High assurance cryptography · Formal verification Primitives · Security API

1 Introduction

Our increasingly digital society critically relies on the security of software systems, ranging from small IoT devices, through personal computers and smartphones to cars and software in critical infrastructure. For each of those systems we *trust* that they perform certain important tasks, but do not show malicious or unpredictable behavior, even when under attack. Typically, the recommended approach to building such trustworthy systems is the following:

1. first defining clear security goals;
2. then identifying the so-called *trusted code base* (TCB) i.e., the part of the software system that is critical to achieving these goals;
3. isolating the TCB from the rest of the code, and implementing well-defined interfaces between the TCB and the rest of the code; and
4. assuring that the code in the TCB (including the interfaces) achieves the security goals.

Unfortunately, almost none of the above-listed *trusted* systems are systematically built according to this recipe; they are historically grown, without a clear separation between TCB and “non-critical” code; often even without a clear definition of security goals, and essentially everywhere without high-assurance software in the TCB. As a consequence, we are realizing that our society has *trust* in *untrustworthy* low-assurance software.

2 The Role of Cryptography

Cryptography takes a special role in building trustworthy software. While most other parts of typical TCBs—e.g. operating-system kernels, hypervisors, or critical drivers—implement functionality that is not primarily aiming at security, cryptographic software’s sole purpose is to achieve security goals that would not be achievable without cryptography. This inherently places cryptographic code inside the TCB and thus *always* requires high assurance of correctness and security of cryptographic software, including interfaces (i.e. APIs) that cannot be misused to violate the security goals.

Although there has been much academic work on the theoretical security of cryptographic protocols, security proofs constructed by theoreticians rely critically on security and functionality assumptions for underlying primitives, i.e., they implicitly require the existence of interfaces to implementations of those primitives that do not violate the assumptions in the proofs. Indeed, if these mathematical assumptions do not match what is offered by real-world implementations, then the proofs do not apply. When this is the case, there are no security guarantees for the software and catastrophe ensues.

Unfortunately, this happens in practice. Despite the high confidence that society has in cryptography to maintain the security and privacy of their transactions, every year we see devastating attacks against widely deployed cryptographic software. Most of those attacks do not break the cryptographic protection in a mathematical sense, but instead exploit weaknesses in how these primitives are used or implemented. Weaknesses in cryptographic software include mistakes in the implementation of cryptographic primitives, and hard-to-detect bugs in the underlying arithmetic, programming interfaces (APIs) that enable (or even encourage) wrong use, subtle (and sometimes less subtle) flaws in the protocol layer, and side-channel vulnerabilities, that allow an attacker to obtain extra information about secret data through, for example, timing.

The most prominent example of an exploitable bug in an implementation was probably the weakness enabling the Heartbleed attack [13], which allowed a remote attacker to read memory content that in many cases contained secret data. Yet even more common than bugs in cryptographic primitive implementations are the incorrect use of these primitives by developers: A recurring problem is the unjustified trust placed by programmers on API developers to provide good random generation routines and to preconfigure the various cryptographic components with secure parameters by default, as seen in numerous examples of security problems in Android applications due to the incorrect usage of cryptography APIs are given in [11] that range from the usage of weak encryption modes to the fixing of initialization vectors and salt parameters that should be freshly sampled at random for each operation.

All these errors are no longer issues of obscure academic debate, but merit front-page news across the globe as the sensitive data of billions of people can be compromised by a single error in a cryptographic library, causing billions in damages. *How can we make sure that the software we trust for the security of our digital society actually **is** trustworthy?* Although testing is a relatively

cheap way to eliminate many vulnerabilities, but it will never be able to guarantee the absence of vulnerabilities. In fact, the attacks listed earlier all affected cryptographic software that did undergo serious testing before being deployed. Auditing, the process of careful code review by independent experts, typically reveals more vulnerabilities than automated testing but is much more expensive than testing and therefore does not scale—and does not guarantee the absence of vulnerabilities.

Formal verification is the only approach that can *guarantee correctness and security* of cryptographic software and thus establish the confidence society needs in cryptography. The idea of formally verifying cryptographic software has been an active of research for years, but little of the software deployed on a wide scale today comes with any guarantee of correctness or security. What is needed is a comprehensive plan to aim at formally verify cryptographic software deployed in the real-world with a plan to migrate real-world applications from current “low-assurance” (or, in many cases, no-assurance) cryptography to ***high-assurance cryptography*** that provides the guarantees provided by formal proofs of correctness and security of the needed cryptographic primitives but also provides access to them in an “easy to use” API. In other words, the goal should be to replace OpenSSL with formally verified cryptography.

3 Formally Verified and Optimized Cryptographic Primitives

Currently, almost no primitives used in real-world deployed software are formally verified due to the speed lost. Cryptographic software is one of the few examples of software that is commonly hand-optimized at the assembly level. The reason for this is that this approach is, at the same time, feasible and worth the effort, because relatively small portions of code are used to encrypt huge amounts of data, perform many key exchanges, compute many signatures, etc. In particular, on busy servers or on battery-powered devices, even small improvements in performance of some core cryptographic routine translate to noticeable improvements in overall system performance or battery life. Consequently, a very active area of research is devoted to optimizing cryptographic software and essentially every serious cryptographic library (especially OpenSSL) contains hand-optimized assembly routines for the most important primitives and target (micro-)architectures. Yet there have been subtle bugs in low-level arithmetic functions that attackers can exploit in this hand-optimized assembly, such as the multiple carry bugs in big-integer arithmetic in OpenSSL [6]. Yet typically, formally verified primitives are not used in real-world deployment because of a performance penalty, as formal verification is done over models of the code using specialized programming environments such as Coq that do not directly translate into running code, and if so, the code is far too slow to be used.

The largest breakthrough so far has been the HAACL* library for high assurance cryptography,¹ which was initially focused on Curve 25519 elliptic

¹ <https://github.com/mitls/hacl-star>.

curves [18]. The HA^{CL}* library is now expanding to include popular stream ciphers (Chacha20, Salsa20, XSalsa20), MACs (Poly1305, HMAC), and more. As it can be compiled in a verified manner down to C, allowing the real world use of verified cryptography [16]. This work is fast enough for real world deployment, as shown by the use of HA^{CL}* by Mozilla in their NSS library.

The challenge is to provide at the same time high speed *and* high assurance. Another example of software that comes reasonably close is the hand-optimized assembly implementation of X25519 key exchange presented in [3,4] (after the bug fix), which has been, to a large extent, proven correct [8]. The proof does not cover the full implementation but only the core loop. More importantly, the proof reveals the main issue with formally proving all widely deployed highly optimized crypto software correct: like also many proofs of less optimized cryptographic software, it required serious manual effort and expert knowledge about both the optimization techniques and the tools used for verification. The amount of code annotations needed for verification by far exceed the amount of actual code. This amount of manual effort does not scale to a larger set of relevant primitives, or to an ever-increasing amount of hand-optimized assembly implementations for an ever-increasing set of microarchitectures.

To allow formally verified cryptography to be usable in practice, there is the need for a verified “low level virtual machine” (LLVM) compiler that optimizes code for micro-architectures in a fully-verified manner in order to permit reaching the performance levels of hand-optimized assembly *and* obtaining formally verified implementations. This includes the verification of the primitives needed by almost all applications—symmetric encryption and authentication, hash functions, key exchange, and digital signatures—while maintaining speed comparable to hand-optimized assembly code for each primitive. This is not impossible if a selected group of modern cryptographic primitives is chosen: Many legacy primitives, such as the MD5 and SHA-1 hash functions, can be broken; so there is no reason to create formally verified implementations of these primitives. Further, although RSA-based cryptography is still widely deployed, it is gradually being replaced by more efficient alternatives that are easier to manage and implement, such as elliptic-curve cryptography (ECC). For example, ECC-based key exchange and digital signatures combined with AES-GCM mode are being adopted in many modern cryptographic deployments on the Internet; this trend is led by large companies such as Amazon, Google, and Microsoft. Curve25519 [2], which will be used for key exchange, encryption/decryption and signature/verification, was recently standardized by the IETF and is used in new versions of TLS and Signal.

The way forward for the formal verification community to accomplish these research goals in terms of cryptographic primitives can be done two phases. A first step towards this goal is to produce possibly slow but formally verified reference implementations in the C programming language of a set of core primitives that are used in state of the art protocols like TLS 1.3 or the Signal secure-messaging protocol. The second step would be to move from C reference implementations to formally proven cryptographic software that is hand-optimized on

the assembly level. In order to avoid the same issue of scalability that previous efforts have been facing, this goal could be achieved producing tools that allow cryptographic engineers to optimize software then obtain a “click-button” verification of correctness; integrated into typical software build environments. This could be done by working with languages such as Jasmin,² a formally verified low-level programming language (largely inspired by the Qhasm programming language by Bernstein [1], which is already today used to write highly optimized cryptographic software). In addition to the efficient register allocator and the instruction-by-instruction translation to assembly offered by Qhasm, Jasmin features a formal specification of its semantics, which allows translation of Jasmin code not only to assembly, and input into formal verification tools that can be produce proofs of equivalence using tools such as GFVerif³ for elliptic curve cryptography between an optimized Jamin implementation and a (verified) CompCert C reference implementation. The tool will aim primarily at proving equivalence of implementations of symmetric primitives such as permutations, block ciphers, or compression functions. This will not come without work, the programmer will be required to annotate code with statements about operations in the underlying finite field; something that sensible programmers already include now as comments in their code.

4 A Developer-Resistant API

The fastest formally optimized cryptographic primitives will still lead to untrustworthy and broken security if they are incorrectly used. A cryptographic API (Application Programming Interface) is used by programmers to access cryptographic primitives and control cryptographic key material as needed in their applications and higher level protocols. Since APIs usually sit between the primitives themselves and their use in applications, secure API design is an important aspect of secure software engineering. However, as shown by the analysis of Android applications in [11], a huge percentage of applications (88%) tend to have errors in their use of cryptographic APIs. Moreover, existing APIs of libraries such as OpenSSL have been shown to be prone to errors,⁴ and these errors can be propagated upwards.

Formal verification is just beginning to be applied to the standardization of security API design, and ad-hoc design by committee should be replaced by a design based on formal foundations. A *security API* consists of a set of functions that are offered to some other program that uphold some security properties, regardless of the functions called or the program calling them [5]. For example, one would hope that an API like PKCS#11 that provides access to key material in hardware tokens would prevent any private key material from being tampered with, regardless of the application [10]. These kinds of security properties are

² <https://github.com/jasmin-lang/jasmin>.

³ <http://gfverif.cryptojedi.org/>.

⁴ <https://www.openssl.org/blog/blog/2016/03/01/an-openssl-users-guide-to-drown/>.

particularly critical in many applications, and classically security APIs have been studied in the realm of hardware security modules [5] and increasingly in developer-facing APIs such as the W3C WebCrypto API for Javascript [7]. Most early work did not use generalizable formal techniques, but customized each technique for the API at hand [5], although some work allowed the automatic discovery of common errors in key management [15]. Formal modeling has also been used to successfully reveal a number of API-based attacks on standards, including the commercially available tamper-resistant hardware security tokens PKCS#11 [10]. Although a single program may only use one (or a few) APIs, complex systems such as banking operations consist of thousands of applications, with even more calls to multiple APIs. Of these, although some APIs may be standard, other APIs may be hand-crafted by amateurs, and basic errors such as calling deterministic “random” number generators from the programming language are common.

API design should not only be based on sound formal foundations, but also from the concrete results from usability studies of APIs [17]. Almost all APIs across programming languages allow users to make common errors, ranging from nonce re-use to failure to randomize initialization vectors [12]. These account for the vast majority of errors in code and the “top errors” in APIs that have recently been collated by Google’s Project Wycheproof⁵. Key management is often underspecified in APIs, and is a common source of errors in systems relying for example on PKCS#11 [10] and the WebCrypto API [7], and simply putting the key material in “trusted hardware” such as hardware tokens may end up having little effect, as shown by errors discovered via formal analysis Yubico’s YubiHSM.⁶ APIs created by standards committees seem to fare no better: implementations of standardized APIs such as PKCS#11 are often susceptible to multiple attacks.⁷ Even worse, when APIs such as PKCS#11 and OpenSSL are used in hardware tokens, errors in the API can cause expensive withdrawal of hardware tokens [14].

The API market today is fractured, with the vast majority of even commercial software being bound to OpenSSL (including the embedding of OpenSSL even in hardware tokens) or various programming-language specific cryptographic APIs. Due to the number of bugs, a number of branches of OpenSSL have happened, ranging from Google’s BoringSSL to WolfSSL for lightweight embedded systems. However, none of these efforts have been formally verified, and all of them are under the control of some external entity. IPSec libraries used in VPNs such as OpenSwan and LibreSwan are similarly unverified. Naturally, few other companies want to be dependent on the commercial interests and whims of Google’s strategy by becoming tied to BoringSSL.

What is lacking is a flexible API with safe defaults for developers that covers core modern cryptographic primitives – cryptographic primitives that themselves are verified. In order to allow easy “drop-in” replacement of OpenSSL,

⁵ <https://github.com/google/wycheproof>.

⁶ <https://www.yubico.com/wp-content/uploads/2012/10/Security-Advisory.pdf>.

⁷ <https://cryptosense.com/the-untold-story-of-pkcs11-hsm-vulnerabilities/>.

as much of the OpenSSL library should be mapped to the new API as possible for upgrading legacy software. Still, access to these primitives by themselves, even for well-known primitives such as AES-CBC, will almost certainly result in developer errors. So for new software and as a recommended developer-facing API, the API will provide “safe” defaults and layers of abstraction to defend the programmer against their own errors, such as preventing the re-use of nonces and randomly initializing initialization vectors. Furthermore, common errors involving key management, such as key generation, rotation, revocation, and wrapping, can also all be dealt with on a level of abstraction that enforces usages boundaries and sensible “defaults” for best practices for key-handling. For example, if a key is generated, the minimum size as recommended by the ECRYPT “Algorithms, Key Size, and Parameters” report will be used.⁸ If there is only a limited number of modern cryptographic primitives verified, then finding “safe” defaults for those primitives and building in proper key-handling (for example, to prevent the same keys for being used in signing and encryption) should be possible in a new high-assurance API. In terms of deployment, a three-pronged strategy is needed (1) The older unverified OpenSSL or other API bindings can be removed and replaced with a high assurance API if the cryptographic primitive is supported (2) Application developers that do not have much experience in cryptography can also use a version with simplified primitives that will automatically chose fast, verified algorithms with “safe defaults” for the developer (3) Advanced developers should be able to override all defaults.

5 Conclusion

In order to make high assurance cryptography a reality, two steps need to be taken. First, formally verified primitives must be comparable in speed to hand-optimized assembly on a per-platform basis. This can be done through formally verified C compilation (including C produced from higher-level verified specifications using languages), and per-architecture optimization using a LLVM that can have equivalence proofs to the formally verified specifications. Second, deploying these primitives in actual applications will require an API that can replace OpenSSL for modern applications, and be easier to use than OpenSSL with safer defaults. Furthermore, as new privacy-preserving primitives such as algebraic MACs and post-quantum primitives reach maturity, these new primitives can be formally verified and added to the API.

It should be noted that the task also extends beyond simply replacing the cryptographic primitives, as the correct usage both OpenSSL and any verified replacement requires the verified correct parsing of data formats, as exemplified by the difficult work of “Project Everest” to parse X.509 certificates in its complete reimplementaion of TLS.⁹ Without at least one usable API featuring formally verified primitives, there is a little chance of moving beyond OpenSSL.

⁸ <https://www.cosic.esat.kuleuven.be/ecrypt/csa/documents/D5.2-AlgKeySizeProt-1.0.pdf>.

⁹ <https://project-everest.github.io/>.

Another aspect that formal verification tools can help address is safe memory management, but further work also needs to be done to ensure that sensitive data such as keys are kept in memory for the minimal needed amount of time. Lastly, we are assuming the processor has correctly implemented the LLVM model and that the LLVM has no features outside the LLVM model capable of being used in an attack, and thus more research needs to be done in formally verifying that actual processors match their specifications [9].

The next step is to present the architecture of a library as sophisticated as OpenSSL and for each building-block of the API explain the security goals and how they can be addressed using formal methods, including the verification of their non-trivial composition in higher-level protocol frameworks such as the Noise framework.¹⁰ With such primitives easily usable by an API, formal verification can serve as the foundation for high assurance cryptography.

Acknowledgments. Harry Halpin is funded in part by the European Commission H2020 European Commission through the NEXTLEAP Project (Grant No. 6882). Harry Halpin would like to thank Peter Schwabe for many of the original ideas in this paper and for some of the text itself, while taking sole responsibility for any lack of clarity or problems with this paper.

References

1. Bernstein, D.J.: Qhasm: tools to help write high-speed software. <http://cr.yp.to/qhasm.html>. Accessed 04 Dec 2017
2. Bernstein, D.J.: Curve25519: new Diffie-Hellman speed records. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 207–228. Springer, Heidelberg (2006). <https://doi.org/10.1007/11745853-14>. <http://cr.yp.to/papers.html#curve25519>
3. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 124–142. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23951-9_9. See also full version [4]
4. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.-Y.: High-speed high-security signatures. *J. Cryptograph. Eng.* **2**(2), 77–89 (2012). <http://cryptojedi.org/papers/#ed25519>. See also short version [3]
5. Bond, M., Anderson, R.: API-level attacks on embedded systems. *Computer* **34**(10), 67–75 (2001)
6. Brumley, B.B., Barbosa, M., Page, D., Vercauteren, F.: Practical realisation and elimination of an ECC-related software bug attack. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 171–186. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-27954-6_11
7. Cairns, K., Halpin, H., Steel, G.: Security analysis of the W3C web cryptography API. In: Chen, L., McGrew, D., Mitchell, C. (eds.) SSR 2016. LNCS, vol. 10074, pp. 112–140. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49100-4_5

¹⁰ <http://noiseprotocol.org>.

8. Chen, Y.-F., Hsu, C.-H., Lin, H.-H., Schwabe, P., Tsai, M.-H., Wang, B.-Y., Yang, B.-Y., Yang, S.-Y.: Verifying curve25519 software. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 299–309. ACM (2014). <http://cryptojedi.org/papers/#verify25519>
9. Choi, J., Vijayaraghavan, M., Sherman, B., Chlipala, A., et al.: Kami: a platform for high-level parametric hardware specification and its modular verification. *Proc. ACM on Program. Lang.* **1**(ICFP), 24 (2017)
10. Delaune, S., Kremer, S., Steel, G.: Formal security analysis of PKCS#11 and proprietary extensions. *J. Comput. Secur.* **18**(6), 1211–1245 (2010)
11. Egele, M., Brumley, D., Fratantonio, Y., Kruegel, C.: An empirical study of cryptographic misuse in android applications. In: Sadeghi, A.-R., Gligor, V.D., Yung, M. (eds.) 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, Berlin, Germany, 4–8 November 2013, pp. 73–84. ACM (2013)
12. Gorski, P.L., Iacono, L.L.: Towards the usability evaluation of security APIs. In: HAISA, pp. 252–265 (2016)
13. The heartbleed bug (2014). <http://heartbleed.com>
14. Künnemann, R., Steel, G.: YubiSecure? Formal security analysis results for the Yubikey and YubiHSM. In: Jøsang, A., Samarati, P., Petrocchi, M. (eds.) STM 2012. LNCS, vol. 7783, pp. 257–272. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38004-4_17
15. Longley, D., Rigby, S.: An automatic search for security flaws in key management schemes. *Comput. Secur.* **11**(1), 75–89 (1992)
16. Protzenko, J., Zinzindohoué, J.-K., Rastogi, A., Ramananandro, T., Wang, P., Zanella-Béguelin, S., Delignat-Lavaud, A., Hrițcu, C., Bhargavan, K., Fournet, C., et al.: Verified low-level programming embedded in F. *Proc. ACM Program. Lang.* **1**(ICFP), 17 (2017)
17. Whitten, A., Tygar, J.D.: Why Johnny can’t encrypt: a usability evaluation of PGP 5.0. In: USENIX Security, vol. 1999 (1999)
18. Zinzindohoué, J.K., Bartzia, E.-I., Bhargavan, K.: A verified extensible library of elliptic curves. In: 2016 IEEE 29th Computer Security Foundations Symposium (CSF), pp. 296–309. IEEE (2016)

Privacy



Privacy-Preserving Equality Test Towards Big Data

Tushar Kanti Saha¹^(✉) and Takeshi Koshihara²

¹ Division of Mathematics, Electronics, and Informatics,
Graduate School of Science and Engineering, Saitama University, Saitama, Japan
saha.t.k.512@ms.saitama-u.ac.jp

² Faculty of Education and Integrated Arts and Sciences,
Waseda University, Tokyo, Japan
tkoshihara@waseda.jp

Abstract. In this paper, we review the problem of private batch equality test (PriBET) that was proposed by Saha and Koshihara (3rd APW-ConCSE 2016). They described this problem to find the equality of an integer within a set of integers between two parties who do not want to reveal their information if they do not equal. For this purpose, they proposed the PriBET protocol along with a packing method using the binary encoding of data. Their protocol was secured by using ring-LWE based somewhat homomorphic encryption (SwHE) in the semi-honest model. But this protocol is not fast enough to address the big data problem in some practical applications. To solve this problem, we propose a base- N fixed length encoding based PriBET protocol using SwHE in the same semi-honest model. Here we did our experiments for finding the equalities of 8–64-bit integers. Furthermore, our experiments show that our protocol is able to evaluate more than one million (resp. 862 thousand) of equality comparisons per minute for 8-bit (resp. 16-bit) integers with an encoding size of base 256 (resp. 65536). Besides, our protocol works more than 8–20 in magnitude than that of Saha and Koshihara.

Keywords: Private batch equality test · Base- N encoding
Homomorphic encryption · Packing method

1 Introduction

Since the establishment of Internet technology, data are increasing day by day in an expeditious speed. Also, users are extensively using the computers, laptops, tabs along with the Internet. Besides, smart-phones and Wi-Fi devices are helping us to use the Internet even when we are mobile. So data are becoming very big which are called big data. Managing and analyzing big data is a big challenge for its user where new tools and techniques are indispensable. In addition, local storage is not enough for the users to store their data. Recently, banks, insurance companies, hospitals, research institutes, public service centers, and so on have come forward to store their customers' data electronically

and maintain their databases. Besides, cloud computing has established itself as a reliable service provider by giving remote on-line storage to its users at an affordable price. Moreover, organizations are interested in outsourcing their data to the cloud servers to access them anytime from versatile locations. Also, these organizations want to provide security to their data. Here encryption is one of the procedures to provide data security. In addition to this, the organizations want to do their required operations on the encrypted data which is hard to perform before decryption. So homomorphic encryption [9] is a solution for them which allows meaningful computations like additions and multiplications on the encrypted data.

On the contrary, many research works in secure computation (for example, [1, 5, 10, 12, 14]) have already been conducted using ring-LWE based homomorphic encryption after the breakthrough work of fully homomorphic encryption (FHE) by Gentry [4] in 2009. However, fully homomorphic encryption allows any number of additions and multiplications on the encrypted data which makes it slower for practical use [10]. In 2011, Brakerski and Vaikuntanathan [1] proposed one more somewhat homomorphic encryption (SwHE) using the concept of ring-LWE which works little faster due to supporting many additions and few multiplications. Thereafter it had been used in the literature [5, 10, 14] which showed the practicality of SwHE. Now some organizations are needed to share their data with other organizations for different purpose like private data mining, machine learning, searching, information retrieval, and so on where private equality test is important. But data protection regulation does not allow these organizations to share their data with one another or even with the cloud. Here PET protocol using SwHE can be used by these organizations which was proposed by Saha and Koshiba [11]. They also proposed private batch equality test (PriBET) protocol for finding the equality of an integer with a set of integers using the batch technique. But their achieved performance is not good enough to handle a large dataset of millions of integers. Furthermore, most of the organizations deal with a large dataset which varies from several gigabytes to terabyte where a faster performance is the prerequisite to search something within their datasets.

1.1 Prior Works

The idea of secure computation was first introduced by Yao in 1980 [17]. In this section, we review some papers on secure computation for the PriBET protocols. To date, very few protocols have been proposed for the PriBET protocol. In 2016, Couteau [3] proposed the PriBET protocol in the semi-honest model that required 7 rounds communication between two-party to compare data size of 16–128-bit. But they did not show any implementation. To understand the practicality of the protocol, some implementations are indispensable. Recently, Saha and Koshiba [11] proposed private batch equality test (PriBET) protocol which shows some practicality of the protocol. Their protocol was able to do about 140 thousand comparisons per minute for 8-bit data only. The performance decreases if the data size is increased more. To date, existing equality

protocols are not enough to handle a large database for many equality queries. So new method is desirable which can be a big step towards big data processing.

1.2 Typical Applications

In 2016, Saha and Koshiba [11] showed some application areas of the PriBET protocols including on-line auction, genomic computation, machine learning, data mining, and database query processing where batch equality protocol is required. Besides, PriBET protocol is useful for private information retrieval [18]. Over and above that, our protocol may be useful in some practical applications like credit card number verification, criminal database searching using social security number, insurance number verification, and so on.

1.3 Motivation

In 2016, Saha and Koshiba [11] proposed private batch equality test (PriBET) protocol for comparing integers of 8–32 bits with a new packing method using ring-LWE based SwHE. Here they achieved an acceptable performance for handling private equality computation of a small dataset like several megabytes to a gigabyte in size. But this performance is not enough for addressing big data which refers several gigabytes to a terabyte. Saha and Koshiba performed equality computations on binary vectors which required a large lattice dimension to process a kilobyte of data. For example, processing 1 MB data (2^{23} bits) requires a lattice dimension of 2^{23} where they consider a lattice dimension of 2^{12} to get more efficiency. If we engage the protocol with a lattice dimension of 4096, then it would require about 351 s to process 1 MB of data for equality which in turns requires about 100 h to process 1 GB of data with a single machine. This performance could be further improved by using some parallel processing techniques or engaging many computers in a distributed computing environment. But this performance is not enough for handling a large database. Furthermore, the processing speed of this protocol is mostly dependent on lattice dimension where they used binary encoding to find the equality. At this point, this protocol can be improved further if we would have a method to reduce the lattice dimension.

1.4 Our Contribution

In this paper, we propose a base- N fixed length based PriBET protocol for finding the equalities of some integers with 8–32 bits along with an efficient data encoding technique to reduce the lattice dimension as well as processing time. Theoretically, we achieve a reduction of the lattice dimension by a factor $\log_2(N)$ than Saha and Koshiba protocol [11] due to using an efficient encoding technique where N represents the encoding size. Also, our practical experiments show that our protocol works more than 8 times faster than Saha and Koshiba protocol [11]. In addition, we have been able to process more than 1 million comparisons per minute for the 8-bit integers and 862 thousand comparisons per minute for

the 16-bit integers with the encoding size of 2^8 and 2^{16} respectively which could be helpful to process a big database. Besides, in the PriBET protocol of Saha and Koshiha Bob needed a decryption help from Alice to check a part of his result for some decision making after his computation. In this case, he leaked some additional information to Alice due to sending whole encrypted polynomial to Alice. Here we minimize the information leakage problem through random masks that occurred in [11]. Besides, we show the practical upper bound of the encoding size of our protocol using ring-LWE SwHE.

Remark 1. Here we compare the performances of the both methods which are implemented in a single PC environment configured with one 3.6 GHz Intel core-i7 CPU and an 8 GB RAM in Linux environment.

2 Data Encoding Technique

In this section, we review the Saha and Koshiha data encoding technique [11] and discuss our base- N data encoding technique. The base- N encoding was also used by Yasuda et al. [16] to pack a large integer vector of 16–32-bit for an efficient statistical analysis. But we use base- N fixed length encoding where most significant digits (MSDs) are filled up by ‘0’ if it is empty in the encoded number. The reasons for choosing the base- N encoding are described below.

Notations. In this paper, \mathbb{Z} denotes the ring of integers. In addition, R denotes a ring of integer of the form $\mathbb{Z}[x]/f(x)$ where $f(x)$ denotes a cyclotomic polynomial of degree n as $f(x) = x^n + 1$ with a lattice dimension n . For a prime number q , the ring of integer modulo q is denoted by \mathbb{Z}_q . The ciphertext space is denoted by the ring $R_q = R/qR = \mathbb{Z}_q[x]/f(x)$. For an integer $t < q$, the message space is defined as $R_t = \mathbb{Z}_t[x]/f(x)$. Besides, $\mathbb{Z}[x]$ denotes the ring of polynomials over integers. For a vector $\mathbf{A} = (a_0, a_1, \dots, a_{n-1})$, the maximum norm of $\|\mathbf{A}\|_\infty$ is defined as $\max |a_i|$. Let $\langle \mathbf{A}, \mathbf{B} \rangle$ denote the inner product between two vectors \mathbf{A} and \mathbf{B} . Moreover, the function $\text{Enc}(m, pk) = ct(m)$ defines the encryption of message m using the public key pk to produce the ciphertext ct . Also, l and l_N denote the length of an integer in binary and base- N fixed length encoding respectively. Besides, γ and k represent the block size and the total number of records respectively where a block is a collection of records.

In 2016, Saha and Koshiha [11] used binary encoding technique over the alphabets $\{0, 1\}^l$ for their private batch equality protocol (PriBET) where they got an acceptable performance for practical use for a batch data size k . But the protocol is not fast enough for big data processing. From the Table 1 in [11], we observed that the speed of the PET protocol mostly depends on the lattice dimension. In the secure computation, they required three homomorphic multiplications over a polynomial ring R_q . In the ring-LWE lattice-based homomorphic encryption scheme, a homomorphic multiplication requires doing a large polynomial multiplication over a lattice dimension of at least $n = 2048$ to achieve a security level over 128-bit [11]. Saha and Koshiha showed the private batch equality tests for 8–32-bit integers within the lattice dimension of 2048–4096.

Here we observed that the lattice dimension is increasing with the increase of data size l and batch size k . For example, to process a 16-bit integers comparison, executing PriBET on the block size of 128 required a lattice dimension of $n = l \cdot k = 16 \cdot 128 = 2048$. On the other hand, it required a lattice dimension of $n = l \cdot k = 16 \cdot 256 = 4096$ for a batch size of 256. At this point, minimizing the lattice dimension is indispensable to minimize the computation time of the PriBET protocol. If we able to use an encoding technique other than binary, then we can reduce the lattice dimension. Moreover, we call the used binary encoding in [11] as base-2 encoding where alphabet set is $\{0, (2-1)\}^l = \{0, 1\}^l$. In addition, a binary encoding is using alphabets '0' and '1' to convert any decimal number z which requires $l = \lceil \log_N(z) \rceil + 1$ digits where $N = 2$ in this case. Saha and Koshiba used an l -bit binary conversion algorithm for any integer of $l = 8-32$ -bit. That means, if the required number of binary digit to represent any integer is less than l then rest of the MSBs are filled up by zeros. If we can do the encoding over a large alphabet set, then we can reduce the lattice dimension.

Data: z, N and l ;
Result: base- N fixed length number;
 Input z, N and l ;
 Set $l_N = l / \log_2(N)$;
 $z_{baseN} = \text{baseNConvert}(z, N, l_N)$;
 Output z_{baseN} ;
 Procedure: $\text{baseNConvert}(z, N, l_N)$
forall the $i \in l_N$ **do**
 | set $digit[i] = 0$;
end
 set $ind = 0$;
while ($z \neq 0$) **do**
 | $r = z \bmod N$;
 | $z = z / N$;
 | $digit[ind] = r$;
 | $ind++$;
end
 return $digit$;

Algorithm 1. Base- N fixed length encoding algorithm

Now we show the mathematical structure how the base- N encoding can work faster than base-2 encoding where $N > 2$. Here we are dealing with lattice-based cryptography where the working speed mostly depends on the lattice dimension n . To process k data of size l -bit using binary encoding, Saha and Koshiba required a lattice dimension n' of

$$n' = k \cdot l. \quad (1)$$

On the contrary, to represent an l -bit integer in base- N encoding, we need a vector of size of

$$l_N = \lceil l / \log_2(N) \rceil. \quad (2)$$

So using base- N encoding, the new lattice dimension n'' can be determined from batch size k and base- N vector size l_N as

$$n'' = k \cdot l_N. \quad (3)$$

Now by dividing Eq.(1) by Eq.(3) and with the help of Eq.(2), the relation between new lattice dimension n'' and Saha and Koshiha lattice dimension n' can be obtained as

$$n'/n'' = l/l_N = \log_2(N). \quad (4)$$

Here we get the lattice dimension reduction rate as a factor of $\log_2(N)$. So we use base- N fixed length encoding for any positive integer in \mathbb{Z}_t using the alphabet set $\{0, 1, 2, \dots, N - 1\}^{l_N}$. Now we slightly modify the basic base- N conversion algorithm to make it fixed length to get our algorithm for base- N encoding as shown in Algorithm 1. From this algorithm, we get the base- N fixed length encoding of an integer in \mathbb{Z}_t by putting '0' in the MSDs if the actual length of the base- N encoded vector is less than l_N . From the above discussion, it is clear that our encoding scheme also reduces the size of any integer vector from its binary representation with the ratio of $l : l/\log_2(N)$. In addition, we believe that our encoding technique can be used in other contexts where the length reduction of binary encoded vectors and batch computation of the many Euclidean distances are indispensable.

3 Our Protocol

Saha and Koshiha [11] proposed the PriBET protocol using SwHE with binary encoding in the semi-honest model. Here we propose another protocol called base- N PriBET protocol using base- N encoding described in Sect. 2 to increase its efficiency that is described as follows.

Consider a bank (Alice) wants to sanction some home loans to its customers who have good credit score and are paying their taxes regularly. Suppose that a customer of the bank now applies for a new home loan who has good credit score with the required information along with his tax certificate. Furthermore, the bank (Alice) needs to verify the customer's tax identification number (TIN) to check his status while sanctioning a new loan. On the contrary, the national tax department (Charlie) is maintaining the database of all its taxpayers. Now neither the bank can disclose its customer's information to the national tax department nor the national tax department can disclose it's all customers' information to the bank. Here a third party like Bob in the cloud can solve this problem and does the verification on behalf of them without knowing the actual tax number from both parties. This is a problem of verifying the equality of a large integer with a large set of integers.

From the above scenario, let Alice has an l -bit integer which can be represented by a base- N vector as $\alpha = (\alpha_1, \dots, \alpha_{l_N})$ by using Algorithm 1. In addition, the national tax department has k integers with the same size that can be represented by the base- N integer vectors as $\beta_\lambda = (\beta_{\lambda,1}, \dots, \beta_{\lambda,l_N})$ by applying same algorithm where $1 \leq \lambda \leq k$. As we know from Saha and Koshiha [11],

the Hamming distance between two l -bit integers can find out whether they are equal or not. But the Hamming distance works for only binary vectors. Therefore, we use the concept that two integers are equal when the square Euclidean distance (SED) between their vectors using base- N fixed length encoding will be 0. Now the equality test for batch comparisons can be realized by the following equation as

$$E_\lambda = \sum_{i=1}^{l_N} (\alpha_i - \beta_{\lambda,i})^2 \quad (5)$$

where $1 \leq \lambda \leq k$. Moreover, E_λ represents the square Euclidean distances (SEDs) between two base- N vectors α and β_λ . Moreover, if E_λ in Eq. (5) is 0 for some positions of λ then we can say that $\alpha = \beta_\lambda$; otherwise $\alpha \neq \beta_\lambda$. In this way, Alice securely verifies her customer with the help of Bob. Now we describe our protocol by the following steps.

Inputs: $\alpha = (\alpha_1, \dots, \alpha_{l_N})$ and $\{\beta_1, \beta_2, \dots, \beta_k\}$, where $\beta_\lambda = (\beta_{\lambda,1}, \dots, \beta_{\lambda,l_N})$ for each λ in $\{1, 2, \dots, k\}$.

Output: $\exists \lambda[\alpha = \beta_\lambda]$ or $\forall \lambda[\alpha \neq \beta_\lambda]$

Base- N PriBET protocol:

1. By using SwHE, Alice constructs the public key and private key by herself and sends the public key to Charlie through a secure channel.
 2. Then Alice encrypts the customer's TIN $\alpha = (\alpha_1, \dots, \alpha_{l_N})$ using her public key and sends it to Bob.
 3. The national tax department (Charlie) also uses the public key given by Alice to encrypt k TINs $\beta_\lambda = (\beta_{\lambda,1}, \dots, \beta_{\lambda,l_N})$ where $1 \leq \lambda \leq k$ and sends the value to Bob.
 4. Bob does the computation in Eq. (5) on the encrypted TINs and sends the encrypted result $ct(E_\lambda)$ to Alice to verify whether at least one of E_λ is equal to 0.
 5. For $1 \leq \lambda \leq k$, Alice decrypts $ct(E_\lambda)$ using her secret key and checks each value E_λ .
 6. If Alice finds at least one of the $E_\lambda = 0$ then she decides the match; otherwise, she decides no match.
-

Remark 2. Here our protocol provides the passive security under the assumption that Bob is semi-honest. In other words, Bob follows the protocol but tries to learn information from the protocol. Furthermore, we use the same ring-LWE based SwHE scheme used in Saha and Koshiba [11] for the security of our protocol. In this section, we skip its review due to page limitation. Besides, the security assumption of the scheme is based on the ring-LWE assumption which is reducible to the worst-case hardness of problems on ideal lattices that is believed to be secure against the quantum computer as mentioned by Lyubashevsky et al. [7].

Remark 3. The goal of our protocol is to find $\alpha = \beta_\lambda$ or $\alpha \neq \beta_\lambda$ for some $1 \leq \lambda \leq k$. Now we also think about the security of index λ . In our base- N PriBET protocol, Alice can know such index λ . Since such index is not actual index that exists in the databases of the national tax department, so leakage of such information to Alice does not harm the security of our protocol.

4 Packing Method

Packing method is the process of representing many data in a single polynomial. We know from some existing literature [5, 10, 11, 14] that packing method makes many secure computations using ring-LWE SwHE more practical. Recently, Saha and Koshiha [11] used binary vectors to address their packing. Here we consider the same packing with base- N fixed length vectors. Now we review the packing method of PriBET protocol in [11] for our protocol using our base- N encoding by the following way.

4.1 Packing Method for Our Protocol

As mentioned in our protocol of Sect. 3, we need to compute the SEDs E_λ in Eq. (5) using few polynomial additions and multiplications to reduce the cost where $1 \leq \lambda \leq k$. Let us construct a base- N integer vector $\mathbf{A} = (\alpha_0, \dots, \alpha_{l_N-1}) \in R_t$ from a base- N vector $\alpha = (\alpha_1, \dots, \alpha_{l_N})$ of length l_N . Furthermore, let us consider another base- N integer vectors \mathbf{B} which is constructed by combining all base- N vectors in $\beta_\lambda = (\beta_{\lambda,1}, \dots, \beta_{\lambda,l_N})$ as $\mathbf{B} = (\beta_{1,0}, \dots, \beta_{1,l_N-1}, \dots, \beta_{k,0}, \dots, \beta_{k,l_N-1}) \in R_t$ of length $k \cdot l_N$. Here we want to compute many SEDs E_λ in one computation which can be done by measuring the SEDs between the vector \mathbf{A} and each sub-vector in \mathbf{B} . Moreover, existing literature showed [10, 14] that the secure inner product $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle$ helps to compute the SED between \mathbf{A} and \mathbf{B}_λ . Hence, we pack these integer vectors by some polynomials with the highest degree ($x = n$) in such a way so that inner product $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle$ does not wrap-around a coefficient of x with any degrees. For the integer vectors \mathbf{A} and \mathbf{B} with $n \geq k \cdot l_N$ and $1 \leq \lambda \leq k$, the packing method of Saha and Koshiha [11] in the same ring $R = \mathbb{Z}[x]/(x^n + 1)$ is rewritten as

$$Poly_1(\mathbf{A}) = \sum_{i=0}^{l_N-1} \alpha_i x^i, \quad Poly_2(\mathbf{B}) = \sum_{\lambda=1}^k \sum_{j=0}^{l_N-1} \beta_{\lambda,j} x^{l_N \cdot \lambda - (j+1)}. \quad (6)$$

Here the coefficients α_i and $\beta_{\lambda,j}$ are in the alphabets $\{0, 1, 2, \dots, N-1\}^{l_N}$ instead of alphabets $\{0, 1\}^l$ as in [11]. If we multiply the above two polynomials, we can get the inner product computations which in turn helps the many square Euclidean distances computation between \mathbf{A} and \mathbf{B}_λ . Moreover, this multiplication will produce another big polynomial where each of the SEDs can be obtained as a coefficient of x with different degrees. According to the SwHE described in

Sect. 5 of [11], the packed ciphertexts for $Poly_i(\mathbf{A}) \in R$ are defined for some $i = \{1, 2\}$ using the public key pk as

$$ct_i(\mathbf{A}) = \text{Enc}(Poly_i(\mathbf{A}), pk) \in (R_q)^2. \quad (7)$$

To get the inner product of the vectors \mathbf{A} and \mathbf{B}_λ , we multiply the polynomials $Poly_1(\mathbf{A})$ and $Poly_2(\mathbf{B})$ in the same base ring R as follows.

$$\begin{aligned} \left(\sum_{i=0}^{l_N-1} \alpha_i x^i \right) \times \left(\sum_{\lambda=1}^k \sum_{j=0}^{l_N-1} \beta_{\lambda,j} x^{l_N \cdot \lambda - (j+1)} \right) &= \sum_{\lambda=1}^k \sum_{i=0}^{l_N-1} \sum_{j=0}^{l_N-1} \alpha_i \beta_{\lambda,j} x^{i+l_N \cdot \lambda - (j+1)} \\ &= \sum_{\lambda=1}^k \sum_{i=0}^{l_N-1} \alpha_i \beta_{\lambda,i} x^{l_N \cdot \lambda - 1} + \text{ToHD} + \text{ToLD} = \sum_{\lambda=1}^k \langle \mathbf{A}, \mathbf{B}_\lambda \rangle x^{l_N \cdot \lambda - 1} + \dots \end{aligned} \quad (8)$$

Here \mathbf{A} is the vector of length l_N and \mathbf{B}_λ is the λ -th sub-vector of \mathbf{B} of the same length with $1 \leq \lambda \leq k$. Moreover, the ToHD (terms of higher degree) means $\deg(x) > l_N \cdot \lambda - 1$ and the ToLD (terms of lower degrees) means $\deg(x) < l_N \cdot \lambda - 1$. The result in Eq. (8) shows that one polynomial multiplication includes the many inner products of $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle$. In addition, the following proposition is needed to hold for computing the many inner products over packed ciphertexts.

Proposition 1. *Let $\mathbf{A} = (\alpha_0, \alpha_1, \dots, \alpha_{l_N-1}) \in R_t$ be an integer vector where $|\mathbf{A}| = l_N$ and $\mathbf{B} = (\beta_{1,0}, \dots, \beta_{1,l_N-1}, \dots, \beta_{k,0}, \dots, \beta_{k,l_N-1}) \in R_t$ be another integer vector of length $k \cdot l_N$. For $1 \leq \lambda \leq k$, the vector \mathbf{B} includes k sub-vectors where the length of each sub-vector is l_N . If the ciphertext of \mathbf{A} and \mathbf{B} can be represented as $ct_1(\mathbf{A})$ and $ct_2(\mathbf{B})$ respectively by Eq. (7) then under the condition of Lemma 1 (See Sect. 5 in [11] for details), decryption of homomorphic multiplication $ct_1(\mathbf{A}) \boxtimes ct_2(\mathbf{B}) \in (R_q)^2$ will produce a polynomial of R_t with $x^{l_N \cdot \lambda - 1}$ including coefficient $\langle \mathbf{A}, \mathbf{B}_\lambda \rangle = \sum_{i=0}^{l_N-1} \alpha_i \beta_{\lambda,i} \pmod{t}$. Alternatively, we can say that homomorphic multiplication of $ct_1(\mathbf{A})$ and $ct_2(\mathbf{B})$ simultaneously computes the many inner products for $1 \leq \lambda \leq k$ and $0 \leq i \leq (l_N - 1)$.*

5 Secure Computation Using Euclidean Distance

We perform the computation of base- N PriBET protocol of Sect. 3 using the SwHE scheme used in [11] and the packing method in Sect. 4.1. In addition, according to Eq. (5), we need to find out the values of the many SEDs E_λ . Let us consider two same base- N integers vectors \mathbf{A} and \mathbf{B} constructed by Algorithm 1 where $\mathbf{B}_\lambda = (\beta_{\lambda,0}, \dots, \beta_{\lambda,l_N-1})$ is the λ -th sub-vector of \mathbf{B} with $1 \leq \lambda \leq k$. From these integer vectors, E_λ can be computed with the help of the arithmetic computation between \mathbf{A} and \mathbf{B}_λ as

$$E_\lambda = \sum_{i=0}^{l_N-1} (\alpha_i^2 + \beta_{\lambda,i}^2 - 2\alpha_i \beta_{\lambda,i}). \quad (9)$$

Now we construct $Poly_1(\mathbf{A})$ and $Poly_1(\mathbf{A}^2)$ (resp., $Poly_2(\mathbf{B})$ and $Poly_2(\mathbf{B}^2)$) from vector \mathbf{A} (resp., \mathbf{B}) using the packing method in Eq. (6). With the help of inner product property in Eq. (8), now we compute $ct(E_\lambda)$ of the Eq. (9) over packed ciphertext $ct_1(\mathbf{A})$, $ct_1(\mathbf{A}^2)$, $ct_2(\mathbf{B})$, and $ct_2(\mathbf{B}^2)$ which are obtained from $Poly_1(\mathbf{A})$, $Poly_1(\mathbf{A}^2)$, $Poly_2(\mathbf{B})$, and $Poly_2(\mathbf{B}^2)$ respectively by the Eq. (7). Moreover, we calculate $ct(E_\lambda)$ from Proposition 1 and the packed ciphertext vector $ct_1(\mathbf{A}) \in R_q$, $ct_1(\mathbf{A}^2) \in R_q$, $ct_1(\mathbf{B}) \in R_q$, and $ct_2(\mathbf{B}^2) \in R_q$ in three homomorphic multiplications and two homomorphic additions. Here $ct(E_\lambda)$ equals

$$ct_1(\mathbf{A}^2) \boxtimes ct_2(\mathbf{V}_\epsilon) \boxplus ct_2(\mathbf{B}^2) \boxtimes ct_1(\mathbf{V}_{l_N}) \boxplus (-2ct_1(\mathbf{A}) \boxtimes ct_2(\mathbf{B})) \quad (10)$$

where \mathbf{V}_ϵ denotes an integer vector like $(1, \dots, 1)$ of length $k \cdot l_N$ and \mathbf{V}_{l_N} denotes another integer vector $(1, \dots, 1)$ of length l_N . In addition, \boxplus (resp. \boxtimes) stands for homomorphic addition (resp., multiplication). The above-encrypted polynomial $ct(E_\lambda)$ includes many SEDs as the coefficients of different degrees of x . Bob sends $ct(E_\lambda)$ to Alice for decryption. According to Proposition 1 and our protocol, Alice decrypts $ct(E_\lambda)$ in the ring R_q using her secret key and extracts E_λ as a coefficient of $x^{l_N \cdot \lambda - 1}$ from the plaintext of $ct(E_\lambda)$. Then Alice checks whether at least of one of the E_λ contains 0 or not to help Bob to decide either equality or non-equality.

Concealing Extra Information from Leakage. In the PriBET protocol of Saha and Koshiha [11], Bob in the cloud sent the whole encrypted polynomial to Alice for decryption to decide something after the computation. Here Bob could see every coefficient of the polynomial whereas she needs to check some coefficients with a particular degree of x . For this reason, some extra information leakage problem exists in the Saha and Koshiha protocol. To compute E_λ for our protocol by Bob in the cloud, he also needs a decryption help from Alice for some decision making since he does not have the secret key. From the above discussion of secure computation, Alice needs to check only the coefficient of $x^{l_N \cdot \lambda - 1}$ for the large polynomial $ct(E_\lambda)$ produced by Bob. Also, all other coefficients of our n degree polynomial can be published to Alice if Bob does not conceal those coefficients. In our protocol, we conceal the extra information from leakage to Alice by adding some random masks at the cloud (Bob) ends. We can conceal $ct(E_\lambda)$ by adding a random polynomial r in the base ring R as $r = \sum_{h=1}^{n/l_N} \sum_{i=l_N(h-1)}^{l_N \cdot h - 2} r_i x^i$. Now Bob adds r to the ciphertext $ct(E_\lambda)$ as $ct(E'_\lambda) = ct(E_\lambda) \boxplus r$. Besides, the resulting ciphertext $ct(E'_\lambda)$ contains all required information as a coefficient of $x^{l_N \cdot \lambda - 1}$ and conceals all other coefficients using the random masks. In this way, we protect $ct(E_\lambda)$ from leaking any information to Alice except the coefficient of $x^{l_N \cdot \lambda - 1}$.

6 Experimental Analysis

In this section, we show the parameter settings of our experiments along with security level. We also show the selection process of our base- N encoding size and the performance of our protocol towards big data.

Table 1. Performance of base- N PriBET protocol for a data size of 16384

Data size (bits)	Encoding size (N)	Block size (γ)	Plaintext space (t)	Ciphertext space (q)	Total computation time (ms)	Lattice dimension (n)	Security level
8	2^4	1024	2^{11}	61-bit	1469	2048	≥ 140
16		512			2953		
32		256			5860		
8	2^8	4096	2^{16}	71-bit	875	4096	
16		2048	2^{17}	73-bit	1454		
32		1024	2^{18}	75-bit	2844		
16	2^{16}	4096	2^{32}	103-bit	982		
32		2048	2^{33}	105-bit	1718		
64		1024	2^{34}	107-bit	3157		
32	2^{32}	8192	2^{64}	167-bit	1312	8192	
64		4096	2^{65}	169-bit	1937		

6.1 Parameters Settings

As discussed in Sect. 5 of [11], we selected proper values of the parameters (n, t, q, σ) of our used security scheme for successful decryption and to achieve a certain security level. In addition, we need to select the appropriate value for our encoding size N . The security analysis of this protocol is skipped due to page limitation which can be addressed in the full version of the paper.

Correctness Side. Here we show the correctness of our protocol for computing $ct(E_\lambda)$ for different lattice dimensions. According to the Lemma 1 in [11], the correctness of ciphertext $ct(E_\lambda)$ holds if

$$\|\langle ct(E_\lambda), s \rangle\| \leq q/2. \quad (11)$$

As mentioned in [14], we consider the upper bound Φ of ∞ -norm size $\|\langle ct, s \rangle\|_\infty$ for any fresh ciphertext $ct \in (R_q)^2$. In addition, the value of the upper bound Φ is $2t\sigma^2\sqrt{n}$ (see Theorem 3.3 in [5]). Here the ∞ -norm size of $ct(E_\lambda)$ in Eq. (10) is defined by the inequality as $\|\langle ct(E_\lambda), s \rangle\|_\infty < 2n\Phi^2 + 2n\Phi^2$ (see [14] for details). Furthermore, we take the value of Φ as $2t\sigma^2\sqrt{n}$ (see [5] for details). Now the inequality in Eq. (11) can be represented as $\|\langle ct(E_\lambda), s \rangle\|_\infty < 2n\Phi^2 + 2n\Phi^2 \approx 8n^2t^2\sigma^4$. The correctness for the inequality in Eq. (11) for the ciphertext $ct(E_\lambda)$ can be found if it satisfies

$$16n^2t^2\sigma^4 \leq q. \quad (12)$$

Chosen Parameters. Here we need the lattice dimension n to be greater than $k \cdot l_N$ for our protocol. Since we required to compute the SED between two base- N integer vectors of length l_N . Now the plaintext space t should satisfy the relation

$$t \geq l_N \cdot N^2. \quad (13)$$

As shown in Table 1, we consider encoding size $N = 2^4$ – 2^{32} for the lattice dimension 2048, 4096, and 8192 with the data size $k = 16384$. We also consider integer data size l to be 8-bit, 16-bit, 32-bit, and 64-bit for comparison in the base- N

PriBET protocol. Furthermore, we set t according to Eq. (13) for our plaintext space R_t . According to the work in [5], we choose $\sigma = 8$ and the value of q must be greater than $16n^2p^2\sigma^4$ for the ciphertext space R_q as in Eq. (12). Therefore, we fix our parameters as (n, t, q, σ, N) as shown in Tables 1 and 2. We did a block-wise computation to manage our dataset of 16384 integers within lattice dimension of 2048, 4096, and 8192. We set the block size γ as 256, 512, 1024, 2048, 4096, and 8192 for the lattice dimension of 2048, 4096, and 8192.

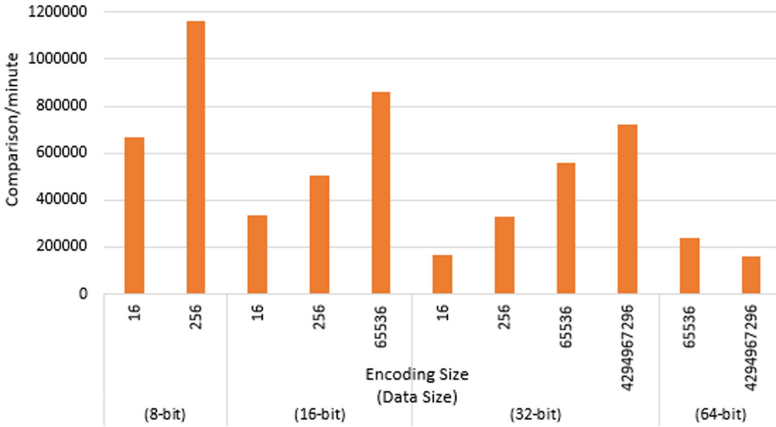


Fig. 1. Comparative performances of our protocol for different encodings (base-16, base-256, base-65536, and base-4294967296) using the lattice dimensions of 2048, 4096, and 8192 with 8-64-bit integers.

Security Level. In our experiment, we consider the security of the encryption scheme against two attacks namely distinguishing attack [8] and decoding attack [6]. According to the discussion of Lindner and Peikert [6], we consider every parameter setting to provide more than 128-bit security level to secure our protocol against the distinguishing attack and more powerful decoding attack with the advantage $\epsilon = 2^{-64}$. In addition, a root Hermite factor $\delta < 1.0050$ is required to achieve an 80-bit security level that was shown by Chen and Nguyen [2] in lattice-based cryptographic schemes. As discussed in [5], the running time t_{adv} is defined as $lg(t_{adv}) = 1.8 / lg(\delta) - 110$ where the root Hermite factor δ is expressed as

$$c \cdot q / \sigma = 2^{2\sqrt{n \cdot lg(q) \cdot lg(\delta)}} \tag{14}$$

As shown in Eqs. (12) and (13), both t and q should be increased with the increase of the encoding size N . If we use a low lattice dimension for a high encoding size, we will get security level less than 128-bit according to Eq. (14) which is not desirable. As shown in Table 1, if the encoding size N is 16 and lattice dimension is 2048 then we get a security level of 140. But if $N = 256$ and $n = 2048$ again then we get a security level 104 which is not acceptable for our case. So we increase the lattice dimension with the increase of encoding size to

get a better security level. According to data of Table 2 in [15], our parameters setting provides more than 140-bit security level to protect the security algorithm from some distinguishing attacks as shown Table 1.

6.2 Implementation Details

We implemented both Saha and Koshiba [11] and our protocols in C programming language with Pari C library (version 2.7.5) [13] and ran the programs on a single machine configured with 3.6 GHz Intel core-i7 processor and 8 GB RAM using Linux environment. Here we did two types of experiments. One is for selection of encoding parameter and another for comparative analysis with the existing method. To do these experiments we selected suitable values of our parameters for our security scheme in [11] and encoding technique described in Sect. 2 respectively. We considered maximum data size of 16384 with 8–64-bit integers for our experiments.

Table 2. Parameter settings of Saha and Koshiba protocol [11] and our protocol

Integer size (l)	Data size (k)	Encoding size (N)		Lattice dimension (n)		Plaintext space (t)		Ciphertext space (q)	
		Saha and Koshiba	Our method	Saha and Koshiba	Our method	Saha and Koshiba	Our method	Saha and Koshiba	Our method
8	4096	2	2^8	32768	4096	2048	2^{16}	69 bits	73 bits
16	4096		2^{16}	65536	4096		2^{32}	71 bits	105 bits
32	2048			65536	4096		2^{33}	71 bits	107 bits

6.3 Selection of Encoding Size (N) and Performance Towards Big Data

Table 1 shows the performance of our base- N PriBET protocol for the lattice dimension of 2048, 4096, and 8192 with a data size of 16384. Here we did the experiments for different values of our encoding size $N(2^4-2^{32})$. Furthermore, we show a comparative performance of our different encoding size for the lattice dimension of 2048, 4096, and 8192 as shown in Fig. 1. Also, we were able to select the value of our encoding size N as low as $2^4 = 16$ and as high as $2^{32} = 4294967296$. We tried to select the maximum value of encoding 2^{64} where the computation is out of the capacity our machine due to a buffer overflow. It also happens due to increasing the value of plaintext space t and ciphertext space q . From this figure, it is clear that batch equality comparison is faster if data size and encoding size are same for most of the cases. In addition, we achieved a good performance for the encoding size of $2^8 = 256$ and $2^{16} = 65536$ with the data size of 8 and 16-bit respectively. So we chose two effective values of base- N encoding size as 2^8 and 2^{16} .

Moreover, our experiments also showed that our protocol was able to do over 1.1 million and 862 thousand of equality comparisons per minute with the encoding size of $2^8 = 256$ and $2^{16} = 65536$ for a data size of 8-bit and 16-bit respectively. Moreover, our protocol was able to compute more than 700 thousand (resp. 200 thousand) equality comparisons per minute for 32-bit (resp. 64-bit) data with an encoding size of 2^{32} .

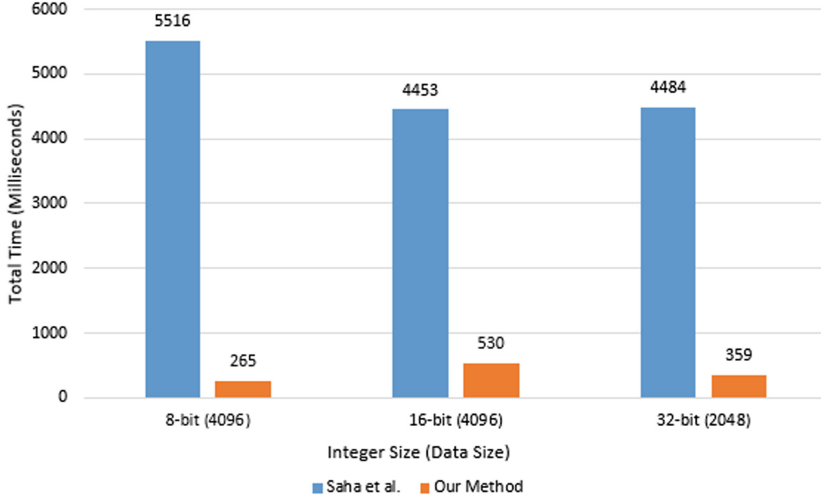


Fig. 2. Performance comparison between Saha and Koshiba [11] and our method for the data size 2048 and 4096 with length of 8–32-bit using base-256 and base-65536 encodings

6.4 Comparative Analysis

In this section, we show comparative performances of our protocol with respect to Saha and Koshiba protocol [11] for batch comparison to find out the equalities of an integer with a set of k integers. Saha and Koshiba used the Hamming distance computation for their PriBET protocol over binary encoding because Hamming distance computation works only for binary data. Here we used the base- N encoding to minimize the cost of computation by reducing lattice dimension. As mentioned in Sect. 2, we achieved the lattice dimension reduction by a factor of $\log_2 N$ than Saha and Koshiba which reflects in the parameter settings of lattice dimension for both of the protocols as shown in Table 2. Due to using base- N encoding, we use the SED computation to find the distance between a given query and an existing dataset. According to Sect. 6.3, we used the two best encoding size of 2^8 and 2^{16} for getting the better performance. Table 2 shows the used parameters settings for both of the protocols. Furthermore, we took the integers set of 2048 and 4096 with a practical bit size of 8-bit, 16-bit, and 32-bit for the comparison. For the data size of 2048 and 4096 using base-256 and

base-65536 encoding, the comparative performance of our protocol with respect to Saha and Koshiba protocol is shown in Fig. 2 where timing was taken in milliseconds. Our protocol showed the best performance than that of Saha and Koshiba for 8-bit integer comparison with a data size of 4096 and a less good performance for a 16-bit integer with the same data size. Overall, our protocol performed more than 8–20 times as fast as Saha and Koshiba protocol for the batch equality comparison. Besides, we achieved more than 140-bit security using our parameter settings described in Sect. 6.1.

7 Conclusions

In this paper, we discussed an efficient base- N fixed length encoding based Pri-BET protocol using ring-LWE based somewhat homomorphic encryption in the semi-honest model. For this purpose, we have shown a base- N fixed length encoding algorithm to reduce the cost of equality comparison. In addition, we experimented our protocol using different encoding size to find out the best value of our encoding size N . Our protocol was able to do more than 1.1 million (resp. 862 thousand) comparisons per minute for 8-bit (16-bit) integer batch comparison. Also, we have been able to show that our protocol works more than 8–20 times faster than the protocol of Saha and Koshiba. We also believe that this achievement of around a million of comparisons per minute is big a step towards big data processing. We hope that our research will inspire future researches to use base- N encoding rather than binary encoding for many computation purposes because of reducing the lattice dimension by a factor of $\log_2(N)$.

Acknowledgments. This work is supported in part by JSPS Grant-in-Aids for Scientific Research (A) JP16H01705 and for Scientific Research (B) JP17H01695.

References

1. 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). https://doi.org/10.1007/978-3-642-22792-9_29
2. Chen, Y., Nguyen, P.Q.: BKZ 2.0: better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 1–20. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_1
3. Couteau, G.: Efficient secure comparison protocols, Cryptology ePrint Archive, Report 2016/544 (2016). <http://eprint.iacr.org/2016/544>
4. Gentry, C.: Fully homomorphic encryption using Ideal lattices. In: Symposium on Theory of Computing - STOC 2009, pp. 169–178. ACM, New York (2009)
5. Lauter, K., Naehrig, M., Vaikuntanathan, V.: Can homomorphic encryption be practical? ACM Workshop on Cloud Computing Security Workshop. CCSW 2011, pp. 113–124. ACM, New York (2011)
6. 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). https://doi.org/10.1007/978-3-642-19074-2_21

7. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, pp. 1–23. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_1
8. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Bernstein, D.J., Buchmann, J., Dahmen, E. (eds.) Post-Quantum Cryptography, pp. 147–191. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_5
9. Rivest, R.L., Adleman, L., Dertouzos, M.L.: On data banks and privacy homomorphism. In: DeMillo, R.A., Dobkin, D.P., Jones, A.K., Lipton, R.J. (eds.) Foundations of Secure Computation, pp. 169–177. Academic Press, New York (1978)
10. Saha, T.K., Koshiha, T.: An enhancement of privacy-preserving wildcards pattern matching. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (eds.) FPS 2016. LNCS, vol. 10128, pp. 145–160. Springer, Cham (2017). <https://doi.org/10.1007/978-3-319-51966-1-10>
11. Saha, T.K., Koshiha, T.: Private equality test using ring-LWE somewhat homomorphic encryption. In: 3rd Asia Pacific World Congress on Computer Science and Engineering (APWConCSE), pp. 1–9. IEEE (2016)
12. Saha, T.K., Mayank, Koshiha, T.: Efficient protocols for private database queries. In: Livraga, G., Zhu, S. (eds.) DBSec 2017. LNCS, vol. 10359, pp. 337–348. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61176-1_19
13. The PARI~Group: PARI/GP version 2.7.5, Bordeaux (2014). <http://pari.math.u-bordeaux.fr/>
14. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiha, T.: Practical packing method in somewhat homomorphic encryption. In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boulahia, N., Foley, S., Fitzgerald, W.M. (eds.) DPM/SETOP 2013. LNCS, vol. 8247, pp. 34–50. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54568-9_3
15. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiha, T.: Secure pattern matching using somewhat homomorphic encryption. In: Proceedings of the 2013 ACM workshop on Cloud computing security workshop, pp. 65–76. ACM (2013)
16. Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K., Koshiha, T.: Secure statistical analysis using RLWE-based homomorphic encryption. In: Foo, E., Stebila, D. (eds.) ACISP 2015. LNCS, pp. 471–487. Springer, Springer (2015). https://doi.org/10.1007/978-3-319-19962-7_27
17. Yao, A.C.: Protocols for secure computations. In: 23rd Annual Symposium on Foundations of Computer Science, pp. 160–164. IEEE (1982)
18. Yi, X., Kaosar, M.G., Paulet, R., Bertino, E.: Single-database private information retrieval from fully homomorphic encryption. IEEE Trans. Knowl. Data Eng. **25**(5), 1125–1134 (2013)



Multi-level Access Control, Directed Graphs and Partial Orders in Flow Control for Data Secrecy and Privacy

Luigi Logrippo^{1,2}(✉)

¹ Université du Québec en Outaouais, Gatineau, Canada
luigi@uqo.ca

² University of Ottawa, Ottawa, Canada

Abstract. We present the view that the method of multi-level access control, often considered confined in the theory of mandatory access control, is in fact necessary for data secrecy (i.e. confidentiality) and privacy. This is consequence of a result in directed graph theory showing that there is a partial order of components in any data flow graph. Then, given the data flow graph of any access control system, it is in principle possible to determine which multi-level access control system it implements. On the other hand, given any desired data flow graph, it is possible to assign subjects and data objects to its different levels and thus implement a multi-level access control system for secrecy and privacy. As a consequence, we propose that the well-established lattice model of secure information flow be replaced by a model based on partial orders of components. Applications to Internet of Things and Cloud contexts are briefly mentioned.

Keywords: Security · Secrecy · Confidentiality · Privacy · Access control
Flow control · Mandatory access control · Multi-level security
Multi-layer security · Internet of things · Cloud

1 Introduction

We present the view that Multi-level (ML) access control methods, in the sense that will be defined here, have fundamental importance for access control, data secrecy and data privacy; in fact, any access control system that intends to provide secrecy and privacy must implement such methods. By using a result in directed graph theory, we show that data flow graphs representing data flow networks are partial orders of maximal strongly connected components. By generating the data flow graphs of access control systems, one can see what ML systems they implement. By appropriately assigning data to the strongly connected components of data flow graphs, one can implement ML data security and secrecy.

Note that our use of the term *data privacy* in this paper refers to accessibility of private data only. Other research, such as in *privacy by design*, has much wider motivations and requirements [4] and is usually concerned with making it impossible to identify personal information in data sets. Note also that the term *confidentiality* is often considered to be a synonym of *secrecy*. We subscribe to the view by which “*the fundamental nature of a*

privacy violation is an improper information flow” [14] and we propose new analysis and design methods to enable only proper flows. Data secrecy is a prerequisite to data privacy, and the latter will be implied in the rest of this paper.

In Sect. 2, we present some established concepts on ML systems. In Sect. 3, we present the mentioned result of directed graph theory. In Sect. 4 we show that it is possible in principle to find the ML model implicit in any access control system that can be represented by a data flow graph. In Sect. 5 we show how, given a desired data flow graph, it is possible to populate it with subjects and objects thus realizing secrecy-preserving data flows in concrete systems. In Sect. 6 we make a synthesis of our results, with recommendations.

2 Data Flow Control and Multi-level Access Control Methods

The study of data flows in access control networks was addressed, directly or indirectly, in many papers in the early years of research on access control methods [13]. Such research was based on the following main ideas:

- Distinction between *secure* or *legal* and *insecure* or *illegal* flows.
- *State-based*: following the famous Bell-La Padula model (BLP) proposed in 1973 [2, 3], it was usually assumed that models for secure information flow could be proved secure by reasoning in terms of state transitions, caused by reading and writing operations.
- *Lattice-based*: following an equally famous 1976 paper by Denning [6], it was usually accepted that secure data flows could be guaranteed by imposing a lattice-structure on the data flow. Entities should be placed in the nodes of a lattice and data should flow along the order relations of the lattice structure. So, much research was directed to ensuring such lattice structuring in information systems [9, 19, 23].

This research introduced models that implement both access control and flow control, with a single mechanism. These became known as the *Mandatory access control models (MAC)* [22], and are usually considered to include the ML methods. However MAC models seemed to be too restrictive for enterprise applications. Their realm of application is often considered limited to the military or to operating systems, and even there, with some relaxations. Subsequently, research moved on to flexible models capable of implementing in practice the access control needs of organizations, leading to the Role based access control model (RBAC) [8] and to the Attribute based access control model (ABAC) [11]. Of these, many variants exist but they are mostly conceived for access control and flow control requires further attention.

ML access control methods have been defined and used in the literature and practice in different ways [22, 24]. One of the best-known early proposal for such methods was the BLP access control model, whose goal is to ensure that in an organization data can move only upwards, from the less secret to the more secret levels. Many variants and generalizations of this concept have been proposed.

In this work, we react to the limiting view of ML system by demonstrating the opposite view that being ML is an intrinsic property of any data flow; so *secrecy must implemented according to this ML structure, failing which the system will not*

implement secrecy. That is, any data security system that is not designed according to the intrinsic ML structure of its data flow cannot implement secrecy. This holds for systems specified in RBAC or ABAC or other models. We will see that this view implies a significant correction to the view of ML structures as lattices.

We review briefly here other well-known concepts that lead to our conclusions, before presenting in the next section the graph-theoretical foundation for them.

In any data secrecy system, the following principles are generally accepted:

- (1) there are at least two types of data: the data to be protected (let us call them *secret*) and the rest (let us call them *public*); they are usually segregated to different databases.
- (2) there are at least two types of subjects: those that should be able to know secret data, and the others.

This creates a *two-level hierarchy* of data and subjects. The extension to hierarchies of *n-levels* is straightforward, and leads to the following well-known principles:

- (3) no read up: subjects at a given level of the hierarchy should be able to read at their own or lower levels only;
- (4) no write down: subjects at a given level of the hierarchy should be able to write at their own or higher levels only;
- (5) databases containing high secrecy data can also contain low secrecy data, but not vice-versa.

Further, the theory of non-interference [21] is also based on the existence of at least two levels of data secrecy.

Finally, in many organizations data are routinely classified according to sensitivity levels and personnel are classified according to clearance, with policies defining what clearance is necessary to read or write which data, given their sensitivity levels.

Therefore, ML methods are necessary for data secrecy, and also relate closely to practical needs.

The combination of state-based concepts and relational concepts (as in the lattice model) leads to complex proofs. In this paper, as in [15, 16], we use relational concepts only, while acknowledging that state-based concepts can be more expressive for modeling attacks [12].

3 Data Flow Digraphs as Partial Orders of Components

We use data flow graphs for abstract, relational views of data flows in systems. Data flow graphs are represented here as directed graphs, or *digraphs*. In our first presentation of the theory, nodes in our data flow digraphs are *entities* that will represent in a unified way the usual subjects and objects of access control systems. Edges between two entities represent the fact that data *can flow* between the two entities, e.g. if entity *A* is a subject and entity *B* is an object, then an edge from *A* to *B* means that *A can write* on *B*, while an edge from *B* to *A* means that *A can read* from *B*. This simple view enables us to present synthetically some results that can be adapted to several interpretations and contexts. We also take a pessimistic assumption, common in security

theory, by which, if any data at all can flow from A to B , then any other data of A can also flow to B . This leads to assuming the *transitivity* of the data flow relationship, i.e. if data can flow from A to B and also from B to C , then it can flow from A to C . The transitivity of data flows is a property that cannot be postulated in general [21], but, since it is based on the mentioned pessimistic assumption, cannot lead to systems that are under-protected. Finally, it is reasonable to assume the *reflexivity* of data flows.

In Fig. 1(a), taken from [1] we represent an arbitrary digraph, where the arrows can be interpreted to denote possible data flows among entities in a system, perhaps in an Internet of Things context. This digraph does not represent a partial order (thus of course not a lattice) because of the presence of symmetric relationships; however it is easy to see that it defines a data flow where all data can end up in entities L, M, N .

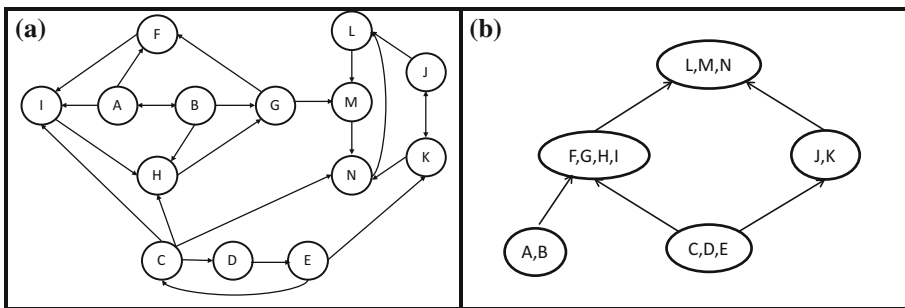


Fig. 1. A digraph showing allowed data flows in a network and its component digraph [1]

We see that entities A and B can send or receive data from each other. We conclude that A and B can share all data they have or, the data that one can originate or receive the other can also receive, so they can be considered to be one entity for access control purposes. We will speak of a *strongly connected component* $\{A, B\}$, which is also *maximal* because it is not part of a larger strongly connected component. Henceforth, for conciseness we will use the term *component* to denote a *maximal strongly connected component*. By the same reasoning, entities F, G, H, I can receive data from each others, and so they should be considered to form a component also. Proceeding in this way for the whole digraph, we detect the components $\{C, D, E\}$, $\{L, M, N\}$ and $\{J, K\}$. Since we have assumed transitivity, all the edges in a component can be thought of as bidirectional, and there is an implied bidirectional edge between F and H . Of course, there can be singleton components consisting of only one node.

Using this information, we can derive the *component digraph* of Fig. 1(a), shown in Fig. 1(b). We note that this second digraph preserves all the essential information of the first, except for the fact that components have been condensed into one node: symmetric relationships, which are equivalence relationships, have been encapsulated. Elementary results of digraph theory [1, 10] inform us that:

1. this construction is always possible and will always lead to an acyclic digraph, which represents a *partial order* because of the reflexivity and transitivity we have assumed;

- the component digraph has the same connectedness as the original one, in the sense that there is a directed path from X to Y in the original digraph iff there is such a path between components containing X and Y in the component digraph.

This leads us to conclude that *any data flow digraph can be understood as a partial order of components*.

For access control systems and flow control systems this result is very useful because the digraph of Fig. 1(b) shows more concisely the essential information in Fig. 1a. We can also assume that each entity can have some data of its own (we say that these data *originate* in the entity), which can be shared with other entities according to the data flow relationships. The digraph of Fig. 2 shows concisely how data can circulate in the original digraph. A comparison between Figs. 1(b) and 2 shows that the greater entities in the partial order can have available more data, also that data originating higher in the partial order can be available to fewer entities.

The nodes in the partial order of Fig. 2 can be thought of as *security levels* in a ML model.

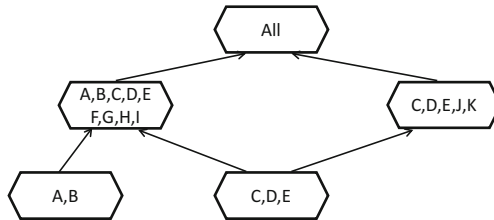


Fig. 2. The data flow digraph of the digraphs of Fig. 1.

We can use this information in several ways. For example, if entities in a node of Fig. 2 represent databases, we know that they can contain the same data and thus could perhaps be merged; if they represent subjects, then they can have the same *role* in an RBAC system; if they represent roles, they can perhaps be merged. Merging decisions however should be conditional to administrator’s approval because there may be reasons not to implement them. Also, the condensed digraph of Fig. 1(b) shows us how to reorganize the original digraph, see Fig. 3(a), where the original digraph is shown more explicitly as a partial order of components, where each component can again be thought of as a security level in a ML system. In Fig. 3(b) one further transformation has been done: only one edge between any two components has been selected, also relationships between components are implied when they can be derived by transitivity. This could be useful in practice if it is desired to place protection mechanisms in the edges that run between components. Note that there is some amount of arbitrariness in Fig. 3(b), for example instead, or in addition to, the edge $\langle A, I \rangle$ we could have had any edge from any of $\{A, B\}$ to any of $\{F, G, H, I\}$. But the transitive closures of the digraphs of Figs. 1(a) and 3, and of all possible digraphs similarly obtained, are the same, they all represent the same data flows.

Each component in these figures represents a set of entities where there can be complete data sharing, without any secrecy. But then data can also move to the next

component up in the partial order, if there is one. Data cannot move down in the partial order, and this implements secrecy. This is the way data circulates in ML networks, and so we define ML networks as partial orders of components, leading to the conclusion that any data flow digraph can be understood as a ML network.

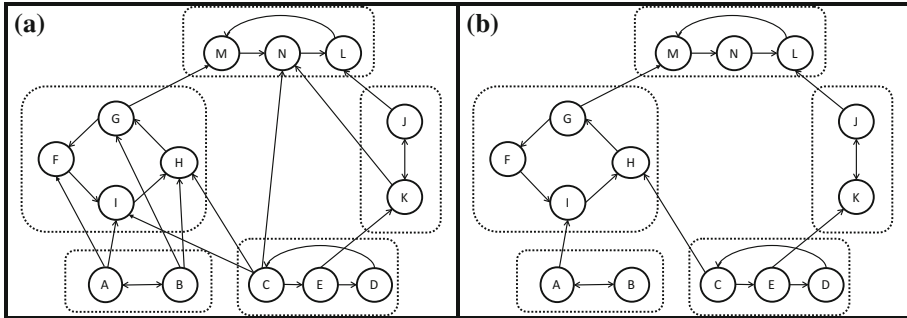


Fig. 3. The digraph of Fig. 1 reorganized and then simplified

Generic entities or subjects and objects can be associated with the nodes of Fig. 3 just as they were assumed to be in Fig. 1. The access control systems for these digraphs can be constructed in the following way:

- (1) data flow is permitted between any two elements of a component;
- (2) data flow is permitted between two elements of different components according to the partial order relationships represented by the paths in the original or derived digraphs, for example data can flow, directly or indirectly, from *B* to *N*.

Access control matrices will have to be constructed or roles with permission lists, or other policies. If the digraph must be implemented as a distributed network, then routing lists will have to be constructed. Encryption mechanisms can also be used to establish different data flows. Depending on the method used, the reduced number of edges in Fig. 3(b) might make the task easier. These are the same things that should be done to construct the access control system for the digraph of Fig. 1(a), however our construction has made it possible to see clearly the underlying partial order logic.

There are efficient algorithms to obtain component digraphs. For example, the time complexity of the well-known algorithm reported in [26] is linear on the number of edges plus the number of nodes.

It is interesting to observe that similar methods have a history of being used for data flow analysis in programs, where one of the main concerns is to identify the main components in the data flows [18].

4 Finding Levels in Existing Access Control Systems

Table 1 gives the permissions for a network with five subjects *S1* to *S5* and five objects *O1* to *O5*, using the notations *CR* for *can read*, and *CW* for *can write* [15, 16]. This an arbitrarily constructed network, and not one constructed to prove our conclusions.

Diagrams like this can be obtained for access control systems specified by means of access control matrices, RBAC permissions [20], etc.

Table 1. Read-write relationships for the network of Fig. 4

CR(S1, O1)	CW(S1, O2)
CR(S2, O2)	CW(S2, O3)
CR(S3, O4)	CW(S3, O3)
CR(S4, O3)	CW(S4, O3)
CR(S5, O2)	CW(S4, O4)
CR(S5, O5)	CW(S5, O5)

Figure 4 gives a digraph representation of this network, using ovals for subjects and rectangles for objects.

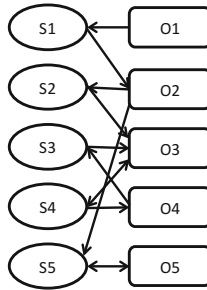


Fig. 4. An access control network

For uniformity and to justify transitivity, we can think that all edges represent a single transitive relationship *can flow* [20] rather than two distinct relationships as presented in Table 1. It remains that, in conformity with the concepts of access control, this is a bipartite digraph of *subjects* and *objects*.

By using the principles we have presented, the digraph of Fig. 4 can be shown as in Fig. 5(a). In Fig. 5(b) we see clearly the partial order of components implicit in Fig. 4. Using the terminology of [15, 16], from Fig. 5(a) it is clear that databases *O3* and *O4* can store the same data, thus possibly they can be merged. Subjects *S3* and *S4* also can know the same data, and so it is possible to give them the same role. Thus this view has implications for role engineering, but we will leave such considerations to future work.

Assuming that all data are in the objects or databases *O1–O5*, we show where the data of each of these databases can possibly be found in the network, by using the terminology ‘*Area of*’. We see that the data of *O1* can be available anywhere in the network, while the most secret data are the ones originating in objects *O3*, *O4* and *O5*, which can be known or stored in the most internal (or topmost) areas only. This may not be intended by the designers of the system of Table 1 or Fig. 4 but is a necessary consequence of the structure of the data flow.

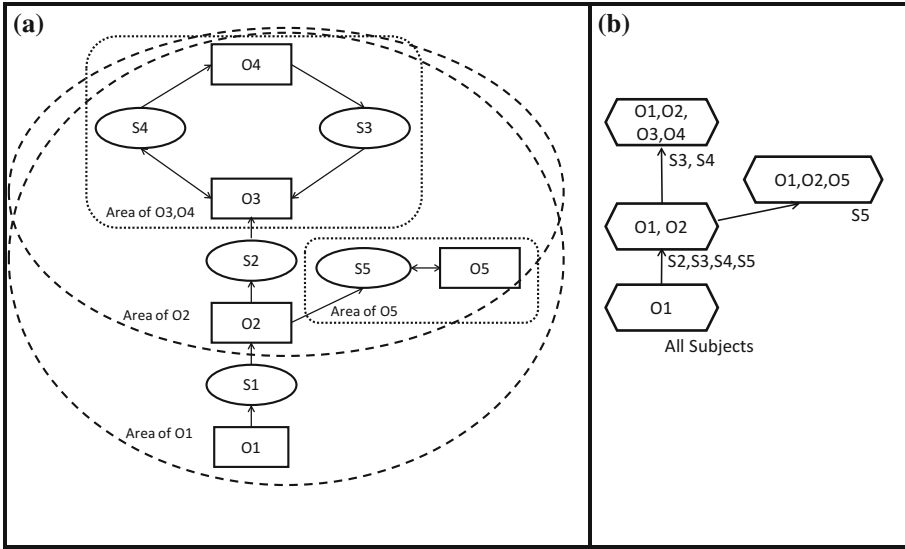


Fig. 5. Components and data flow for the example of Fig. 4

Because of the efficient algorithms we have mentioned, this analysis can be done in practice on systems of moderate size. Reference [25] presents this fact, analytically and by simulation.

5 Constructing Multi-level Systems

The previous discussion has not been helpful from the design point of view. In the example of the previous section, we have made some observations about the secrecy status of some data, but this was an observation on a randomly generated network of entities, it was not the result of design decisions. The initial representation of the system of Fig. 4 did not show clearly that the data in *O3*, *O4* or *O5* have the least visibility, thus are the most secret.

Once again, we will proceed by example. We wish to design an access control network for the following application, possibly in a Cloud context. We have two banks in conflict of interest, *Bank1* and *Bank 2*. *Bank1* has only one category of data, called *B1*, which it wishes to keep private. However *Bank 2* has public data labelled *B2P* that can be available to anyone, and secret data *B2S* that should be available only to its own employees. There is also a *Company 1* that collaborates with *Bank 2* and so shares all its data *C1* with *Bank 2*. However *Bank 2* does not want its secret data *B2S* to be known to *Company 1*, nor to *Bank 1* of course. Note that here we have added another type of entity, which we can call *organization*, and which will turn out to be a set of subjects and objects. Note also that we have expressed both *need to know* and *conflict* requirements. A Boolean analysis of these requirements leads to the data flow diagram shown in Fig. 6.

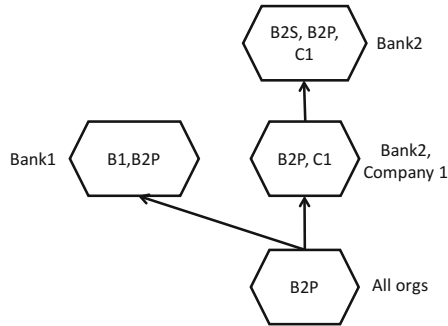


Fig. 6. Data flow in a hypothetical network

We now populate this data flow with subjects and objects, or employees and databases. This can be done in different ways. We will use a very simple structure with one database for each possible data contents and one employee for each database. We use the following notation: $Bob:\{B1, B2P\}$ means that employee *Bob* has clearance only to read the data of the types indicated, and similarly $Bk1:\{B1, B2P\}$ means that database *Bk1* can store only data of the types indicated. Taking *Bob* as an employee of *Bank 1*, in charge of the bank's database; *Alice* as an employee of *Bank 2* in charge of making available public data for *Bank 2* from a database that she administers for this purpose; *Carla* as an employee of *Company 1* and *Dave* as an employee of *Bank 2*, the populated diagram is shown in Fig. 7.

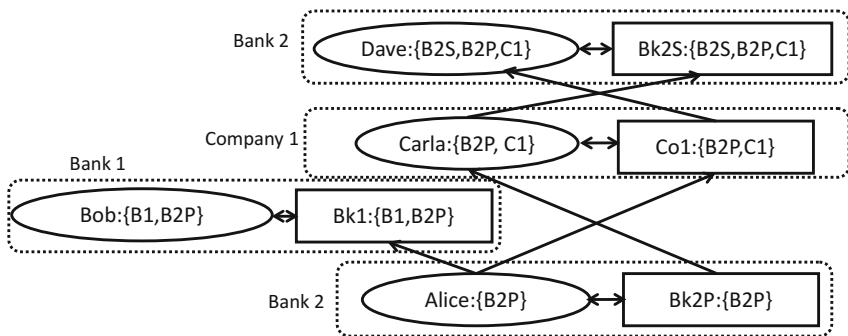


Fig. 7. A network of entities and organizations for the data flow of Fig. 6

Arrows that can be inferred by transitivity are not shown in Fig. 7, i.e. we can imagine that *Alice* is also authorized to write directly on *Bk2S*. This simplification can be considered to be inadequate from the security point of view, since in Fig. 7 the transfer of data from *Alice* to *Bk2S* depends on decisions by *Carla*. We have mentioned that transitivity cannot be given for granted in data flow systems. However our data flow diagrams show only the *possibility* of data flows, based on our pessimistic hypothesis.

Many other realizations of the original requirements are of course feasible, e.g. *Bank 1* may wish to keep separate *B1* and *B2P* data.

The classical BLP model can be obtained, in its essential aspects, as a special case of our construction. If we wish a BLP system with three levels: *Public*, *Confidential* and *Secret*, then the necessary labels are: $\{Public\}$, $\{Public, Confidential\}$, $\{Public, Confidential, Secret\}$.

This mechanism of constructing data flows by using label sets is powerful. We have seen above how it can be used to express conflicts. It can be used to express other types of constraints, but this is left to future papers.

6 Synthesis and Conclusions

In conclusion, using a basic result of digraph theory, we have established intuitively the following facts for access control and data flow systems that can be described as transitive, directed graphs:

- They define partial orders of components.
- These directed graphs and partial orders can be obtained efficiently from access control policies in some practical cases.
- No data secrecy is possible within a single component, since in each component, all entities can have available the same data.
- Data available in one component can also be available in the greater components in the partial order; data originating in one component cannot be available in lower components.
- As we move up in the partial order, the amount of data that can be available there will monotonically increase; also the number of entities that can have available data originating there will monotonically decrease.
- In order to have data secrecy, a system must have at least two components.
- Data secrecy can then be defined in terms of data being available only in *some* components.
- For data secrecy, data *must* be distributed among the components according to the desired levels of secrecy, with the most secret data in the top components of the partial order (from where they cannot move down). This will allow, *all* and *only*, *legal* or *secure* data flows.

While the *sufficiency* of some of these facts as principles for the design of data security systems has been understood for a long time, their *necessity* has been overlooked (except for the acceptance in theory of the lattice model, to be further discussed below). Any system that intends to protect data secrecy in this sense must implement appropriate partial orders of components; this is done by construction in strict BLP systems and similar ones, but must also be implemented in systems using other access control methods, such as RBAC or ABAC. Implementation can be done by using appropriate role assignments [19], policies, access control matrices, encryption or, in truly distributed systems, by using data forwarding policies.

As shown, these principles can be used not only for data protection within an organization, but also for networks of organizations (Sect. 4), in the Internet of Things and in Cloud environments where data of different ownerships coexist.

Some difficulties present themselves, of course.

A common objection against ML methods is that the constraint of allowing data flow in one direction only is impractical. However we have shown that all directed graphs describe unidirectional flows in their partial orders, and that this is necessary for secrecy. But this was based on the pessimistic assumption that when a flow is allowed between two entities, all data can move from one to the other by reading and writing operations. This view can be refined by distinguishing among types of data, limiting the operations to specific types of data and constructing different data flow digraphs, which means different partial orders, for different types of data. For example, in an organization we could have tables showing salaries with names, and tables showing salary statistics without names. Allowed data flows will normally be different for the two types of data. In the process called *sanitization* sensitive data can be transformed into less sensitive ones and *declassified*, with different data flow requirements [17]. Typically, salary tables with names could arrive at an office at the top of one partial order, and this office could produce statistics available for everyone, placing itself at the bottom of another partial order. Ideally, the office should be certified to produce such sanitization and declassification, or simply it should be an office that can be trusted not to divulge the secret data it might receive. Different data flow relationships for different types of data can be specified with any access control method if one supposes that different types of data are put in different objects. Access control systems with data labeling offer more flexibility [5, 17].

Another major difficulty is the fact that many modern access control systems do not define fixed data flows. These can change by administrative changes or environmental changes, leading to changes in the values of Boolean conditions. Graphs that describe such flows can be complex, with edges labelled by conditions. Changes must be conceived in a way that they do not modify essential partial order relationships. How to achieve this appears to be an interesting research topic.

Established theory considers lattices as the basic structuring model for secure data flows [6, 9, 23], however it seems that this view must be corrected. Lattices are restrictive, in the sense that they require the presence of joins and meets. Partial orders can be extended to lattices but in order to do so, unnecessary entities may have to be introduced. For example, to extend the partial order of Fig. 6 into a lattice, it is necessary to add a node containing both B1 and B2S, contradicting the requirements without any advantage; such a node must be excluded from the solution of Fig. 7. Further, to extend the partial orders of Fig. 1(b) (or Fig. 5(b)) to lattices it is necessary to add superfluous empty components that do not correspond to any entities. Lattices are also restrictive in the sense that they forbid symmetric relationships, which in our model are encapsulated in components. In [6] it is assumed that equivalent nodes can be merged, but in practice this may not be possible. In contrast, partial orders of components always exist in data flows that can be represented as digraphs, without any extensions.

Therefore, the ML model as outlined here should be seen as the obligatory design pattern [7] for systems intended to enforce strict data secrecy.

Finally, it should be mentioned that data flow theory has many aspects, by which our definitions can be considered to be very simplified. Still, this simplified view has led to results of practical significance for the analysis and synthesis of secrecy systems, as shown by our examples.

We have remained on an intuitive level, to avoid tying our discussion to a specific formalism. We are continuing work towards a suitable formalism to reason about secrecy properties [16], for which a first version was presented in [15].

Acknowledgment. This work was partially supported by a grant of the Natural Sciences and Engineering Research Council of Canada. The author is indebted to Sofiene Boulares and Abdelouadoud Stambouli for many useful discussions, and to Guy-Vincent Jourdan for useful comments on the draft copy.

References

1. Bang-Jensen, J., Gutin, G.Z.: *Digraphs: Theory, Algorithms and Applications*. Springer, Heidelberg (2010). <https://doi.org/10.1007/978-1-84800-998-1>. p. 17 and Fig. 1.12
2. Bell, D.E., La Padula, L.J.: *Secure computer systems: unified exposition and Multics interpretation*. TR MTR-2997 Rev. 1, Mitre Corporation (1976)
3. Bell, D.E.: Looking back at the Bell-La Padula model. In: 21st Annual IEEE Computer Security Applications Conference (2005, on line, no page numbers)
4. Cavoukian, A.: *Privacy by design. The 7 Foundational Principles*. White Paper, Information and Privacy Commissioner of Ontario, Canada (2009)
5. Damiani, E., De Capitani di Vimercati, S., Paraboschi, S., Samarati, P.: A fine-grained access control system for XML documents. *ACM Trans. Inf. Syst. Secur.* **5**(2), 169–202 (2002)
6. Denning, D.E.: A lattice model of secure information flow. *Commun. ACM* **19**(5), 236–243 (1976)
7. Fernandez-Buglioni, E.: *Security Patterns in Practice*. Wiley, Hoboken (2013)
8. Ferraiolo, D.F., Kuhn, D.R., Chandramouli, R.: *Role-Based Access Control*, 2nd edn. Artech House, Norwood (2007)
9. Foley, S.N.: Aggregation and separation as noninterference properties. *J. Comput. Secur.* **1**(2), 159–188 (1992)
10. Harary, F., Norman, R.Z., Cartwright, D.: *Structural Models: An Introduction to the Theory of Directed Graphs*. Wiley, Hoboken (1966). Chap. 3
11. Hu, V.C., Kuhn, D.R., Ferraiolo, D.F.: Attribute-based access control. *Computer* **48**(2), 85–88 (2015)
12. Jaume, M., Viet Triem Tong, V., Mé, L.: Flow based interpretation of access control: detection of illegal information flows. In: Jajodia, S., Mazumdar, C. (eds.) *ICISS 2011*. LNCS, vol. 7093, pp. 72–86. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25560-1_5
13. Landwehr, C.E.: Formal models for computer security. *ACM Comput. Surv.* **13**(3), 247–278 (1981)
14. Landwehr, C.E.: Privacy research directions. *Commun. ACM* **59**(2), 29–31 (2016)
15. Logrippo, L.: Logical method for reasoning about access control and data flow control models. In: Cuppens, F., Garcia-Alfaro, J., Zinicir Heywood, N., Fong, P.W.L. (eds.) *FPS 2014*. LNCS, vol. 8930, pp. 205–220. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17040-4_13

16. Logrippo, L.: A first-order logic formalism for access control and flow control, with application to multi-level access control. In preparation
17. Myers, A.C., Liskov, B.: Protecting privacy using the decentralized label model. *ACM Trans. Softw. Eng. Methodol.* **9**(4), 410–442 (2000)
18. Nielson, F., Nielson, H.R., Hankin, C.: *Principles of Program Analysis*. Springer, Heidelberg (2004). <https://doi.org/10.1007/978-3-662-03811-6>
19. Osborn, S.L., Sandhu, R., Munawer, Q.: Configuring role-based access control to enforce mandatory and discretionary access control policies. *ACM Trans. Inf. Syst. Secur.* **3**(2), 85–106 (2000)
20. Osborn, S.L.: Information flow analysis of an RBAC system. In: *Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, (SACMAT 2002)*, pp. 163–168 (2002)
21. Rushby, J.: Noninterference, transitivity, and channel-control security policies. TR CSL-92-02. Computer Science Lab., SRI International, Menlo Park, CA (1992)
22. Samarati, P., de Vimercati, S.C.: Access control: policies, models, and mechanisms. In: Focardi, R., Gorrieri, R. (eds.) *FOSAD 2000*. LNCS, vol. 2171, pp. 137–196. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45608-2_3
23. Sandhu, R.: Lattice-based access control models. *Computer* **26**(11), 9–19 (1993)
24. Smith, R.: Multilevel security. In: Bidgoli, H. (ed.) *Handbook of Information Security: Threats, Vulnerabilities, Prevention, Detection and Management*, vol. 3. Wiley, Hoboken (2005). Chap. 205
25. Stambouli, A., Logrippo, L.: Data flow analysis with access control matrices or RBAC permission lists. Submitted for publication
26. Tarjan, R.E.: Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**(2), 146–160 (1972)

Physical Security



Generation of Applicative Attacks Scenarios Against Industrial Systems

Maxime Puys^(✉), Marie-Laure Potet, and Abdelaziz Khaled

Université Grenoble Alpes, CNRS, VERIMAG, UMR 5104,
700 av. centrale, IMAG/CS-40700, 38058 Grenoble Cedex 9, France
{Maxime.Puys,Marie-Laure.Potet,Abdelaziz.Khaled}@univ-grenoble-alpes.fr

Abstract. In the context of security, risk analyzes are widely recognized as essential. However, such analyzes need to be replayed frequently to take into account new vulnerabilities, new protections, etc. As exploits can now easily be found on internet, allowing a wide range of possible intruders with various capacities, motivations and resources. In particular in the case of industrial control systems (also called SCADA) that interact with the physical world, any breach can lead to disasters for humans and the environment. Alongside of classical security properties such as secrecy or authentication, SCADA must ensure safety properties relative to the industrial process they control. In this paper, we propose an approach to assess the security of industrial systems. This approach aims to find applicative attacks taking into account various parameters such as the behavior of the process, the safety properties that must be ensured. We also model the possible positions and capacities of attackers allowing a precise control of these attackers. We instrument our approach using the well known model-checker UPPAAL, we apply it on a case study and show how variations of properties, network topologies, and attacker models can drastically change the obtained results.

1 Introduction

In the context of security, risk analyzes are widely recognized as essential. However, due to the extremely fast evolution of the state of the art of attacks, they need to be replayed frequently to take into account new vulnerabilities, new protections, etc. It is also often required for auditors to be able to replay risk analyses made by vendors in a certification process. Moreover, the increasing number of updates to apply encourages to replay both security and safety tests to ensure that new updates do not break the system. Thus, we need tools able to quantify the robustness of applications or to find attack scenarios. Furthermore, as a whole ecosystem is emerging around vulnerabilities and attacks, exploits can easily be found on internet, allowing a wide range of possible intruders from script-kiddies to governments including hacktivists, mafias, or terrorists organizations. Those attackers can present various capacities, motivations and

This work has been partially funded by the SACADE (ANR-16-ASTR-0023) project.

resources and can even collude together. Such differences must be taken into account when assessing the security of a system.

In this paper, we focus on industrial systems. Generally called SCADA, they control industrial processes such as electricity production, water treatment or transportation. Since those processes are usually critical, any incident can potentially harm humans and the environment. One of the most advertised attack was Stuxnet in 2010 [1] where a worm managed to sabotage a nuclear facility in Iran. This attack made people realize that a computer attack can have disastrous effects in the physical world. More recent attacks against these systems have been revealed in the past few years. For instance in 2014 against a German steel mill [2] where attackers managed to take control of a blast furnace or in 2015 in Ukraine [3] causing a massive power outage in winter.

Industrial systems are specific in various ways. First they want to ensure mainly availability and integrity while traditional IT systems often focus on confidentiality and authentication. Also the lifetime of their devices can vary between 20 to 40 years and they are really difficult to be updated in case of vulnerabilities. Industrial systems communicate over particular protocols which where not designed with security in mind. For example, MODBUS and DNP3 do not provide any security at all while a more recent communication protocol named OPC-UA includes the use of cryptography and has been show secure [4,5] (but currently rarely used in practice).

Related Work. Verifying the security of industrial systems have keep gaining in interest and various approaches were proposed since Byres et al. in 2004 [6]. In 2015, Cherdantseva et al. [7] performed a survey of 24 methods published between 2004 and 2014. They base their list on criteria such as the domain of application, the use of probabilities or not, the presence of case studies or if the method is implemented. Similar surveys have been released in 2012 by Piètre-Cambacédès and Bouissou [8], and in 2015 by Kriaa et al. [9]. We briefly summarize some of the works listed in these surveys either for their notoriety or for their closeness to our approach. In 2004, Byres et al. [6] propose a qualitative approach relying on attack trees to evaluate the security of industrial systems. Their approach is focused on systems communicating over MODBUS and targeting the electrical domain. In 2012, Kriaa et al. [10] present a method based on fault trees combined with Markov processes to model attacks on industrial systems. They implement this approach with the KB3 [11] tool and apply it to the Stuxnet attack. In 2015, they publish S-CUBE [12], an implementation of the former approach in the Figaro language. This approach takes into account the applicative logic of the process. In 2017, Rocchetto and Tippenhauer [13] present a method based on the cryptographic protocol verification tool CL-Atse [14]. They use the ASLAN++ language to model the industrial system and its applicative logic alongside with an augmented Dolev-Yao intruder, able to physically interact with the process [15].

Contributions. In this context, we propose an approach to assess the security of industrial systems. This approach aims to find what we call *applicative attacks*.

That is, considering an attacker that already exploited some security breaches to gain access to the system, we focus on finding what actions can he actually perform and what are the consequences on the industrial process. To find such attacks, we take into account various parameters such as the behavior of the process, the safety properties that must be ensured. We also model the possible positions and capacities of attackers allowing a precise and flexible control of these attackers. We implement our approach within the UPPAAL model-checker [16] to automate the discovery of attacks scenarios.

Outline. We first describe our global approach in Sect. 2. Then in Sect. 3 we detail how we instrument it using the UPPAAL model-checker. In Sect. 4, we apply the resulting framework on a concrete industrial example.

2 Context

In this section, we first detail how the analysis presented in this paper is included in a larger approach. Then we propose a case study and detail the parameters we will take into account.

2.1 The A²SPICS Approach

Our goal is to create a framework to detect applicative attacks against industrial systems. In this framework, industrial systems are modeled along with safety properties that they must ensure (e.g.: A furnace should not be started if its door is open). Then using formal methods such as *model-checking*, the model is analyzed in presence of intruders. In a later stage, found attacks could then be concretized into real networks packets that can be sent to a testbed representing the modeled system. Benefits are two-fold: besides being able to find applicative attacks, we can check if they are feasible and quantify their plausibility on the testbed.

In Fig. 1, we present the A²SPICS approach for *Applicative Attack Scenarios Production for Industrial Control Systems*. We focus on systems that respect safety properties in absence of attackers. In this context, we consider two phases of analysis. In the first phase (depicted in blue), we perform what we call an attack vector analysis [17]. It is a risk analysis in terms of security aiming to model attackers. It differs from well-known risk analysis methods such as EBIOS or MEHARI [18, 19] since they are focused on the assets to protect and the threats they face. Our risk analysis method relies on the topology of the system and the security features of communication protocols and produces what we call *attacker models*. Such models consists in placing possible attackers in the topology alongside as their capacities. For instance, if a protocol between two devices is considered secure, then no attacker is placed on this network channel. Similarly, if the protocol provides authentication but neither confidentiality nor freshness of messages, then we can place an attacker that can listen and replay messages. This first analysis thus allows us to place attackers in the network and

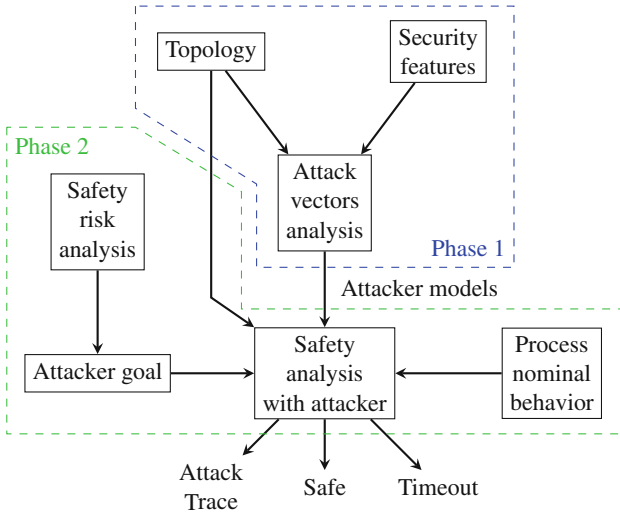


Fig. 1. The A²SPICS approach (Color figure online)

choose their capacities according to their objectives and the security features of the communication protocols. In a second phase (depicted in green), we take advantage of the fact that industrial systems are usually well analyzed in terms of safety. Thus, we consider as attacker goals the negation of a subset of the properties that the system has to ensure, resulting of these safety risk analyses. Then, based on the nominal behavior of the system, we are able to conclude if the safety properties can be jeopardized by the attackers. This second phase is the one presented in this paper.

2.2 Case Study

To illustrate our approach and show its validity, we will apply it on a case study along this paper. We choose as example a bottle filling factory taken from the VirtualPlant simulator¹. This simulator, designed by Jan Seidl, aims at providing a process simulator for experimentations. Empty bottles are carried by a conveyor belt. A sensor tells when a bottle is positioned under a nozzle which then pours liquid into the bottle. A second sensor detects when the bottle is full and then tells the nozzle to close and the conveyor belt to move until the next bottle is in place. Finally, a client can start and stop the whole process. Regardless of the communication protocol used, messages sent by the clients to the servers are read or write requests followed by read or write responses from the server of the form:

$$\begin{aligned}
 C &\rightarrow S : \text{READ}, \text{variableToRead} \\
 S &\rightarrow C : \text{READ}, \text{variableToRead}, \text{valueRead}
 \end{aligned}$$

¹ <https://github.com/jseidl/virtuaplant>.

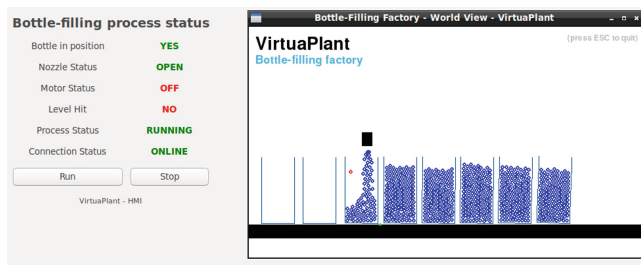


Fig. 2. VirtualPlant simulator

And respectively for write requests and responses:

$$C \rightarrow S : WRITE, variableToWrite, newValue$$

$$S \rightarrow C : WRITE, variableToWrite, writeSuccessOrNot$$

Figure 2 shows a synoptic view of the bottle factory process from the VirtualPlant process simulator. Although this example is quite simple, it allows a wide variety of instantiations. First, several properties to guarantee can be expressed: (i) bottles must leave the factory full, (ii) liquid should not be spilled out of bottles, (iii) the conveyor belt should start when a bottle is full, etc. Different topologies of the network controlling the process can also be studied. We can consider the conveyor belt and the nozzle as two distinct components. They could both be controlled by a single server (as shown in Fig. 3) or they can each be controlled by a individual server. Moreover, the communication protocols used in the network can present different levels of security allowing more or less powerful attackers. Even the positions of attackers can be considered. It can for instance be positioned on a network channel as a Man-In-The-Middle or as a corrupted client or server (e.g.: a legitimate device infected by a virus).

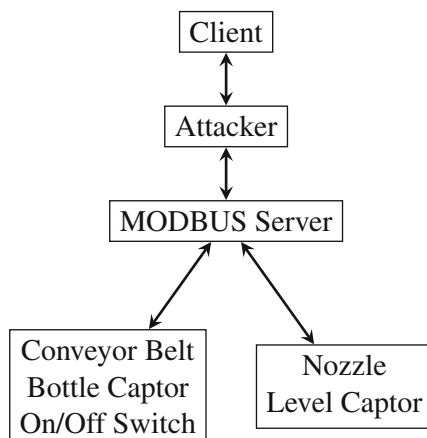


Fig. 3. Example of topology

2.3 Parameters of the Model

Our model is composed of various parameters including different entities communicating together:

Process. The process is the industrial application controlled by the system. It can for instance describe electricity production, liquid treatment or transportation. It is composed of a set of variables \mathcal{V}_P linked together by an automaton \mathcal{B}_P . We denote this automaton as the behavior of the process.

Clients. The clients \mathcal{C} are used to send commands to monitor and modify the process. They manage a set of variable $\mathcal{V}_c \subseteq \mathcal{V}_P, \forall c \in \mathcal{C}$ and a behavior $\mathcal{B}_c, \forall c \in \mathcal{C}$ determining which command they send and how they react to responses sent by the servers.

Servers. The servers are receiving commands sent by clients and applying them to the process. The security of the communication channel they use is determined by the protocol they implement (e.g.: MODBUS or OPC-UA). They also manage a set of variables $\mathcal{V}_s \subseteq \mathcal{V}_P, \forall s \in \mathcal{S}$.

Properties. The safety properties Φ to check on the system in presence of possibly active intruders are logical predicates (e.g.: CTL [20] temporal logic properties) on variables from \mathcal{V}_P .

Attackers. The attackers \mathcal{A} are possibly active intruders aiming to violate the safety properties from Φ . Their position in the network determines the clients and servers they will be able to communicate with while their capacities determine what type of action they will be able to perform (e.g.: intercept a message, encrypt a message, etc.). Depending on their capacities, attackers can also possess their own knowledge.

Topologies. We denote as components all clients, servers and attackers. We also denote the network channels linking these components as network topology of the system.

3 Implementation in UPPAAL

In this section, we describe how we deploy our approach in the UPPAAL model-checker [16]. We first show how to model the system. Then we detail the attackers we consider and finally the specifications of the safety properties.

3.1 Framework Architecture

Figure 4 depicts the overall architecture of our framework. It contains three components: (i) the system's model, (ii) the attacker models, and (iii) the specification of the the safety properties. Several models are already predefined as templates in a library we provide to the user (including clients, servers, attackers, security primitives, etc.). Thus, the user is only required to provide the topology of the system using templates from the library and behaviors of clients and servers.

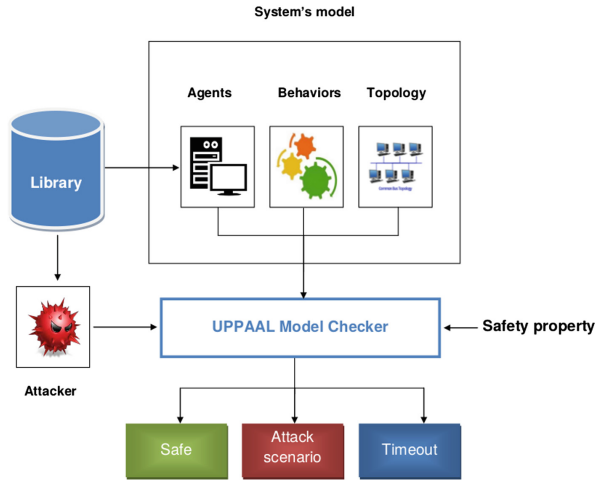


Fig. 4. Framework architecture

3.2 The System's Model

In UPPAAL, we model the components interacting with attackers as a composition of timed automata. Clients can create, send requests and receive responses while the server can receive requests, send responses and execute actions according to the clients' requests. Attacker act as *Man-In-The-Middle* intruders and have different capacities depending on the configuration. Among those capacities, they can listen to the network, stop, forge, replay or modify some messages according to its knowledge.

In our framework, we model six automata named: *Client*, *BehaviorClient*, *Server*, *BehaviorServer*, *SecureData* and *Attacker*. They access global variables such as cryptographic keys, messages exchanged over channels², as well as the system variables $V_{\mathcal{P}}$. According to Sect. 2.2, commands are formatted using the data structure $\langle cmdType, variable, value \rangle$ where:

- *cmdType* is a constant that expresses the purpose of the command (e.g.: read or write);
- *variable* is a constant denoting the different variables of the system;
- *value* is a the value of *variable* when needed by the command (for instance the new value of the variable in a write request or the value read in a read response).

To send a message, the *Client* automaton first asks the *BehaviorClient* automaton to obtain the applicative content he will send. Then, in the case of a client with using a secure communication protocol, the message will by signed

² In UPPAAL, messages are not exchanged directly on channels. Instead signals are sent telling processes to access messages as global variables.

and/or encrypted using the *SecureData* automaton. Concerning the *Server* automaton, it waits for a message sent by the *Client* automaton. When received, if the server implements a secure protocol, it decrypts the message and/or checks the message signature. Then, depending on the type of message (read/write), it either writes the new value of the variable addressed or reads its current value. Either way, the server creates and sends a response to the client according to the security of the request.

The *SecureData* automaton is used to manage security operations (encryption, decryption, signature, verification, etc.) according to cryptographic keys known by each component (including attackers).

3.3 Attackers

We consider four attackers with different capacities, each modeled as an automaton. Attacker A_1 (shown in Fig. 5a), based on the Dolev-Yao model [21], can listen to the network, stop, forge, replay or modify messages according to its knowledge. Such attacker is often considered as extremely powerful [22] making him really suited to prove absence of attacks but less realistic when considered within a vulnerability analysis. In Fig. 5a, the execution of the state diagram of

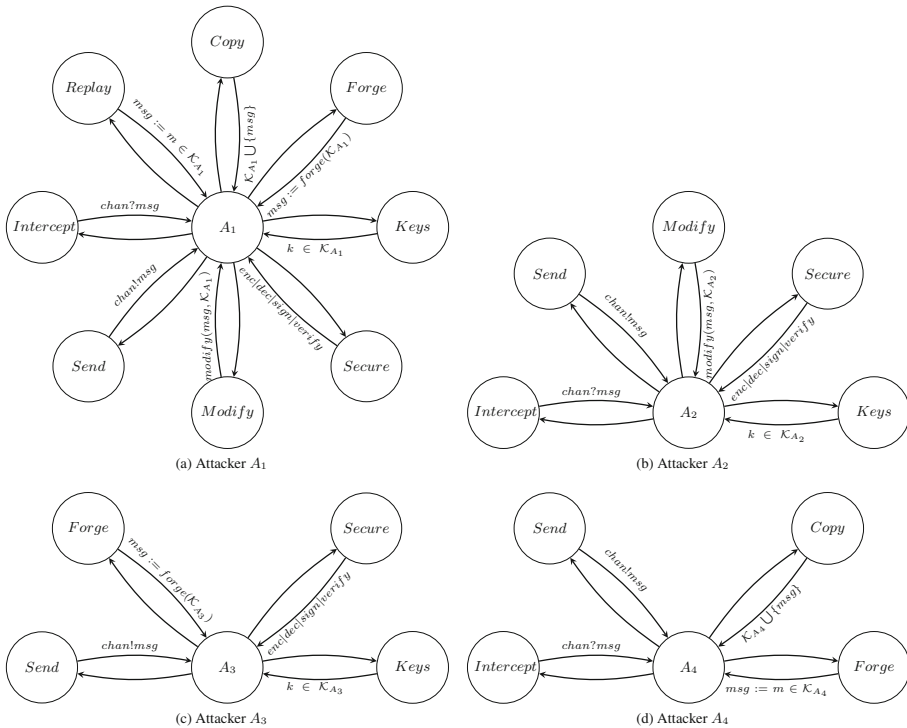


Fig. 5. Attackers considered

the attacker begins in state A_1 where the attacker can choose the action it can execute where:

- *Intercept* allows the attacker to intercept a message msg sent by a client or a server on some channel $chan$;
- *Send* allows the attacker to send a message msg to a client or a server on some channel $chan$;
- *Copy* allows the attacker to memorize a message msg into its knowledge \mathcal{K}_{A_1} ;
- *Keys* allows the attacker to retrieve cryptographic keys from its knowledge \mathcal{K}_{A_1} ;
- *Secure* allows the attacker to perform cryptographic operations according to its knowledge on the keys;
- *Forge* allows the attacker to create a new message msg from its knowledge \mathcal{K}_{A_1} ;
- *Modify* allows the intruder to modify an intercepted message msg according to its knowledge \mathcal{K}_{A_1} ;
- *Replay* allows the attacker to replay a message msg from its knowledge \mathcal{K}_{A_1} .

Capacities *Modify* and *Replay* could be seen as special cases of *Forge* in the sense that modifying a message is the action of forging a message at the time where a legitimate message is intercepted rather than sending a message at any time. Similarly, replaying a message can occur at any time but restricts the set of possible messages to the one previously memorized. Attacker A_2 (shown in Fig. 5b) is a subset of A_1 which can only to modify messages or parts of messages. To be more realistic, it can for example be limited to only modify the variable and value fields in order to not transform a read message into a write and vice versa. Such attacker would represent an attacker that want to avoid coarse attacks to be discrete. Attacker A_3 (shown in Fig. 5c) is a subset of A_1 which can only to forge new messages according to its knowledge. Thus it can be used to model a blind attacker that is not able to wiretap communications. Finally attacker A_4 (shown in Fig. 5d) is a subset of A_1 which can only to replay messages after memorizing them in its knowledge.

3.4 Safety Properties

To specify the properties, UPPAAL uses a simplified version of CTL that is expressed by the following syntax.

$$\Phi ::= A\Box\Phi \mid E\Diamond\Phi \mid E\Box\Phi \mid A\Diamond\Phi \mid \Phi \rightarrow \Phi \mid \neg\Phi$$

$A\Box\Phi$ means that Φ should be true on all paths in all reachable states. $A\Diamond\Phi$ means that Φ should be eventually true on all paths. $E\Box\Phi$ means that there exists a path where Φ is true in all reachable states. $E\Diamond\Phi$ means that there exists a path where Φ is eventually true. Symbols \rightarrow and \neg denote the implication and the negation propositional logic operators, respectively. To model safety properties we will only rely on $A\Box\Phi$.

4 Case Study

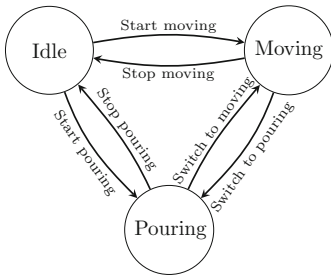
In this section, we illustrate our approach with the example described in Sect. 2.2. We show how we implemented it in the UPPAAL model-checker and we discuss the results we obtained by composing various attackers and topologies.

4.1 Behaviors

As described in Sect. 2.2, our case study is a bottle filling factory. Empty bottles are carried by a conveyor belt. A sensor tells when a bottle is positioned under a nozzle which then pours liquid into the bottle. A second sensor detects when the bottle is full and then tells the nozzle to close and the conveyor belt to move until the next bottle is in place. A client can start and stop the whole process. In this example, the process is composed of five boolean variables:

$$\mathcal{V}_{\mathcal{P}} = \{motor, nozzle, levelHit, bottleInPlace, processRun\}$$

They respectively denote the conveyor belt (*motor*), the nozzle (*nozzle*), the liquid level sensor (*levelHit*), the conveyor belt sensor (*bottleInPlace*) and the process on/off switch (*processRun*). Figure 6a shows an automaton describing the behavior of the process while Fig. 6b details the transitions of the automaton. Three states are considered: *Idle* means that the process is stopped, *Moving* that the conveyor belt is moving to position the next bottle and *Pouring* that the nozzle is filling a bottle. Each transition is labeled with two predicates: the guard and the output. The client will only start and stop the whole process when it wants³. Thus the variables that can be accessed by the client are $\mathcal{V}_c = \{processRun\}$.



(a) Process' behavior automaton

Current state	Next state	Guard	Actions
Idle	Moving	$processRun = true \wedge$ $bottleInPlace = false$	$motor := true$
Idle	Pouring	$processRun = true \wedge$ $bottleInPlace = true$	$nozzle := true$
Moving	Pouring	$bottleInPlace = true$	$motor := false \wedge$ $nozzle := true$
Pouring	Moving	$levelHit = true$	$motor := true \wedge$ $nozzle := false$
Moving	Idle	$processRun = false$	$motor := false \wedge$ $nozzle := false$
Pouring	Idle	$processRun = false$	$motor := false \wedge$ $nozzle := false$

(b) Details of the transitions

Fig. 6. Behaviors considered

The safety properties we want the process to guarantee would be a subset of properties considered as critical, resulting from a risk analysis in safety. For this case study, we exhibit the following properties, expressed as CTL formulas.

³ This models the actual behavior of the client in VirtualPlant and is not a limitation of our approach.

- Φ_1 : The nozzle opens only when a bottle is in position (i.e.: at all time and on all possible execution traces, *nozzle* is never true if *bottleInPlace* is false).
 $A\Box \neg(\text{nozzle} = \text{true} \wedge \text{bottleInPlace} = \text{false})$
- Φ_2 : The motor starts only when a bottle is full (i.e.: at all time and on all possible execution traces, *motor* is never true if *levelHit* is false).
 $A\Box \neg(\text{motor} = \text{true} \wedge \text{levelHit} = \text{false})$
- Φ_3 : The nozzle opens only when the motor stops (i.e.: at all time and on all possible execution traces, *nozzle* is never true if *motor* is true).
 $A\Box \neg(\text{nozzle} = \text{true} \wedge \text{motor} = \text{true})$

4.2 Network Topologies

We consider two network topologies T_1 and T_2 . In topology T_1 , a single server s_{MODBUS} using the MODBUS protocol controls both the conveyor belt and the nozzle. A single client c communicates with s_{MODBUS} . The MODBUS protocol is among the most used in industrial communications and does not provide any security at all. This topology is presented in Fig. 7a with:

- Set of servers $\mathcal{S} = \{s_{MODBUS}\}$ with:
 - Variables $\mathcal{V}_{s_{MODBUS}} = \mathcal{V}_{\mathcal{P}}$
- Set of clients $\mathcal{C} = \{c\}$ with:
 - Variables $\mathcal{V}_c = \{\text{processRun}\}$

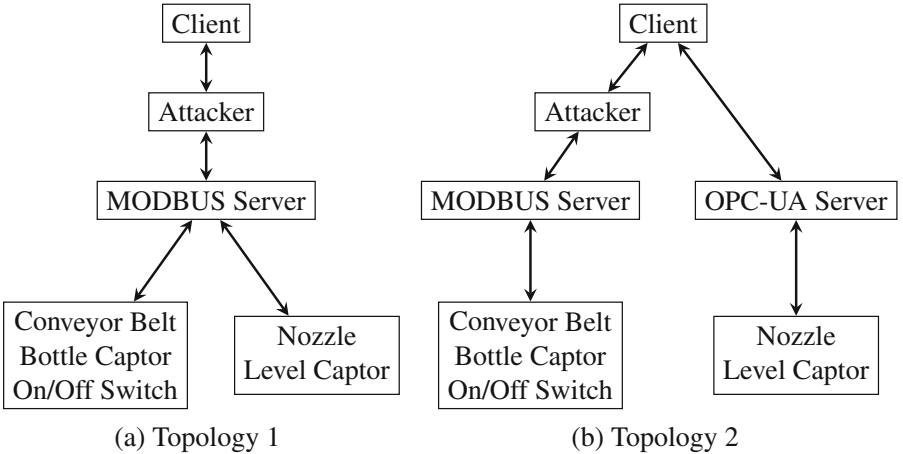


Fig. 7. Topologies considered

In topology T_2 , the conveyor belt and the nozzle are each be controlled by a individual server. The first server s_{MODBUS} communicates using MODBUS and controls the conveyor belt, the position sensor, and the on/off switch. The second server s_{OPC-UA} communicates using OPC-UA and controls the nozzle and

the level sensor. OPC-UA provides three security modes: *None*, *Sign* and *SignEncrypt*. Security mode *None* does not provide any security. According to Puys et al. [4], security mode *Sign* adds cryptographic signatures and provides authentication, integrity and freshness of communications and mode *SignEncrypt* also adds encryption providing confidentiality. We suppose that security mode *SignEncrypt* is used in our second topology, thus the attacker is not able to interfere with the channel between the client c and the OPC-UA server s_{OPC-UA} . This topology is presented in Fig. 7b with:

- Set of servers $\mathcal{S} = \{s_{MODBUS}, s_{OPC-UA}\}$
 - Variables $\mathcal{V}_{s_{MODBUS}} = \{processRun, motor, bottleInPlace\}$
 - Variables $\mathcal{V}_{s_{OPC-UA}} = \{nozzle, levelHit\}$
- Set of clients $\mathcal{C} = \{c\}$
 - Variables $\mathcal{V}_c = \{processRun\}$

4.3 Attackers

To demonstrate the modularity of our framework, we test both topologies against the four attackers proposed in Sect. 3.3. We recall the capacities of each attacker in Table 1 where ✓ means that the attacker has the capacity.

Table 1. Summary of capacities for each attacker

Attacker	Modify	Forge	Replay
A_1	✓	✓	✓
A_2	✓	✗	✗
A_3	✗	✓	✗
A_4	✗	✗	✓

4.4 Results Obtained Using UPPAAL

After experimenting different settings in UPPAAL, we chose to apply *Breadth first search* algorithm and to represent the states as *DBM (Difference Bounded Matrices)*. The results are summarized in Table 2 where ✓ means an attack has been found and ✗ means that the property is safe as well ◆ means that UPPAAL could not conclude. This happened because the tool was requesting more memory than available. Our experiments were run on a Intel(R) Core(TM) i5-4590 CPU @ 3.30 GHz with 16 GB of RAM. Times of analysis can be found and discussed in Sect. 5.1.

In theory, none of the four attackers can violate property Φ_1 in topology T_2 . The reason is that the OPC-UA server controls the *nozzle* variable, preventing any attack on this variable. Even with the MODBUS server controlling the *bottleInPlace* variable, if *bottleInPlace* is forced to *false* by an attacker while *nozzle* is *true*, then *nozzle* will automatically switch to *false* due to the process

Table 2. Results obtained

		A_1	A_2	A_3	A_4
T_1	Φ_1	✓	✓	✓	✗
	Φ_2	✓	✓	✓	✗
	Φ_3	✓	✓	✓	✗
T_2	Φ_1	◆	◆	✗	✗
	Φ_2	✓	✓	✓	✗
	Φ_3	✓	✓	✓	✗

behavior (and vice versa). Thus, the only way to break Φ_1 that is to force opening the *nozzle* which is not possible in topology 2 (as we can see with attackers A_3 and A_4). Similarly, attacker A_4 cannot violate any property, since the messages transmitted between the client and the server are only relative to start or stop the process.

Figure 8 shows the attack scenario found by UPPAAL with attacker A_2 against Φ_2 in topology T_2 . The client sends a message to the MODBUS server to start the process, the motor starts and the bottles advance on the conveyor belt. After some time, the client sends a message to stop the process. The attacker intercepts the message and modifies both the variable targeted by the write request and the new value to force the motor to start. This experimentation shows that we do not need the whole power of Dolev-Yao to find attacks. It also helps to find which are the capacities needed by an attacker to perform attacks. Thus, it allows tailored proofs of robustness resulting of a risk analysis.

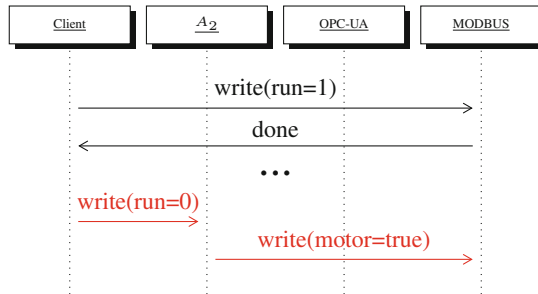


Fig. 8. Attack scenario with A_2 against Φ_2 in topology T_2

5 Discussions

In this section, we discuss the times taken for each analysis. We then compare our approach to related works presented in Sect. 1 and address some limitations and hypotheses we made.

5.1 Discussion of Analysis Timings

According to Tables 2 and 3, attacker A_2 obtains the same results as A_1 (Dolev-Yao) in shorter time. Attacker A_3 takes a bit longer but is able to conclude on property Φ_1 in topology 2 while attackers A_1 and A_2 cannot due to the system being out of memory.

These results show that really powerful intruders such as Dolev-Yao are often too complex and only parts of them are sufficient to find attacks. Such intruders are however preferred when trying to prove the absence of attacks. On the other hand, attacker A_4 obtains larger times which can be surprising since it is the simplest of our attackers. A likely explanation is that since all of his results are absence of attacks, UPPAAL must explore every possible state which can take way longer than finding a counter example.

Table 3. Verification times

		A_1	A_2	A_3	A_4
T_1	Φ_1	0.43 s	0.07 s	1.05 s	0.84 s
	Φ_2	0.52 s	0.10 s	0.69 s	0.35 s
	Φ_3	0.47 s	0.04 s	0.37 s	0.42 s
T_2	Φ_1	Out of memory		601 s	31.55 s
	Φ_2	0.66 s	0.23 s	2.17 s	35.20 s
	Φ_3	0.78 s	0.21 s	2.35 s	34.85 s

5.2 Comparing to State-of-the-Art

Our approach differs from most of the works presented in [7–9] that look more like risk analysis methods such as EBIOS [18, 19] for security or FMEA [23] for safety. It is typically the case of Byres et al. [6] who quantifies criteria such as likelihood or severity on a scale of four values. Moreover, 18 out of the 23 approaches listed in Cherdantseva et al. [7] are quantitative (i.e.: probabilistic) and thus require an initial distribution of probabilities to work. Nevertheless, a lot of these approaches give very few details on the source of these probabilities and their trustworthiness. It is also hard to evaluate the impact of variations of these probabilities. These approaches have however the advantage to quantify the likelihood and severity of resulting attacks. In [12], Kriaa et al. define four criteria to classify approaches combining security and safety:

1. analyzing formal models;
2. being both qualitative and quantitative;
3. being automated;
4. being adaptable to different assumptions.

Kriaa et al. also list some related works and conclude that none validate the automation criterion. In our case, the A²SPICS approach respects criteria 1, 3 and 4 (relying on a formal and automated verification tool, UPPAAL and allowing to simply change attacker’s positions and capacities as well as behaviors). To the best of our knowledge, the closest related work to the A²SPICS approach from Rocchetto and Tippenhauer [13] which also seem validates criteria 1, 3 and 4. Our approach shows nevertheless key differences with it, particularly in terms of considered attackers. Using cryptographic protocol verification tools such as CL-Atse allows to not require to model the attacker which is hardcoded in the tool making the Dolev-Yao attacker difficult to restrict. In their work, Rocchetto and Tippenhauer strengthen it by adding equational theories (allowing to handle physical interactions with the process [15]). We aim to focus on attackers resulting of a risk analysis which are often less powerful than Dolev-Yao. Moreover, to the best of our knowledge, Rocchetto and Tippenhauer do not take into account the network topology of the system, although it seems possible in ASLAN++. It means in their case that all agents (or multiple groups of agents) communicate over one unique channel accessible to the attacker, which is again not very realistic.

5.3 Discussion of Limitations and Hypotheses

Similarly to [13], we consider that time is discretized (i.e.: expressed as steps of execution). The state of the process is also discretized (e.g.: the bottle is either empty or full). Moreover, due to the complexity of attackers A_1 , A_2 , and A_3 , we have to bound the number of actions they can perform in an attack. This limit of the number of action being configurable. This is a classical limitation of model-checking approaches that will not terminate if the model can loop infinitely. Moreover, an under-approximation of the approach can lead to some attacks not being found and robustness not being established. In the results showed in Table 2, we pointed that property Φ_1 was never violated. This is due to the fact that two states of the system can be considered: (i) the real state (i.e.: if a bottle is physically present or not), and (ii) the logical state (i.e.: if the variable *bottleInPlace* is set to *true*). It appears that when a captor is modified by the intruder, then a decorrelation is introduced between these two state (in logical state, a bottle could be present while it is not the case in reality). However, properties are checked by UPPAAL on the logical state meaning possibly missing attacks (in particular for property Φ_1). This is a classical limitation due to the fact that we model the system without taking into account the physical environment.

6 Conclusion

We provided a modular approach to assess the security of industrial control systems. This approach aims to find applicative attacks taking into account different parameters such as the behavior of the process, the properties that an

attacker can aim to jeopardize, as well as the possible positions and capacities of attackers. We show how this approach can be implemented using the UPPAAL model-checker. We apply it on an example and show how variation of properties, network topologies, and attackers can change the obtained results. We also discuss key difference with approaches relying on protocol verification tools. Even when considering all possible variations of our example, it remains very simple. Still, the timing results we obtained encourage us to address the question of scalability. In the future, we would be interested into studying how to address the limitation pointed in Sect. 5.3. It would be useful to apply our approach to the case study proposed by Rocchetto and Tippenhauer to obtain a concrete comparison of the two approaches. We are also interested into modeling possible collusion between intruders so they can share knowledge and synchronize during attacks. Finally, we aim to generalize the implementation and build an open-source tool to automatically generate UPPAAL models and interpret the results.

References

1. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. *IEEE Secur. Priv.* **9**(3), 49–51 (2011)
2. Lee, R.M., Assante, M.J., Conway, T.: German steel mill cyber attack. *Industrial Control Systems*, 30 (2014)
3. Lee, R.M., Assante, M.J., Conway, T.: Analysis of the cyber attack on the Ukrainian power grid. *SANS Industrial Control Systems* (2016)
4. Puys, M., Potet, M.-L., Lafourcade, P.: Formal analysis of security properties on the OPC-UA SCADA protocol. In: Skavhaug, A., Guiochet, J., Bitsch, F. (eds.) *SAFECOMP 2016*. LNCS, vol. 9922, pp. 67–75. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45477-1_6
5. Dreier, J., Puys, M., Potet, M.-L., Lafourcade, P., Roch, J.-L.: Formally verifying flow integrity properties in industrial systems. In: *SECRYPT 2017–14th International Conference on Security and Cryptography*, Madrid, Spain, p. 12, July 2017
6. Byres, E.J., Franz, M., Miller, D.: The use of attack trees in assessing vulnerabilities in SCADA systems. In: *Proceedings of the International Infrastructure Survivability Workshop* (2004)
7. Cherdantseva, Y., Burnap, P., Blyth, A., Eden, P., Jones, K., Soulsby, H., Stoddart, K.: A review of cyber security risk assessment methods for SCADA systems. *Comput. Secur.* **56**, 1–27 (2015)
8. Piètre-Cambacédès, L., Bouissou, M.: Cross-fertilization between safety and security engineering. *Reliab. Eng. Syst. Saf.* **110**, 110–126 (2013)
9. Kriaa, S., Piètre-Cambacédès, L., Bouissou, M., Halgand, Y.: A survey of approaches combining safety and security for industrial control systems. *Reliab. Eng. Syst. Saf.* **139**, 156–178 (2015)
10. Kriaa, S., Bouissou, M., Piètre-Cambacédès, L.: Modeling the Stuxnet attack with BDMP: towards more formal risk assessments. In: *2012 7th International Conference on Risk and Security of Internet and Systems (CRiSIS)*, pp. 1–8. IEEE (2012)
11. Piètre-Cambacédès, L., Deflesselle, Y., Bouissou, M.: Security modeling with BDMP: from theory to implementation. In: *2011 Conference on Network and Information Systems Security (SAR-SSI)*, pp. 1–8. IEEE (2011)

12. Kriaa, S., Bouissou, M., Laarouchi, Y.: A model based approach for SCADA safety and security joint modelling: S-Cube. In: IET System Safety and Cyber Security. IET Digital Library (2015)
13. Rocchetto, M., Tippenhauer, N.O.: Towards formal security analysis of industrial control systems. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 114–126. ACM (2017)
14. Turuani, M.: The CL-Atse protocol analyser. In: Pfenning, F. (ed.) RTA 2006. LNCS, vol. 4098, pp. 277–286. Springer, Heidelberg (2006). https://doi.org/10.1007/11805618_21
15. Rocchetto, M., Tippenhauer, N.O.: CPDY: extending the Dolev-Yao attacker with physical-layer interactions. In: Ogata, K., Lawford, M., Liu, S. (eds.) ICFEM 2016. LNCS, vol. 10009, pp. 175–192. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47846-3_12
16. Behrmann, G., David, A., Larsen, K.G.: A tutorial on UPPAAL. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7
17. Puy, M., Potet, M.-L., Roch, J.-L.: Génération systématique de scénarios d’attaques contre des systèmes industriels. In: Approches Formelles dans l’Assistance au Développement de Logiciels, AFADL 2016, Besançon, France (2016)
18. ANSSI. Expression des besoins et identification des objectifs de sécurité. Agence nationale de la sécurité des systèmes d’information (2010)
19. CLUSIF. Méthode harmonisée d’analyse des risques (2010)
20. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982). <https://doi.org/10.1007/BFb0025774>
21. Dolev, D., Yao, A.C.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1981)
22. Cervesato, I.: The Dolev-Yao intruder is the most powerful attacker. In: 16th Annual Symposium on Logic in Computer Science–LICS, vol. 1 (2001)
23. IEC-60812. Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA). International Electrotechnical Commission (1985)



HuMa: A Multi-layer Framework for Threat Analysis in a Heterogeneous Log Environment

Julio Navarro^{1,2(✉)}, Véronique Legrand^{4,5}, Sofiane Lagraa⁶,
Jérôme François⁶, Abdelkader Lahmadi⁶, Giulia De Santis⁶, Olivier Festor⁶,
Nadira Lammari⁴, Fayçal Hamdi⁴, Aline Deruyver¹, Quentin Goux⁵,
Morgan Allard⁵, and Pierre Parrend^{1,2,3}

¹ ICube, Université de Strasbourg, Strasbourg, France

{navarrolara,aline.deruyver,pierre.parrend}@unistra.fr

² Complex System Digital Campus (UNESCO Unitwin), Paris, France

³ ECAM Strasbourg-Europe, Schiltigheim, France

⁴ CEDRIC, Conservation National des Arts en Métiers (CNAM), Paris, France

{veronique.legrand,ilham-nadira.lammari,faycal.hamdi}@cnam.fr

⁵ Intrinsec Sécurité, Nanterre, France

{veronique.legrand,quentin.goux,morgan.allard}@intrinsec.com

⁶ LORIA, Inria Nancy Grand-Est, Villers-lès-Nancy, France

{sofiane.lagraa,jerome.francois,abdelkader.lahmadi,giulia.de-santis,

olivier.festor}@inria.fr

<http://unitwin-cs.org/>

Abstract. The advent of massive and highly heterogeneous information systems poses major challenges to professionals responsible for IT security. The huge amount of monitoring data currently being generated means that no human being or group of human beings can cope with their analysis. Furthermore, fully automated tools still lack the ability to track the associated events in a fine-grained and reliable way. Here, we propose the HuMa framework for detailed and reliable analysis of large amounts of data for security purposes. HuMa uses a multi-analysis approach to study complex security events in a large set of logs. It is organized around three layers: the event layer, the context and attack pattern layer, and the assessment layer. We describe the framework components and the set of complementary algorithms for security assessment. We also provide an evaluation of the contribution of the context and attack pattern layer to security investigation.

Keywords: Security knowledge · Cognitive computing
Cybersecurity · Log analysis

This work was partially supported by the French Banque Publique d'Investissement (BPI) under program FUI-AAP-19 in the frame of the HuMa project.

1 Introduction

Security analysis is a rigorous process encompassing several phases described in ISO 27043. It is conducted by security analysts having expertise in the identification and understanding of indicators of potential threats in logs (computer system traces). They achieve this using business rules mainly based on their past experience and the technical documentation of devices in the network. The continuous growth of the volume of logs to be analyzed, as well as their heterogeneity, make the task of the analyst increasingly difficult and error-prone even with current support tools. Since no automated method exists that integrates human level complex reasoning, the support tools generate confusing results and a multitude of false positives and false negatives. The situation is further complicated by the advent of more complex and hard to find attacks, such as Advanced Persistent Threats (APTs). As a result, new tools capable of addressing the challenge of identifying threats in massive log repositories are needed. A broad distinction can be made between simple attacks, which can be analyzed from individual events, and the more complex or targeted ones, including the APTs, which affect more than one asset and require an in-depth investigation. The more complex attacks can be thought of as being composed of different steps that are spatially and temporally spanned.

Taken individually, the multiple steps composing an APT are not necessarily illegal. Furthermore, since they are spatially and temporally spanned, they may seem to be unrelated. Nevertheless, as a whole, they constitute a single powerful attack. Therefore, in order to detect and predict such threats, it is necessary to collect, analyze and correlate various sources of data and to create summarized views that are exploitable by security analysts. Most systems save the actions related to them in lines of text called *logs*. Then, a security investigation is usually based on manual analysis of these logs [4]. Applications that collect logs are known as SIEM (Security of Information and Event Management) in industry. They include correlation methods for the automatic search of attack evidence. However, in the current context of *big data* and the huge amount of logs generated in a network makes the analysis difficult, if not impossible.

To address this challenge, we propose HuMa, a multi-layer framework for the analysis of complex security threats. It brings together the individual contributions made by the authors under the label of the HuMa project. The framework is composed of three layers: the event layer, responsible for the representation of individual traces of malicious activities; the context and attack pattern layer, responsible for gathering information about technical requirements of the attacks; and the assessment layer, responsible for extracting attack information from massive logs. The event layer is typically based on monolithic rules. The attack pattern layer, is constructed from databases, such as the CVE and CAPEC repositories. The context layer includes information about the system that we aim to defend. The assessment layer represents complex attacks as attack graphs, and searches for matches between the graphs and the actual traffic in the Information System under supervision. Apart from the architecture of the framework, another contribution of this work is a set of complementary approaches for

the assessment layer. Two kinds of dependencies between events are considered: temporal and spatial dependencies. The analysis is performed either through a root-cause approach, or through graph matching using dynamic weighted graphs. This latter approach is implemented in Morwilog, which is an application of the Ant-Colony algorithm to security investigation in logs. An evaluation of the framework is performed by assessing the contribution of the context and attack pattern layer for such investigation.

This work is organized as follows. Section 2 presents the state of the art on Advanced Persistent Threat analysis. Section 3 defines the multi-layer investigation framework. Section 4 focuses on the assessment layer and introduces key algorithms to address the investigation challenge. Section 5 discusses the evaluation of the framework and provides further insight into its application scope. Section 6 concludes this work.

2 State of the Art

In this section we present a brief summary of the state of the art in the detection and analysis of Advanced Persistent Threats.

2.1 Modelling and Analysis of APTs

APTs [7, 30] are one of the most serious information threats that enterprises and government agencies are faced with today. Examples include Stuxnet [8] and Carbanak¹. Although individual APTs vary considerably, they are customized to the target system, and they all share the same 6 phases: reconnaissance, delivery, exploitation, operation, data collection and exfiltration [7].

Some work has been done to model APTs [5, 10, 15]. For instance, APTs can be described using low-level details, such as monitoring events, vulnerability descriptions and exploit information. Others use a top-down approach with high-level abstractions, based on attack trees or graphs. The most widely used model for APTs is the attack tree [29], where leaves or branches are linked by AND or OR gates. An improved attack tree is described in [5], where an O-AND gate and some extra attributes are added. Similar work is described in [3], where a SEQ gate and probability distributions are added to the leaves of the tree. However, attack trees are not the most suitable models for such threats [12], because they lack technical details, and providing them would make the trees too complex and difficult to read. A new conceptual attack model, the *attack pyramid*, is proposed in [14], which shows that an attack path may go across different environments of the organization. Cui et al. [10] focus on identification of attacks at early stages and prediction of their evolution using Hidden Markov Models (HMMs). Abraham and Nair [1] predict changes over time by capturing interrelations of vulnerabilities using attack graphs. They propose a three-layer architecture: layer 1 contains the attack graph model, whose vulnerabilities are quantified in layer 2. Layer 3 describes attacks by applying stochastic processes over the attack graph.

¹ https://securelist.com/files/2015/02/Carbanak_APT_eng.pdf.

2.2 Automatic Analysis of APT Scenarios

Although HuMa still considers the human analyst as a key player in the identification of attack scenarios, automatic methods for finding links between events are used to facilitate the expert analysis. The methods for identifying APT scenarios found in the literature work to a large extent on alerts generated by an Intrusion Detection System (IDS). An example method that uses this approach is AEC, or Active Event Correlation [6], applied on top of a Bro IDS. This allows it to interact with network traffic. Marchetti et al. [22] use alert graphs in a pseudo-Bayesian algorithm with the previous alert history as a reference. In other systems, such as RIAC [34], attacks are deduced from manually described prerequisites and consequences of individual alerts. However, in [32] the knowledge base of alerts is already contained in the system but the casual relationships are automatically extracted. Unsupervised methods have also been developed, such as the one in [31], based on the aggregation of similar alerts from the same time period in possible attack scenarios. Other approaches exist that do not work only with IDS alerts. Mathew et al. [23], for example, propose an anomaly detection method on a heterogeneous dataset using Principal Component Analysis (PCA). Another example is described in [13], where hypothesis and rules are deduced from a dataset of logs without attacks.

2.3 Correlation Between Vulnerabilities and Attack Patterns

Discovered vulnerabilities in information systems are in general publicly disclosed by means of the CVE (Common Vulnerabilities and Exposures) format, an open industrial standard widely adopted by many organizations. Each CVE document has an identifier and mainly provides a textual description of a security vulnerability or exposure. The documents are publicly available through multiple databases². In addition to CVE, CAPEC (Common Attack Pattern Enumeration and Classification)³ patterns are distributed by MITRE in XML format. They also provide a textual description of an attack, its prerequisites, its steps, severity and the attack methods used.

Several papers have studied the known vulnerabilities in order to predict their exploitation. In [33], the authors apply several machine learning algorithms on the National Vulnerability Database (NVD), with the goal of predicting undiscovered vulnerabilities. An interesting work regarding how vulnerabilities are exploited by attackers is described in [2]. The authors found that only a small subset of vulnerabilities in the NVD and Exploit-DB are found in exploit kits in the wild. However, little work has addressed the correlation between security alerts, CVE and CAPEC documents. In [28], the authors use data mining techniques to map CAPEC patterns to security logs. The obtained representations are then matched using a K-nearest-neighbour algorithm to obtain the closest

² <https://nvd.nist.gov>, <http://cve.mitre.org>, <http://www.cvedetails.com>.

³ <http://capec.mitre.org>.

events to an attack pattern. In [16], CAPEC and CVE patterns are used to generate attack graphs and then match security events against them to identify running attack scenarios.

3 The Security Multi-layer Framework

In this section, we discuss how HuMa uses the idea of classical SIEMs to create a cognitive platform where the human being is a key player. Thanks to novel representations of logs and attacks, HuMa combines the power of several analysis algorithms, whose results are continuously improved by feedback from the human expert.

3.1 Handling Security Knowledge

One of the key challenges of HuMa is to capitalize on the complex reasoning of security analysts. It aims to help experts in their decision making by providing them with a guiding tool. The tool should allow them to react on the fly to threats even in an environment where the logs are massive and heterogeneous and where malicious tactics are continuously evolving. The analysis is made more efficient as the focus is placed on the most noteworthy events.

To incorporate guidance, the proposed tool must integrate all the useful knowledge on the security analysis business. This knowledge can be represented using a domain ontology. Given the heterogeneity of the logs, in order to facilitate the correlation between them, we need to define a unique vocabulary of concepts to represent them. The domain ontology must contain this vocabulary and all other knowledge related to the security analysis.

3.2 Representation of Logs

We have developed a new representation of logs suitable for both machines and humans. The HuMa analysis process is based on concepts for representing logs, that are able to extract the relevant information about a security incident. Logs are big data streams, so they inherit the 4 properties of big data, called ‘the 4 V’s’: volume, velocity, veracity and variety. The concepts aim to reduce one dimension of the heterogeneity present in logs, i.e. the variability in expressing similar pieces of information. For instance, a ‘login failure’ event can be represented in a totally different way by a SQL server or by a router, although they have similar meanings and thus share some concepts. An extractor module automatically extracts concepts from the original logs (raw logs), guided by a knowledge representation model and without human intervention.

In addition, HuMa provides vulnerability analysis based on the concepts that have been developed by the MITRE in the CVE documents and CVSS scores. Each CVE identifier includes a unique identifier number, references about its vulnerability and a description of the vulnerability or exposure. This description contains information about the weakness of the affected asset and the outcome

and consequences of exploiting it. The CVSS associated with each CVE takes into account the information from the description. The CVE description and the assigned CVSS include knowledge provided by the security community that maintains the database, so each CVE identifier benefits from the knowledge of security experts on each of the vulnerabilities.

Thanks to machine learning tools, expert knowledge can contribute to reduce error in the process of concept extraction. The concept extraction module automatically generates pertinent questions that are then presented to the analyst in order to improve its internal knowledge database.

Concepts also allow to unify the way logs express information, beyond pre-processing relevant data. This guarantees that the analysis processes have a direct access to the information, regardless of the device or service implementation that generates it. Moreover, our use of concepts leads to a direct reduction in the complexity of the logs, compared to the information extracted by Splunk⁴ from raw logs. For instance, in a test involving a log dataset of 600000 entries from 6 different devices, the number of attributes is reduced from 62 to 16 and the number of values is reduced from 518 to 72. The concepts enrich the raw log such that the data added is compatible with existent technologies, while preserving the original log. The dataset composed of the raw log and its enriched data is processed by the rest of the HuMa platform.



Fig. 1. Modelling elements and their relations for the Carbanak attack

3.3 Representation of Attacks

An important question in HuMa is to find the best way to represent event scenarios that may pose a threat to the system. The representation should be common to all the methods integrated in the framework. To introduce our approach, we use the Carbanak cyberattack as an example. It is possible to describe this attack through the context in which it takes place, the collected events and known attack patterns (Fig. 1). For Carbanak to operate, the presence of Microsoft Office 2003, 2007 or 2010 (context) is required since the .doc file received via e-mail (event) has to be opened (event). The malware installed by this action creates a .bin file (event) in a folder created by Mozilla Firefox (context), which therefore needs to be installed as well. The reception of the .doc file via e-mail and the following double click on it are also related to the attack pattern of spear phishing (attack pattern). HuMa incorporates a multi-layer approach that

⁴ <https://www.splunk.com>.

identifies the links between the elements characterizing an APT like Carbanak: known attack models, detected events and knowledge of the system. Most of the work mentioned in Sect. 2 focuses on modelling already known attacks [5, 15, 30], whereas we focus on predictive modelling of an APT. Our work relies on a multi-layer modelling technique whose schema is shown in Fig. 2. The first layer consists of normal actions and alarms generated by security systems. These events are correlated and matched to the context (i.e. the configuration of the system), and to already known attack patterns. Context and well-known attack patterns form the second layer. The assessment layer contains the model for a possible attack scenario, created from the link between elements of the other layers. The linking process considers elements as the time-to-live of each step, the time in which each step takes place, the probability of success, shared context or users in common.

As shown in Fig. 2, each level is connected to the one above and below it. The collected events contribute to the selection of known attack patterns, and are then matched to them. Vice versa, the selected set of attack patterns helps to direct the search for events in the system. For example, in the Carbanak attack, if the spear phishing attack pattern is included in the selection, a search for the “reception of e-mails with attachments” event is performed. Similarly, the events help to define the context, which in turn guides the search for events. For instance, if the received attachment is a .doc file, HuMa checks whether Microsoft Office is installed in the system. Similar connections also exist between the selected attack patterns and the system context, and the assessment model. These layers work together to select attack models and to identify which part of the context needs to be investigated. For this task, we rely on the CAPEC and CVE databases and the help of security analysts (see Sect. 4.2).

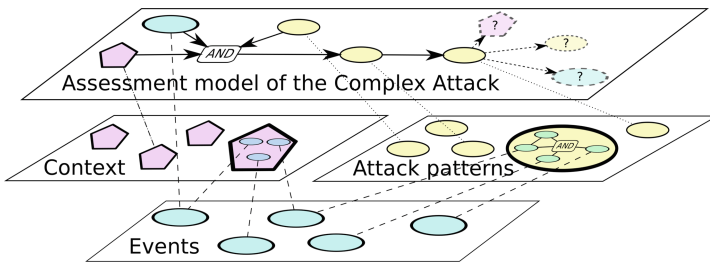


Fig. 2. Proposed approach for predictive APT modelling

4 Analysis Engines in HuMa

The analysis of APTs in HuMa involves several bricks that work together to return a combined result in the same dashboard.

4.1 Dependencies

Temporal dependencies. An attack usually involves periodic changes in behaviour over time. We can find clear examples in denial of service attacks or port scans. Although the time lapse between two requests may vary, there is a certain periodicity that can be identified by studying the shift in the process state of the communication with a target machine. The difficulty of this task is related to the fact that changes are not necessarily explicit and can be mixed with other types of events. HuMa includes a method to find temporal dependencies between logs ordered in time. It is based on data mining techniques and the representation of logs by the high-level semantic concepts introduced in Sect. 3.2. The goal is to discover temporal dependencies via frequent and periodic patterns of logs ordered in time. The method automatically returns unexpected temporal changes, as well as the context in which these changes take place. Figure 3 presents the results of mining with fixed slice-window periods. Patterns are represented on the Y-axis and time windows on the X-axis. The attributes in the patterns refer to those in Snort⁵. Each cell represents a pattern frequency. Darker cells represent frequent patterns and lighter cells, infrequent ones. This approach reduces the number of patterns to be analyzed and guarantees the temporal consistency of conceptualized logs. More details can be found in [17].

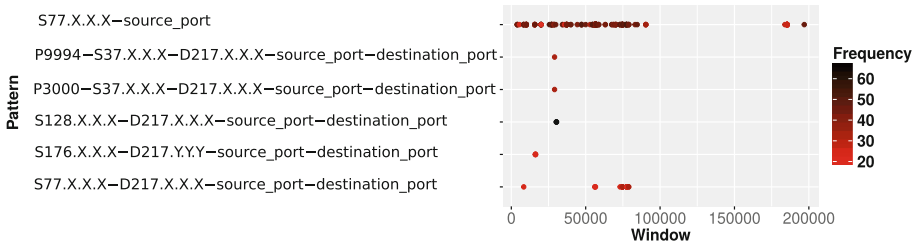


Fig. 3. Overview of frequent and periodic patterns.

Spatial dependencies. In HuMa, we also exploit spatial dependence discovery techniques to find correlations or similarities among events in order to build clusters. Indeed, rather than analyzing a single event, a human expert can benefit from clusters in order to understand a group behaviour and speed up the analysis. Relevant methods can be found in the area of data mining with clustering approaches. There are numerous options available, but a major bottleneck of many of them is their computational complexity as they require pair-wise comparisons between initial data points, *i.e.* events or concepts in our case. We thus propose to use TDA (Topological Data Analysis) [25]. Which reduces the high dimensionality of the data through a simpler representation that can be searched for invariants. Such invariants can then be considered as significant patterns of

⁵ <https://www.snort.org/>.

the underlying data. To achieve our goal, *i.e.*, the Mapper [26] algorithm from TDA is integrated in HuMa. Rapidly, the algorithm works as follows:

1. The original highly dimensional space is decomposed as overlapped hypercubes.
2. In each hypercube, a clustering algorithm is applied.
3. A graph over all data points is created, where a vertex represents a cluster in a hypercube. An edge between two vertices exists if and only if the two underlying clusters share at least one original data point, which is possible due to the overlapping of hypercubes.

There are therefore three major parameters: (1) the resolution, representing the number of hypercubes (the smaller, the greater the amount of hypercubes); (2) the overlap between hypercubes, and (3) the clustering algorithm. In our case, DBSCAN is used. It is a density-based clustering algorithm that does not require an a priori estimate of the number of clusters. However, it induces two other parameters to be set, namely the minimum number of neighbours at a given maximal distance for each clustered point. This technique has been successfully applied to Darknet analysis in our prior work [9] and it has been fine-tuned with respect to the HuMa cognitive framework for the analysis of concepts extracted from logs as a first step.

HMMs applied to logs of scanning activities. During the reconnaissance phase of APTs [7], powerful scanning tools are used by attackers. The availability of models describing various aspects of these scanning activities can help security experts to predict whether an attack is underway. In HuMa, we model intensity, spatial and temporal movements of scanning techniques using mixture distribution models and HMMs, based on logs extracted from a /20 darknet. A combination of mixture distribution models and HMMs are used since logs may be divided into unobserved clusters. First, mixture distribution models provide the probability of the clusters. Second, the corresponding HMM, whose states are the distributions of the mixture, provides the transition probabilities between clusters. The obtained models are presented in the dashboard of HuMa, so that the security analyst can determine whether there is a scanning activity and prepare for a security attack. So far, this engine has been applied to logs in the reconnaissance phase of an APT, but we are currently working on its application in all phases. A more detailed description of the method has been published in [11].

Dependency analysis. The conceptualization of the heterogeneous logs in HuMa implies that abnormal behaviour mining techniques can be used efficiently. These approaches aim to find the dependencies of abnormal behaviour. Given a set of conceptualized log sequences, the problem is both to identify the abnormal behaviours and the set of dependencies able to guide the analyst. Figure 4 shows an abnormal behaviour graph extracted from a set of logs. The graph represents the activity of an IP address 81.89.X.X targeting a web server. The red and blue

boxes represent the start and end of the graphs, respectively. This graph helps the analyst to understand an abnormal behaviour which may be a potential threat and localize it within a time window. Indeed, this graph was confirmed as a representation of a real attack performed from the IP address.

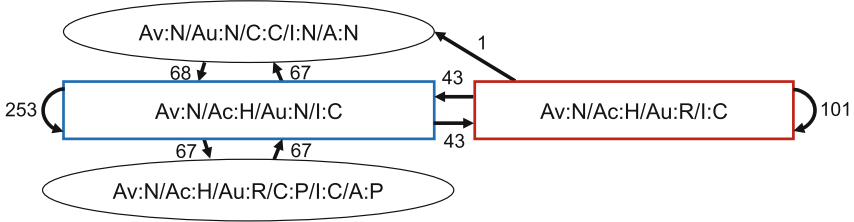


Fig. 4. Abnormal behaviour graph. (Color figure online)

4.2 Root Cause Analysis

A root cause analysis (RCA) is a specific description of an attacker’s procedure that identifies all the requirements and causes that led to an incident [19,20]. A RCA is particularly useful for producing an analysis and incident report. The description of a RCA is different from that of dependencies as it provides an explanation about the incident and details about what happened. The description includes the conditions required for executing the described actions as states of the system. They may also include a description of vulnerabilities. The MITRE database is an extremely rich public database, containing more than 80000 CVE identifiers. Each CVE identifier contains specific details of the affected system. The exploitation of a vulnerability is generally crucial in the execution of an APT. For instance, one of the steps of Carbanak is the exploitation of CVE-2015-5262, related to an incorrect configuration of the ‘keepalive’.

Matching vulnerabilities and attack patterns. Our approach for matching security events to CVE and CAPEC documents is close to that of [28], since we share the same goal. However, we apply a recent machine learning technique, *doc2vec* [18], in order to learn from the textual descriptions of CVE and CAPEC documents. We used the cosine similarity metric to mutually match the embedding vectors obtained from the text, and evaluate their correlation. We applied this technique to the available set of 510 CAPEC patterns and a set of 91405 CVE documents available from the MITRE web site. In both cases, we compute the respective embedding vectors using the *gensim* [27] *doc2vec* python library. Then, for each CVE and CAPEC document, we compute the 10 most similar documents using the learned vectors, that results on 10 similarity values ranging between 0 and 1.

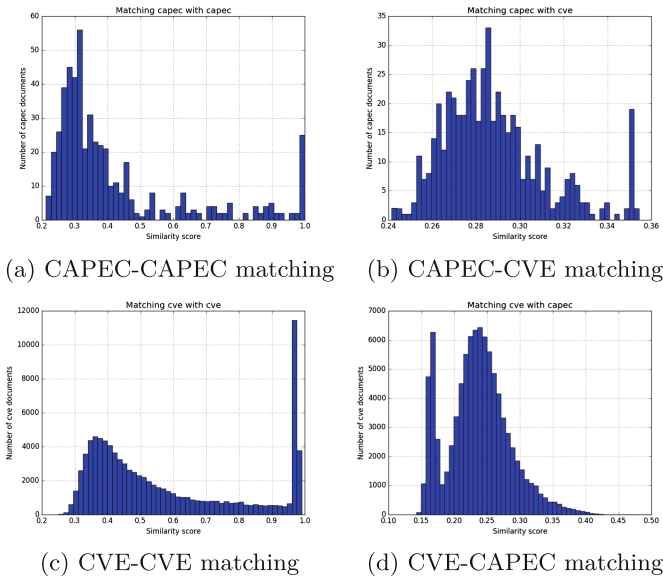


Fig. 5. Distribution of best matching scores using *doc2vec* algorithm.

Evaluation. In this section, we present the experiments performed on the module matching vulnerabilities and attack patterns, which were not published previously. In a first step, we analyzed the distributions of the similarity scores obtained by matching CAPEC and CVE documents. Figure 5 shows the histograms of these scores for the most similar document only. As shown in Fig. 5a, when matching CAPEC to CAPEC documents, we observe that for a similarity value about 0.3 we obtain the most matched documents, around 56, or 10% of the documents. However, when matching CVE to CVE documents, as shown in Fig. 5c, we found that 11458 documents are similar with a score up to 1, representing 12.5% of the analyzed documents. When matching CAPEC with CVE documents, we found that 33 CAPEC documents match CVE documents with a score around 0.28, representing 6.4% of the CAPEC documents, as shown in Fig. 5b. When matching CVE with CAPEC documents, as shown in Fig. 5d, we found 6895 CVE documents match CAPECs with a score around 0.25, representing 7% of the analysed CVE documents. We thus observe better matches between CVE documents with a score up to 1 for 12.5% of the analyzed documents. For the other matching scenarios, the results are close with similarity scores between 0.25 and 0.3.

In a second step, we calculated the similarity scores for a sample of 10000 conceptualized logs coming from the test environment of a security company and compared them with CAPEC and CVE documents. The results are shown in Fig. 6. As shown in Fig. 6a, we observe that log-log matches obtain high similarity scores. The scores are mostly between 0.8 and 1, which means that multiple logs are similar and could be easily aggregated before being presented to a

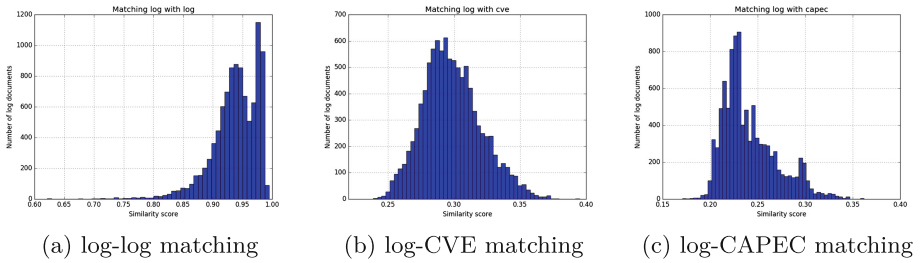


Fig. 6. Distribution of similarity scores using *doc2vec* algorithm.

human analyst. Matching the logs with CVE documents, as shown in Fig. 6b, produces similarity scores normally distributed around 0.3, with 613 logs achieving this score. When matching logs with CAPEC documents, as shown in Fig. 6c, we observe a maximum number of 907 log documents that match CAPECs with a similarity score around 0.22. Using the *doc2vec* technique, we are able to match logs with their respective most similar CAPEC and CVE documents, which means that we can associate them to vulnerabilities and attack patterns. These associations help security analysts to better understand what is happening in the system.

4.3 Searching for Paths in Event Graphs with Morwilog

Once a database of event graphs is generated, we need to know which paths may be of special interest to the analyst. Paths may also exist that are erroneous or that no longer pose a threat. To address this in the context of HuMa, we developed an algorithm called Morwilog based on Ant Colony Optimization (ACO), a metaheuristic to solve discrete optimization problems. ACO is inspired by the behaviour of a colony of foraging ants when they leave the anthill in the search of food. In this process, ants deposit pheromones so that other ants can follow the traces to the food source. This results in the formation of well-ordered trails from the anthill to the food source. After some time, almost every ant follows the shortest path, where pheromones are deposited at a higher rate. In ACO, a set of artificial ants is generated to find the shortest path in a graph.

Artificial ants in Morwilog are called *morwis*, and their generation is associated with the arrival of a log. When a suspicious log arrives at Morwilog, a *morwi* is generated and it proceeds through the event graph whose root node corresponds to this log. Event graphs are deployed here as trees, with deeper levels corresponding to logs arriving later in time. The *morwi* chooses a path to follow according to the level of pheromones on it. A path with a higher pheromone level has a higher probability of being chosen. At each node, the *morwi* waits a certain time for the arrival of logs in the following level. Figure 7 shows an event tree with the path followed by a *morwi*. The sequence of logs found is returned as an alert. After human validation, the level of pheromones in that

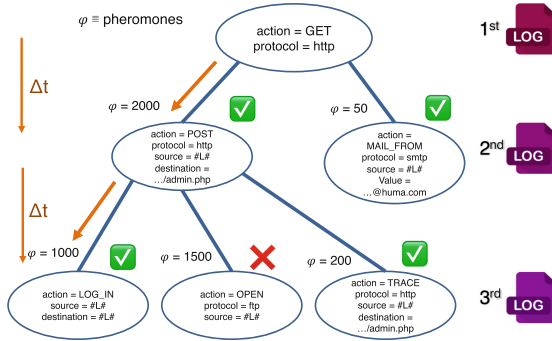


Fig. 7. Tree of log sequences

path is incremented if the sequence corresponds to a threat, and decremented otherwise. More details about the algorithm can be found in [24].

5 Discussion

5.1 Innovation in Log Analysis

In classical SIEMs, correlation is based on rules composed of text strings, which are searched as patterns in the logs in a linear way. If a log matches a rule, it becomes a possible candidate to be part of an attack scenario. Each rule matches one of the steps in the attack. An alert is generated when the whole scenario or part of it is detected. One of the disadvantages of rule-based analysis engines is the description of the rule sets. They are manually written by analysts, so it is easy to find errors. In addition, the volume of rules that must be created is too high to be managed by a human analyst, as the number of technologies used in an organization continually increases. In the cognitive framework developed for HuMa, there are no predefined and static rules. In contrast to classic correlations where the human analyst is situated at the end of the linear process chain, the objective in HuMa is to place the analyst at the core of the analysis process. This means that the analyst can intervene at any point of the process. The interface with the human operator is a key component in the conception of HuMa. The learning loop, which allows the system to automatically learn new links between the logs, turns the system into an extension of the analyst’s way of thinking. Besides, HuMa is not intended as a substitute for classical SIEMs, but to complement them. Rule-based systems are necessary for detecting well-known threats. We can obtain signatures directly from security vendors, whose research teams identify and analyze attacks from all over the world.

5.2 Innovation in the Representation of Logs and Attacks

HuMa also proposes an innovative approach to processing and representing information. SIEMs are generally based on a broad classification of logs, and not much

work has been done on the development of well defined ontologies. In HuMa, we incorporated the work on log concepts developed by Legrand [21], who applies an ontology based on security indicators. The transformation of raw logs into sets of concepts allows the preservation of the original information, which is enriched with underlying meaning provided by security analysts. The automatic semantic analysis is crucial in HuMa, resulting in more enriched logs that allow security analysis methods to work in a more efficient way. Moreover, these concepts are better understood by humans than raw log text, so they are also useful to the security analyst during an investigation. The multi-step nature of APTs necessitate an innovative way of representing attacks. In the context of HuMa, we propose a novel approach to model APTs that integrates low-level events with attack patterns to identify relations between them. The model relies on three layers: one for events, one for context and known attack patterns, and the assessment layer where the model of the advanced persistent threat is stored. Existing approaches focus on handling of events [14], or rely on existing attack patterns to be matched with detected events [10]. Our approach combines both of these. This representation of attacks is at the core of the set of security analysis algorithms developed for HuMa. Having a common format eases the exchange of information between algorithms. The analyst can thus obtain a single result, which is the combined outcome of the set of methods.

6 Conclusions and Perspectives

In this work, we introduce, implement and evaluate a complete multi-layer investigation framework to address the challenge of Advanced Persistent Threats. This framework is organized into three layers: the assessment layer, the context and attack pattern layer, and the event layer. We propose and evaluate a set of algorithms for the assessment layer, including temporal and spatial dependencies, root cause analysis, and ant-colony based analysis. A qualitative application of the framework to the Carbanak attack is presented. The investigation process for the assessment layer algorithm is defined. A quantitative evaluation of the contribution of the context and attack pattern layer to the investigation performance is given. This highlights how the integration of insights from CVE and CAPEC resources improves the ability to identify complex attacks such as APTs in massive logs. This work represents a first step in the definition of a comprehensive framework for the investigation of APTs. HuMa still needs to be complemented with more features for the integration of the human expert, who, beyond being a simple observer, also has the knowledge required to enrich the preliminary analyses proposed by the framework. Assisted learning is likely to become a major topic of interest for security investigation in the near future.

References

1. Abraham, S., Nair, S.: A predictive framework for cyber security analytics using attack graphs. *Int. J. Comput. Netw. Commun.* (2015). <http://arxiv.org/abs/1502.01240>
2. Allodi, L., Massacci, F.: A preliminary analysis of vulnerability scores for attacks in wild: the ekits and sym datasets. In: *Proceedings of the 2012 ACM Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2012*, pp. 17–24. ACM, New York (2012). <https://doi.org/10.1145/2382416.2382427>
3. Arnold, F., Hermanns, H., Pulungan, R., Stoelinga, M.: Time-dependent analysis of attacks. In: Abadi, M., Kremer, S. (eds.) *POST 2014. LNCS*, vol. 8414, pp. 285–305. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54792-8_16
4. Benali, F., Ubéda, S., Legrand, V.: Collaborative approach to automatic classification of heterogeneous information security. In: *Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE 2008*, pp. 294–299. IEEE (2008)
5. Camtepe, S., Yener, B.: Modeling and detection of complex attacks. In: *SecureComm Third International Conference on Security and Privacy in Communications Networks and the Workshops*, pp. 234–243, September 2007
6. Chen, B., Lee, J., Wu, A.S.: Active event correlation in Bro IDS to detect multi-stage attacks. In: *Fourth IEEE International Workshop on Information Assurance (IWIA 2006)*, pp. 16–50. IEEE, London (2006)
7. Chen, P., Desmet, L., Huygens, C.: A study on advanced persistent threats. In: De Decker, B., Zúquete, A. (eds.) *CMS 2014. LNCS*, vol. 8735, pp. 63–72. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44885-4_5
8. Chen, T.M., Abu-Nimeh, S.: Lessons from stuxnet. *Computer* **44**(4), 91–93 (2011)
9. Coudriau, M., Lahmadi, A., Francois, J.: Topological analysis and visualisation of network monitoring data: darknet case study. In: *International Workshop on Information Forensics and Security (WIFS)*. IEEE, Abu Dhabi (2016)
10. Cui, Z., Herwono, I., Kearney, P.: Multi-stage attack modelling. In: *Proceedings of Cyberpatterns 2013*, pp. 78–89 (2013)
11. De Santis, G., Lahmadi, A., Francois, J., Festor, O.: Modeling of IP scanning activities with hidden Markov models: darknet case study. In: *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pp. 1–5. IEEE (2016)
12. Flåten, O., Lund, M.S.: How good are attack trees for modelling advanced cyber threats? *Norw. Inf. Secur. Conf. (NISK)* **7**(1) (2014)
13. Friedberg, I., Skopik, F., Settanni, G., Fiedler, R.: Combating advanced persistent threats: from network event correlation to incident detection. *Comput. Secur.* **48**, 35–57 (2015)
14. Giura, P., Wang, W.: Using large scale distributed computing to unveil advanced persistent threats. *Science* **1**(3), 93 (2013)
15. Kordy, B., Piètre-Cambacédés, L., Schweitzer, P.: Dag-based attack and defense modeling: don't miss the forest for the attack trees. *Comput. Sci. Rev.* **13–14**, 1–38 (2014)
16. Kotenko, I., Chechulin, A.: A cyber attack modeling and impact assessment framework. In: *2013 5th International Conference on Cyber Conflict (CYCON 2013)*, pp. 1–24, June 2013

17. Lagraa, S., Legrand, V., Minier, M.: Behavioral change-based anomaly detection in computer networks using data mining. *Int. J. Network Manag.* (Submitted)
18. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Jebara, T., Xing, E.P. (eds.) *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)*, pp. 1188–1196. *JMLR Workshop and Conference Proceedings* (2014)
19. Legrand, V., State, R., Paffumi, L.: A dangerousness-based investigation model for security event management. In: *The Third International Conference on Internet Monitoring and Protection, ICIMP 2008*, pp. 109–118. *IEEE* (2008)
20. Legrand, V., Ubeda, S.: Enriched diagnosis and investigation models for security event correlation. In: *Second International Conference on Internet Monitoring and Protection, ICIMP 2007*, p. 1. *IEEE* (2007)
21. Legrand, V.: *Confiance et risque pour engager un échange en milieu hostile*. Ph.D. thesis, *INSA-Lyon* (2013)
22. Marchetti, M., Colajanni, M., Manganiello, F.: Identification of correlated network intrusion alerts. In: *Third International Workshop on Cyberspace Safety and Security (CSS)*, pp. 15–20. *IEEE, Milan* (2011)
23. Mathew, S., Upadhyaya, S.: Attack scenario recognition through heterogeneous event stream analysis. In: *IEEE Military Communications Conference (MILCOM)*, pp. 1–7. *IEEE, Boston* (2009)
24. Navarro-Lara, J., Deruyver, A., Parrend, P.: Morwilog: an ACO-based system for outlining multi-step attacks. In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. *IEEE, Athens* (2016)
25. Offroy, M., Duponchel, L.: Topological data analysis: a promising big data exploration tool in biology, analytical chemistry and physical chemistry. *Anal. Chim. Acta* **910**, 1–11 (2016)
26. Pearson, P., Muellner, D., Singh, G.: *TDAMapper: Analyze High-Dimensional Data Using Discrete Morse Theory* (2015). <https://github.com/paultpearson/TDAMapper/>, (R package version 1.0)
27. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50. *ELRA, Valletta, May 2010*
28. Scarabeo, N., Fung, B.C., Khokhar, R.H.: Mining known attack patterns from security-related events. *PeerJ Comput. Sci.* **1**, e25 (2015)
29. Schneider, B.: Attack trees. *Dr. Dobb's J.* **24**, 21–29 (1999)
30. Sood, A.K., Enbody, R.J.: Targeted cyberattacks: a superset of advanced persistent threats. *IEEE Secur. Priv.* **11**(1), 54–61 (2013)
31. Wang, L., Ghorbani, A., Li, Y.: Automatic multi-step attack pattern discovering. *Int. J. Netw. Secur. (IJNS)* **10**(2), 142–152 (2010)
32. Zali, Z., Hashemi, M.R., Saidi, H.: Real-time attack scenario detection via intrusion detection alert correlation. In: *9th International ISC Conference on Information Security and Cryptology (ISCISC)*, pp. 95–102. *IEEE, Tabriz* (2012)
33. Zhang, S., Caragea, D., Ou, X.: An empirical study on using the national vulnerability database to predict software vulnerabilities. In: *Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) DEXA 2011*. *LNCS*, vol. 6860, pp. 217–231. *Springer, Heidelberg* (2011). https://doi.org/10.1007/978-3-642-23088-2_15
34. Zhaowen, L., Shan, L., Yan, M.: Real-time intrusion alert correlation system based on prerequisites and consequence. In: *6th International Conference on Wireless Communications Networking and Mobile Computing (WiCOM)*, pp. 1–5. *IEEE, Chengdu City* (2010)



Monitoring of Security Properties Using BeepBeep

Mohamed Recem Boussaha, Raphaël Khoury^(✉), and Sylvain Hallé

Laboratoire d'informatique formelle, Université du Québec à Chicoutimi,
Chicoutimi, Canada
`raphael.khoury@uqac.ca`

Abstract. Runtime enforcement is an effective method to ensure the compliance of program with user-defined security policies. In this paper we show how the stream event processor tool BeepBeep can be used to monitor the security properties of Java programs. The proposed approach relies on AspectJ to generate a trace capturing the program's runtime behavior. This trace is then processed by BeepBeep, a complex event processing tool that allows complex data-driven policies to be stated and verified with ease. Depending on the result returned by BeepBeep, AspectJ can then be used to halt the execution or take other corrective action. The proposed method offers multiple advantages, notable flexibility in devising and stating expressive user-defined security policies.

1 Introduction

Mobile code has emerged as an effective solution to the challenges of computing in distributed systems. Nonetheless, security concerns remain omnipresent, and may act as a break to the adoption of this technology, in part because of the need for each user to tailor the security policy governing his system to his own need.

In this paper, we show how BeepBeep [5], a complex event processor can be used as a runtime monitor for enforcing a wide array of user-defined security policies. This study also serves to illustrate BeepBeep's capabilities of as a log trace analyzer. BeepBeep takes as input a data stream, in this case an execution trace capturing the method calls and parameters values. This information can be generated using any number of mechanism. BeepBeep has the capacity to efficiently analyse this information in real-time to determine if it conforms with a user defined specification. BeepBeep can also generate useful diagnostic information about the program's runtime behavior, which can in turn be used for further security analysis or debugging.

We rely upon AspectJ [7] to generate the input trace which allows BeepBeep to perform the enforcement. However, the monitoring using BeepBeep is agnostic of the mechanism used to generate the traces, and while AspectJ also exhibits some capabilities to operate as a security policies enforcement mechanism on its own, that is not necessarily the case for other tracers. The use of BeepBeep allows the security enforcement mechanism to be independent from the tracer.

Like other runtime security enforcement mechanism, the approach presented in this paper is *precise*, in the sense that it reject only those executions that violate the security policy, permitting safe executions of the program to proceed, and it results in no false-positives or false-negatives. It is *late*, in the sense that the execution is not halted until a violation is about to occur, thus allowing as much of the execution to take place as is permissible given the security policy in place. The main advantage of the proposed approach over other monitoring tools is the flexibility and expressivity of the policy specification language.

The remainder of this paper is organised as follows. Section 2 surveys related works. In Sect. 3, we give an overview of the architecture of the security enforcement mechanism proposed in this paper. Section 4 describes some of the security properties we can enforce and Sect. 5 presents experimental results. Concluding remarks are given in Sect. 6.

2 Related Work

The Naccio project [4] provides a library of Java security policies that are enforced at runtime. Each policy replaces certain Java Virtual Machine (JVM) classes as needed to allow enforcement, and the JVM must be modified to ensure that the correct (security policy specific) class is used. Any policy part of Naccio's library, as well as a multitude of policies unavailable on that platform, can easily be stated and enforced using BeepBeep.

Several tools leverages machine learning techniques and static analysis to categorize Android applications (written in Java) as either malicious or benign. The tool ANDRANA [3] relies on static analysis of the applications's code to create a vector of features for each application. Classification is then performed to determine if the observed features a typical to those previously observed in malware. Other classifier rely upon the app's manifest file [9], it's service life cycle [6], API calls [1] or a combination of API calls and other statically detected features [2]. Like other methods based on static analysis, these exhibits a risk of false-positive and false-negatives, and are vulnerable to obfuscation.

Another countermeasure in the face of malicious mobile code is the reliance upon of code certification. Code signing utilizes cryptographic keys to guarantee the authorship of code. While a useful security tool, code signing only serves to authenticate the author of a given code, but provides no guarantees as to its actual behavior. The author may be wrongly trusted by a user, and even code from reputable sources can exhibit an exploitable vulnerability.

The approach proposed in this paper is precise and thus risks neither false-positives nor false-negatives. It can be applied to code of unknown origin, and allows the user to easily customize the security policy to his needs. Indeed, as we will show in the next section, it can be used not only to enforce a wide variety of security policies but also to ensure the respect of resource-usage constraints or to generate diagnostic reports about the program's runtime behavior.

3 Architecture

The trace is generated using AspectJ, a tool that allows adding executable blocks to the source code without explicitly changing it. AspectJ allows programmers to set points in the source, known as pointcuts, to where the execution is to temporarily halt and while the newly added code blocks are executed. In our case, we used AspectJ to insert code before and after every method call to record the information needed to perform monitoring. As mentioned above, this is only one of several methods that could be used to generate the trace.

Figure 1 shows a sample of the trace of a simple Java tutorial program. Each line correspond to either a single method call or method return. The former begin with the keyword the keyword ‘call’ contains the following information: the method return type, the method’s containing class, the method’s name, each of the methods parameters type and value, and finally the method’s call level on the stack. The later begin with the keyword ‘Return’ and contain the return value. Values of literals, string and elementary types are provided explicitly but those of objects are provided by references. Arrays are prefixed with ‘[’.

```
Call: return type : void // class: MonitoredProgram // method: main //
type param 1: class [Ljava.lang.String;// value param 1:[Ljava.lang.String;@72ea2f77//level: 0
Call: return type : void // class: java.net.Socket // method: <init> //
type param 1: class java.lang.String // type param 2: int //
value param 1: www.javatutorial.com // value param 2: 80 // level: 1
Return: Socket[addr=www.javatutorial.com/69.172.201.153,port=80,localhost=51706]
Call: return type : class java.net.InetAddress //
class: java.net.Socket // method: getInetAddress // no parameter // level: 1
Return: www.javatutorial.com/69.172.201.153
```

Fig. 1. A fragment of a trace

BeepBeep [5] is a complex event processing tool that can perform complex manipulations on large data streams efficiently. Internally, BeepBeep decomposes the desired data-processing task into a number of atomic *processors*, each of which takes as input one (or more) event streams, and in turn, outputs one or more event streams. These processor are chained together with the output of one (or more) processor being piped to the input of the next one in such a manner that, feeding BeepBeep’s input stream through this chain produces the desired computation. Part of the contribution of this paper is to show how complex, data-driven security properties of programs can be stated in terms of a small number of BeepBeep processors.

A benefit of the approach under consideration is the ease with which the desired security property can be stated. Each BeepBeep processors consists in an average 20 lines of Java code, contained in a single class. Users can reuse these elementary blocs, chaining them together to easily compose complex policies.

4 Security Properties

We began by replicating several of the security properties present in Naccio’s library. Most of these are safety policies and can be enforced with as little as

one or two BeepBeep Processors. Such properties include: NoExec, NoJavaClassLoader, NoNetReceiving, NoNetSending, NoPrinting, NoReadingFiles, NoListingFiles LimitBytesWritten and LimitBytesRead, LimitCreatedFiles, LimitObservedFile. The first 7 of these properties simply halt the execution upon encountering a specific forbidden method call. The latter 4 are only slightly more involved. LimitBytesWritten and LimitBytesRead limit to total number of bytes that are written (resp. read) to files or to the network. LimitCreatedFiles and LimitObservedFile limit the number of files that can be created (resp. read).

Figure 2 gives an example of the BeepBeep processors for the property LimitBytesWritten. It consists of only 3 processors: the first extracts from the trace those method calls that perform write operation and passes those method calls to the second processor. The second extracts number of bytes written by from these method calls, and again passes this information on to the next processors. The final processor computes the sum of the values it receives as input and aborts the execution (through AspectJ) if this sum surpasses a customizable value recorded in the property.

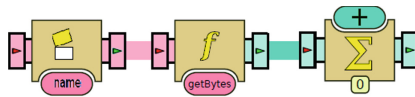


Fig. 2. The BeepBeep processors for property LimitBytesWritten

Schneider introduced the property [10] ‘no send after read’ as a typical example of a safety property. This property states that after having read from a protected file, the program is no longer allowed to access the network. This property is also part of the Naccio library.

The expressiveness of the approach under consideration is illustrated by the following pair of properties: the property ‘a is a key’, states that a given piece of information provided in the trace, such a parameter to a specific method or its return value, never take the same value twice. Conversely, it’s negation, the property ‘a is not a key’ requires that this value be unique. These properties are interesting since, as observed by Segoufin [11], several widely used data models can express one of these properties, but not the other, and neither of these properties is part of the Naccio library. Both can be stated using relatively simple processors, that stores the values that have appeared so far in the execution in a list, and consult this list before allowing the execution to proceed.

Deserialization attacks [8] have recently emerged as important vector of attack against Java programs. Any program that relies upon serialized objects to exchange information with a distant party may be susceptible to a serialization attack, even if the data is validated after having been received. The attack can be performed in any one of several ways, notably by sending data of an unexpected type, by sending an object of the correct type but with a high order nested structure, leading to resource exhaustion when it is deserialized or by sending

an object whose fields values are not consistent with the normal execution of the program. BeepBeep processors offer a simple and effective counter-measure in all cases. Since the trace contains return values and their type, validation can be performed with a processor similar to the ones described above. To protect against a deserialization bomb, we developed a processor that bounds the number of consecutive nested calls of the `readObject()` method. BeepBeep can also ensure important data secrecy properties by preventing data read in sensitive files from being included in the serialized object.

BeepBeep allows us to state more complex policies that relate the values present in different parts of the trace to one another. For example:

- The parameter values of a given function are always increasing/decreasing in consecutive calls. This property ensures the correctness of recursive function.
- After being created, a given data object is not modified (data integrity).
- No thread is frozen for more than 100 ms before resuming its execution (starvation freedom).
- Whether two specific methods work on the same object, or alternatively provide a list of objects that are manipulated by both of these methods.

Since the output of a processor can be of any format, BeepBeep can also provide profiling information about the ongoing execution, such as maximal, minimal and average stacks depth, the number of objects created for each object type, etc. We present two final processors that illustrate this capability of BeepBeep: processor `CallSequenceProfiling` lists, for each method call in the trace, the number of times it directly calls every other method. This information is provided in the form of a directed weighed graph, in which each vertex is labelled with a method name, and a vertex of weight c is present between vertexes v_1 and v_2 iff method v_1 calls method v_2 c times in the trace. We give a schematic representation of this processor in Fig. 3. The processor `BytesWrittenGraph` provides a list of every method that manipulates each data object. Recall that data objects are identified by reference in the trace. These two processors, while not security property enforcers, provide crucial information on data flow analysis that is essential to debugging and to the enforcement of data flow policies.

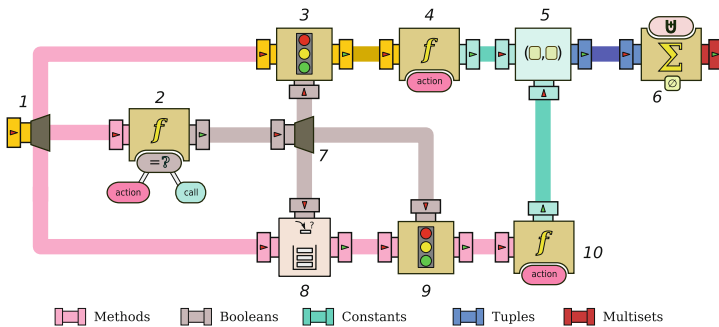


Fig. 3. The BeepBeep processor chain for property `CallSequenceProfiling`

Figure 3 shows the chain of processors required to compute the call graph from an execution trace. The core of this chain is the **Stack** processor, depicted as number 8 in the figure. It takes two event streams as its input: the first (left-hand side) is a stream of events of an arbitrary type; the second (top side) is a stream of Boolean values. This processor internally maintains a stack of received events. To this end, the Boolean stream acts as a *push flag*. When an event e and a Boolean value b arrive at the processor's inputs, two situations may occur. If $b = \top$, the top of the stack (if not empty) is output but not removed, and e is then pushed onto the internal stack if $b = \top$. If $b = \perp$, e is ignored, and the element at the top of the internal stack is popped and discarded.

The original stream of method events is first split in three (1); one of these copies is given as the input to the **Stack** processor (8), while another is sent to a **Function** processor (2). This processor evaluates the function that compares the **action** field of the method event with the constant **call**; the result is a stream of Boolean values, indicating whether the incoming event is a method call (\top) or a method return (\perp). This stream itself is split in three (7), and one of these copies is given as the push flag of the **Stack** processor. The stack is hence instructed to push an incoming event when it is a method call, and to pop the top of the stack when it is a method return. As a result, the output of the **Stack** processor is the method event corresponding to the current method in the program's execution.

A third copy of the original stream of events is sent to a **Filter** processor (3). This processor receives two inputs: an arbitrary event e and a Boolean value b called the *filter flag*. Event e is output if $b = \top$, otherwise e is discarded. The filter flag, in this case, is the result of applying the function **action = call**, which returns a Boolean value; in other words, the processor keeps only method call events, and filters out method returns. The same filtering condition is applied to the output of the **Stack** processor (9).

The end result of this first part of the chain is that processors 3 and 9 synchronously output method call events; events at matching positions in the streams represent the caller of a method (9) and the method being called (3). From this point on, the rest of the processing is straightforward. Both events are processed so that only the value of their **name** field is kept (4, 10); these two values are then joined in a tuple (5), and these tuples are then accumulated into a multiset using a **CumulativeProcessor** (6).

The output stream resulting from 6 is a sequence of multisets, each of the form $\{(m_0, n_0), \dots, (m_k, n_k)\}$; each tuple (m_i, n_i) is a caller-callee pair of method names. The number of times each distinct pair occurs in the multiset corresponds to the number of times n_i was called from m_i in the trace. From then on, it is easy to take the multiset of tuples and convert it into a directed graph that shows the weighted dependencies between methods in the observed execution.

It is worth mentioning that, in this whole graph, only the **action** function (used in processors 2, 4 and 10) and **Stack** processor (8) are specific to our use case. This amounts to 35 lines of custom code. All the remaining processors and functions are generic, and already come in BeepBeep's core or one of its existing palettes.

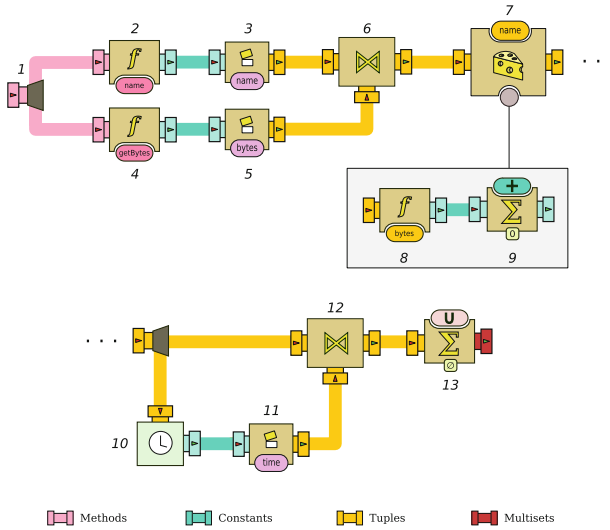


Fig. 4. The BeepBeep processor chain for property BytesWrittenGraph

Figure 4 shows a more informative variant of the `LimitBytesWritten` property. It computes the number of bytes written by each function that does so, and expresses this information in the form of a plot. The processor chain begins by extracting from the trace the methods that perform write operations, discarding all other lines and pairs containing the method name and the number of bytes written are joined in a tuple (processor 6). Processor 7 splits its input stream into multiple distinct streams, each of which contains tuples originating from a single method. This allows the computation of the number of bytes written to be aggregated separately for each method (processors 9 and 10). The remainder of the processor chain aggregates this information with a timestamp, and updates a hash table accordingly. This hash table can then serve as the basis of a plot generated on demand.

5 Experimental Results

We tested this method on traces of length 1,000,000, generated in the manner described above on a Java calculator. Figure 5 plots the execution times (in ms.) for four representative processor chains, namely `NoExec`, `LimitBytesWritten`, `CallSequenceProfiling` and `isAKey`. As can be seen in these results, execution times are largely proportional to the number of processors in each processor chain. Since most security properties require only linear sequencing of processors, their operation can easily be streamlined by merging the operations of multiple monitor in a single class.

Table 1 details the number of processors, number of custom lines of code (not counting code already present in BeepBeep’s template library) and execution

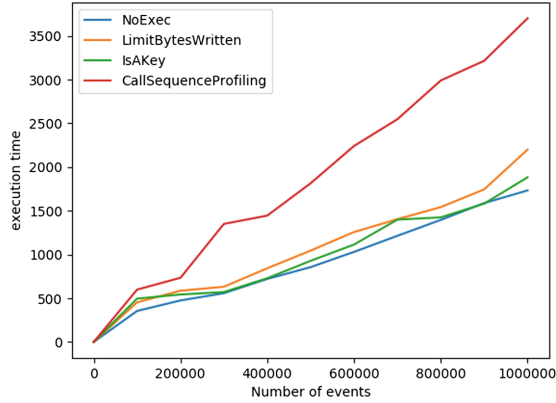


Fig. 5. Experimental Results

time several of the processors mentioned in this paper. This table illustrates the ease with which BeepBeep processors can be composed, often necessitating only a minimal amount of custom code. Once these processor chains are implemented, they can in turn be included as components of processor chains for more complex properties with the addition of a single line of code.

The only processor chain whose execution time is not inconsiderable is `BytesWrittenGraph`, described in the previous section. Much of its execution time is incurred in the final processor, which updates a hash table linking each method to the number of bytes that have been written by that method during the program's current execution. Since a BeepBeep processor chain manipulates events sequentially, and each processor feeds its output to a successor, the final processor of the `BytesWrittenGraph` processor chain performs the update by copying the current hash table before updating it with the information it has received in its last input event. However, since in our particular case, the hash table update is performed in the final processor of the processor chain. As a

Table 1. Description of tested Monitors

Property	No. of processors	Size (lines)	Exec. time (ms)
NoExec	2	6	1590
NoClassLoader	2	6	1636
NoNetwork	2	6	1654
NoReadingFiles	2	6	1699
IsKey	2	8	1883
LimitBytesWritten	3	11	1900
CallSequenceProfiling	8	35	3701
BytesWrittenGraph	13	53	14633

result, it is possible to edit this processor so that it simply updates the hash table, without making a copy. This revision brings `BytesWrittenGraph`'s execution time in line with that of other monitors of its size.

6 Conclusion

In this paper, we showed how the event processor `BeepBeep` can be used for runtime enforcement. The approach is agnostic to the tracer used to generate the trace and itself by the ease by which properties can be stated using `BeepBeep`'s processor chain structure. `BeepBeep` is useful not only for stating and enforcing security properties, but also to generate useful diagnostic information about the trace, as we also illustrated using examples. One avenue of further research which we are currently exploring is to draw on `BeepBeep`'s capabilities to allow us to express a more informative verdict that simply a Boolean indication of the respect/violation of the security property. For instance, the monitor could provide indications as to which parts of the trace caused the violations, rate its severity, and suggest weaker properties that are respected. This information could, in turn, serve as the basis for a more corrective reaction to a potential violation than simply aborting the execution.

References

1. Aafer, Y., Du, W., Yin, H.: DroidAPIMiner: mining API-level features for robust malware detection in Android. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) *SecureComm 2013*. LNCS, vol. 127, pp. 86–103. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-04283-1_6
2. Arp, D., Spreitzenbarth, M., Hubner, M., Gascon, H., Rieck, K.: DREBIN: effective and explainable detection of Android malware in your pocket. In: *NDSS*. The Internet Society (2014)
3. Bedford, A., Garvin, S., Desharnais, J., Tawbi, N., Ajakan, H., Audet, F., Lebel, B.: ANDRANA: quick and accurate malware detection for Android. In: Cuppens, F., Wang, L., Cuppens-Boulahia, N., Tawbi, N., Garcia-Alfaro, J. (eds.) *FPS 2016*. LNCS, vol. 10128, pp. 20–35. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51966-1_2
4. Evans, D., Twyman, A.: Flexible policy-directed code safety. In: *1999 IEEE Symposium on Security and Privacy*, Oakland, California, USA, 9–12 May 1999. pp. 32–45. IEEE Computer Society (1999). <https://doi.org/10.1109/SECPRI.1999.766716>
5. Hallé, S.: When RV meets CEP. In: Falcone, Y., Sánchez, C. (eds.) *RV 2016*. LNCS, vol. 10012, pp. 68–91. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46982-9_6
6. Khanmohammadi, K., Rejali, M.R., Hamou-Lhadj, A.: Understanding the service life cycle of Android apps: an exploratory study. In: *Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM 2015, pp. 81–86. ACM, New York (2015)
7. Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An overview of AspectJ. In: Knudsen, J.L. (ed.) *ECOOP 2001*. LNCS, vol. 2072, pp. 327–354. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45337-7_18

8. Lai, C.: Java insecurity: accounting for subtleties that can compromise code. *IEEE Softw.* **25**(1), 13–19 (2008)
9. Sato, R., Chiba, D., Goto, S.: Detecting Android malware by analyzing manifest files. *Proc. Asia-Pac. Adv. Netw.* **36**, 23–31 (2013)
10. Schneider, F.B.: Enforceable security policies. *ACM Trans. Inf. Syst. Secur.* **3**(1), 30–50 (2000). <https://doi.org/10.1145/353323.353382>
11. Segoufin, L.: Automata and logics for words and trees over an infinite alphabet. In: Ésik, Z. (ed.) *CSL 2006*. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006). https://doi.org/10.1007/11874683_3

Network Security, Encrypted DBs and Blockchain



More Lightweight, yet Stronger 802.15.4 Security Through an Intra-layer Optimization

Konrad-Felix Krentz¹(✉), Christoph Meinel¹, and Hendrik Graupner²

¹ Hasso-Plattner-Institut, University of Potsdam, Potsdam, Germany
{konrad-felix.krentz,christoph.meinel}@hpi.de

² Bundesdruckerei, Berlin, Germany
hendrik.graupner@bdr.de

Abstract. 802.15.4 security protects against the replay, injection, and eavesdropping of 802.15.4 frames. A core concept of 802.15.4 security is the use of frame counters for both nonce generation and anti-replay protection. While being functional, frame counters (i) cause an increased energy consumption as they incur a per-frame overhead of 4 bytes and (ii) only provide sequential freshness. The Last Bits (LB) optimization does reduce the per-frame overhead of frame counters, yet at the cost of an increased RAM consumption and occasional energy- and time-consuming resynchronization actions. Alternatively, the timeslotted channel hopping (TSCH) media access control (MAC) protocol of 802.15.4 avoids the drawbacks of frame counters by replacing them with timeslot indices, but findings of Yang et al. question the security of TSCH in general. In this paper, we assume the use of ContikiMAC, which is a popular asynchronous MAC protocol for 802.15.4 networks. Under this assumption, we propose an Intra-Layer Optimization for 802.15.4 Security (ILOS), which intertwines 802.15.4 security and ContikiMAC. In effect, ILOS reduces the security-related per-frame overhead even more than the LB optimization, as well as achieves strong freshness. Furthermore, unlike the LB optimization, ILOS neither incurs an increased RAM consumption nor requires resynchronization actions. Beyond that, ILOS integrates with and advances other security supplements to ContikiMAC. We implemented ILOS using OpenMotes and the Contiki operating system.

1 Introduction

The major features of the 802.15.4 radio standard are low energy consumption, cheap transceivers, sub-GHz and 2.4-GHz support, and reliable communication thanks to meshing [1]. These features are suitable for implementing wireless sensor and actuator networks. Furthermore, with the advent of 6LoWPAN, an adaption layer for conveying IPv6 packets over 802.15.4 links [14], 802.15.4 is becoming a main choice for implementing Internet of things (IoT) applications.

As for wireless security, many 802.15.4 networks make use of 802.15.4 security. Essentially, 802.15.4 security filters out injected and replayed 802.15.4 frames,

and optionally encrypts the payload of 802.15.4 frames. Specifically, to filter out injected 802.15.4 frames, 802.15.4 security ensures that incoming 802.15.4 frames contain authentic message integrity code (MICs). Both, for generating MICs and for encrypting payloads, 802.15.4 security employs a tweaked version of Counter with CBC-MAC [23]. CCM, in turn, requires a nonce for generating a MIC and encrypting data. 802.15.4 security generates CCM nonces based on incrementing 4-byte frame counters, which 802.15.4 security adds to 802.15.4 frames. Besides, to filter out replayed 802.15.4 frames, 802.15.4 security ascertains that the frame counter of an incoming 802.15.4 frame is greater than that of the last authentic 802.15.4 frame from the sender, thereby providing sequential freshness [20].

Yet, the use of frame counters in 802.15.4 security ensues two drawbacks. First, as a frame counter is added to each 802.15.4 frame, frame transmissions and receptions become more energy consuming as a result. Moreover, since frame counters cut down the maximum payload of 802.15.4 frames, IPv6 packets need to be fragmented at the 6LoWPAN adaption layer more often, thus necessitating additional frame transmissions and receptions. Altogether, frame counters reduce the lifetime of battery-powered 802.15.4 nodes. Second, while frame counters provide sequential freshness, upper layers may require strong freshness. Strong freshness is provided if a receiver can ensure that an incoming 802.15.4 frame was sent within a limited time span prior to its reception [20]. This is particularly desirable if readings from sensors or commands to actuators lose their meaning when being delayed and hence should not be considered fresh.

In order to reduce the per-frame overhead of frame counters, Krentz et al. tailored the Last Bits (LB) optimization to 802.15.4 security [10, 15, 18]. In their version, senders only add the 8 least significant bits (LSBs) of frame counters to outgoing 802.15.4 frames. Nevertheless, receivers can restore higher-order bits using their anti-replay data. On the other hand, to enable receivers to restore higher order bits, each node needs to use a separate frame counter for broadcast frames, as well as separate frame counters for unicast frames for each of its neighbors. Also, every node has to keep track of both the unicast and broadcast frame counter of each of its neighbors. Hence, the LB optimization consumes more RAM than the original anti-replay protection of 802.15.4 security. Moreover, if a node A misses 2^8 unicast or broadcast frames in a row from a neighbor B , A can no longer restore B 's unicast or broadcast frame counters, respectively. To this end, Krentz et al. propose an "UPDATE-UPDATEACK exchange" for resynchronizing frame counters. Unfortunately, this exchange entails sending two unicast frames and is not triggered immediately upon desynchronization, but only delayed.

A seemingly better solution appeared as part of timeslotted channel hopping (TSCH) media access control (MAC) protocol of 802.15.4 [1]. TSCH nodes wake up in certain timeslots so as to receive or transmit data on a certain channel as governed by a schedule. Consequently, TSCH requires network-wide time synchronization. Further, since TSCH requires network-wide time synchronization anyway, TSCH uses implicitly known timeslot indices in lieu of frame counters. This elegantly avoids both mentioned drawbacks with frame counters through an intra-layer optimization. However, Yang et al. outlined various attacks on

TSCH’s mechanisms for time synchronization [25], thus questioning the security of TSCH in general. After all, TSCH’s intra-layer optimization is inapplicable to asynchronous MAC protocols, which work without network-wide time synchronization [13]. A popular asynchronous MAC protocol is, e.g., ContikiMAC [6].

This paper’s main contribution is an Intra-Layer Optimization for 802.15.4 Security (ILOS). ILOS solves both mentioned drawbacks with frame counters by intertwining 802.15.4 security and ContikiMAC. In fact, ILOS reduces the security-related per-frame overhead even more than the LB optimization, as well as achieves strong freshness. Furthermore, unlike the LB optimization, ILOS neither consumes more RAM nor needs resynchronization actions. Beyond that, ILOS integrates with and advances other security supplements to ContikiMAC, namely Krentz et al.’s Adaptive Key Establishment Scheme (AKES) [15], as well as their Practical On-The-fly Rejection (POTR) [17].

The rest of this paper is structured as follows. Section 2 introduces ContikiMAC, as well as security supplements to it. Section 3 specifies the design of ILOS. Section 4 outlines our implementation of ILOS. Section 5 gives an evaluation of ILOS. Lastly, Sect. 6 concludes and suggests topics for future research.

2 Background and Related Work

In ContikiMAC, receivers wake up periodically and perform two clear channel assessments (CCAs). If one of these CCAs indicates a busy channel, receivers stay in receive mode until a frame is received or a timeout occurs, whatever comes first. Senders, on the other hand, repeatedly transmit each frame for a whole wake-up interval, plus once to cover corner cases. This behavior is often called strobing. In the case of unicast frames, senders may stop strobing prematurely if an acknowledgement frame is received in between two consecutive unicast frame transmissions, as shown in Fig. 1. Additionally, to further reduce the time that senders spend in transmit mode, ContikiMAC’s phase-lock optimization schedules the start of a strobe of unicast frames right before the intended receiver wakes up. For this, ContikiMAC’s phase-lock optimization exploits that if an acknowledgement frame is received, the next to last strobed unicast frame must have been transmitted while the receiver woke up. Hence, the time when the transmission of the next to last acknowledged unicast frame began can serve to estimate when the receiver will wake up next. Furthermore, once the wake-up time of a receiver is known, ContikiMAC’s phase-lock optimization no longer strobos unicast frames to that receiver for a whole wake-up interval plus once if no acknowledgement frame returns, but only for a shorter time span plus once. Yet, to account for clock drift, ContikiMAC’s phase-lock optimization relearns the wake-up time of a receiver if unicast transmissions to the receiver tend to fail. This fallback mechanism renders ContikiMAC’s phase-lock optimization susceptible to collision attacks, which provoke longer strobos via jamming [16].

Since its publication, ContikiMAC was improved to support opportunistic routing [7], burst forwarding [8], and channel hopping [2]. In the following, we will however restrict ourselves to introducing security supplements to ContikiMAC since they are fundamental to ILOS [15–17].

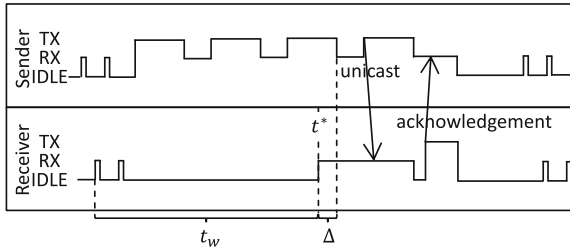


Fig. 1. Operation of a unicast transmission in ContikiMAC

An essential security supplement to ContikiMAC is AKES, which establishes group or pairwise session keys for use in 802.15.4 security [15]. Figure 2 shows the operation of AKES when configured to establish group session keys. A node A initiates session key establishment by broadcasting a HELLO, containing a cryptographic random number R_A . Any receiver B that has not yet established session keys with A also generates a cryptographic random number R_B and, after a random back off period, replies with a HELLOACK. The HELLOACK carries R_B , B 's group session key $K_{B,*}$ encrypted, as well as a MIC. For encrypting $K_{B,*}$ and generating the MIC, B derives a temporary pairwise key $K'_{A,B}$ from a pre-distributed shared secret $K_{A,B}$ between A and B , as well as the two cryptographic random numbers R_A and R_B . Upon receipt of B 's HELLOACK, A decrypts $K_{B,*}$ and checks the MIC by deriving $K'_{A,B}$ analogously. If successful, A acknowledges with an ACK, which includes A 's group session key $K_{A,*}$ encrypted and a MIC. Again, the temporary pairwise key $K'_{A,B}$ serves to generate the ACK's MIC, as well as to encrypt A 's group session key.

Apart from establishing session keys, AKES also deletes neighbors that got out of range. Concretely, if a neighbor sent no fresh authentic frame for a critical period of time, AKES checks if the neighbor is still in range by sending an UPDATE to him, as shown in Fig. 2. If no fresh authentic UPDATEACK returns after a configurable number of retransmissions, AKES deletes that neighbor. Otherwise, if an UPDATEACK returns, AKES extends that neighbor's expiration time. In this regard, ILOS obviates the need for sending UPDATEACKs, which saves energy. Furthermore, thanks to ILOS, AKES no longer needs to keep track of the expiration times of neighboring nodes, which saves RAM.

Despite AKES, 802.15.4 security remains incomplete in the sense that ContikiMAC stays vulnerable to many ding-dong ditching attacks. Ding-dong ditching attacks belong to the larger group of denial-of-sleep attacks, which generally cause an increased energy consumption on victim nodes [4]. Ding-dong ditching attacks, in particular, mislead ContikiMAC nodes into staying more time in receive mode [16]. For example, broadcast and unicast attacks are ding-dong ditching attacks, where an attacker injects or replays broadcast and unicast frames, respectively [4, 16]. Though 802.15.4 security rejects injected and replayed frames, they are still fully received before being rejected, which consumes much energy. As another example of a ding-dong ditching attack, droplet

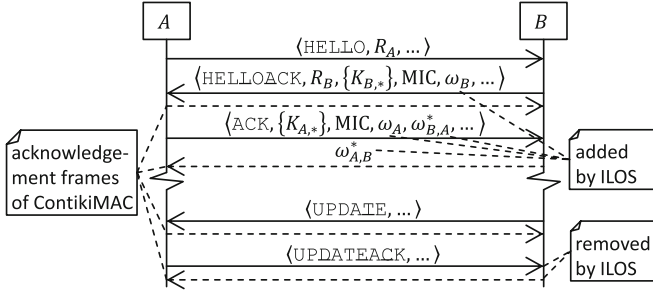


Fig. 2. Adaptive Key Establishment Scheme (AKES) and adaptations to it

attacks exploit that it suffices to transmit an 802.15.4-compliant synchronization header (SHR) plus a few header bytes to keep victim nodes in receive mode for long [11].

A countermeasure against unicast, broadcast, as well as droplet attacks is Krentz et al.’s POTR [17]. The approach of POTR is to cancel the reception of injected and replayed frames early on, similar to what was proposed in related efforts [3, 5, 9, 11, 12, 24]. For this, POTR adapts the headers of 802.15.4 frames like shown in Fig. 3. The Frame Type field encodes the frame’s type. Possible frame types are listed in Table 1. Then, the Source Address field states the sender’s address. Thereafter, the Frame Counter field includes the sender’s 4-byte frame counter. Acknowledgement frames, on the other hand, always just include the 8 LSBs of the frame counter of the frame whose receipt is being acknowledged. Above all, the OTP field contains a one-time password (OTP). POTR validates OTPs during reception by (i) parsing the Frame Type, Source Address, and Frame Counter fields, (ii) deriving an OTP therefrom, and (iii) checking if the derived OTP matches the received one. If they do not match, POTR usually disables the receive mode immediately, which greatly reduces the time that victim nodes stay in receive mode under unicast, broadcast, as well as droplet attacks. Finally, the purpose of the Sequence Number field is to avoid accepting retransmitted frames twice, as we will elaborate on in Sect. 3.4.

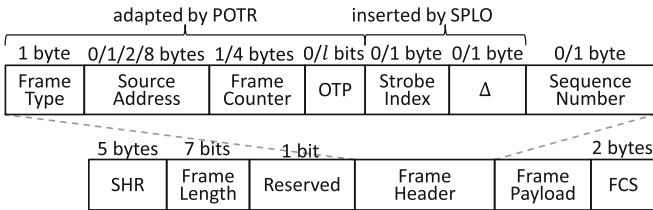


Fig. 3. 802.15.4 frame format as adapted by POTR, SPLO, and ILOS

Table 1. POTR’s frame types and fields present therein

Frame type	Source address	Frame counter	OTP	Strobe index	Δ	Sequence number
Unicast data	✓	✓ (× if using ILOS)	✓	✓	×	✓
Broadcast data	✓	✓ (× if using ILOS)	✓	×	×	×
Acknowledgement	×	✓ (× if using SPLO or ILOS)	×	×	✓	×
HELLO	✓	✓ (× if using ILOS)	✓	×	×	×
HELLOACK	✓	✓ (× if using ILOS)	✓	✓	×	×
ACK	✓	✓ (× if using ILOS)	✓	✓	×	×
Unicast command	✓	✓ (× if using ILOS)	✓	✓	×	✓
Broadcast command	✓	✓ (× if using ILOS)	✓	×	×	×

To shorten POTR’s Frame Counter field and to accelerate POTR’s rejection speed, Krentz et al. suggested using the LB optimization [17]. The LB optimization accelerates POTR’s rejection speed because if the Frame Counter field gets shorter, receivers can validate OTPs earlier. However, as mentioned already, if a node misses a lot of frames, it needs to perform an UPDATE-UPDATEACK exchange. Moreover, in POTR, if a node loses track of the unicast frame counter of a neighbor, both nodes have to establish new session keys with each other for subtle reasons [17]. Such resynchronization actions may also become necessary if an attacker guesses an OTP right, irrespective of using the LB optimization or not [17]. By contrast, ILOS never needs resynchronization actions, even if an attacker guesses an OTP right. Beyond that, ILOS accelerates POTR’s rejection speed slightly more than the LB optimization, as we will show in Sect. 5.

Lastly, the Secure Phase-Lock Optimization (SPLO) protects ContikiMAC’s phase-lock optimization from pulse-delay and collision attacks [16]. Both of these attacks are denial-of-sleep attacks that mislead ContikiMAC’s phase-lock optimization into staying more time in transmit mode. To resist pulse-delay attacks, SPLO ensures the authenticity and timeliness of acknowledgement frames, mainly by adding MICs to acknowledgement frames. This involves inserting the Strobe Index field into unicast frames like shown in Fig. 3. This field indicates how often ContikiMAC strobed a unicast frame already and is incorporated into the CCM nonce of unicast frames, as well as into the CCM nonce of their corresponding acknowledgement frames. To mitigate collision attacks, on the other hand, SPLO limits the maximum duration of a strobe of unicast frames according to the current uncertainty about the wake-up time of the intended receiver. Crucially, unlike ContikiMAC’s original phase-lock optimization, SPLO does not relearn the wake-up time of a receiver if unicast transmissions to the receiver tend to fail. Furthermore, to keep the maximum duration of a strobe of unicast frames below a configurable threshold, SPLO instructs AKES to send an UPDATE to a neighbor that sent no fresh authentic acknowledgement frame for a critical period of time, as well as to delete that neighbor if no UPDATEACK returns. Also, to obtain more precise estimations of wake-up times, SPLO piggybacks Δ on acknowledgement frames. Δ is calculated like shown in Fig. 1 and enables senders of unicast frames to calculate the last wake-up time t^* of a receiver [19].

3 ILOS: Intra-Layer Optimization for 802.15.4 Security

The main idea of ILOS is to replace frame counters with what we call wake-up counters. This necessitates changes to (i) the generation of CCM nonces, (ii) POTR, (iii) anti-replay protection, and (iv) AKES. In the following, we will specify each of these changes. Throughout, we focus on the case of using group session keys, rather than pairwise session keys.

3.1 Notations

Let A and B be adjacent nodes. We denote by:

- t_w ContikiMAC's wake-up interval, as shown in Fig. 1.
- ω_A the wake-up counter of A - A increments ω_A at the rate of t_w in lockstep with ContikiMAC's two regular CCAs. If A skips over doing two CCAs, e.g., due to sending at that time, A must increment ω_A anyway.
- $t_{A,B}^*$ what A stores as the last wake-up time of B - SPLO initializes $t_{A,B}^*$ in parallel to establishing group session keys and updates $t_{A,B}^*$ upon receipt of a timely authentic acknowledgement frame from B .
- $\omega_{A,B}^*$ the wake-up counter of B at time $t_{A,B}^*$ - Likewise, ILOS initializes $\omega_{A,B}^*$ in the course of establishing session keys and updates $t_{A,B}^*$ upon receipt of a timely authentic acknowledgement frame from B .
- ID_A A 's MAC address.
- $K_{A,*}$ A 's group session key.
- α a field in the CCM nonces generated by ILOS.
- λ_A A 's current strobe index - each time a unicast frame is transmitted or retransmitted, λ_A starts over from zero.
- KDF a key derivation function.
- K_n a predistributed network-wide key for use by POTR.

3.2 Adapting CCM Nonces

In order for CCM nonces to be secure, they must never reoccur in conjunction with the same key. ILOS achieves this via two complementary techniques. On the one hand, ILOS uses a common base format and the field α to avoid collisions among CCM nonces of different types of frames. On the other hand, ILOS uses wake-up counters and MAC addresses to avoid collisions among CCM nonces of the same frame type. Concretely, ILOS derives CCM nonces from wake-up counters like shown in Table 2.

Unicast Frames. As for a HELLOACK or ACK from a node A to a node B , ILOS generates the CCM nonce by concatenating ID_A , $\alpha = 0$, λ_A , and ω_A , where ω_A is the wake-up counter of A as A begins to strobe the HELLOACK or ACK. Thus, B needs ω_A to restore the CCM nonce of the HELLOACK or ACK. Therefore, ILOS adds ω_A to the payload of the HELLOACK or ACK, as shown in Fig. 2. It will become apparent that adding wake-up counters to frames is only required

Table 2. CCM inputs as per ILOS

Frame types	CCM nonce	Key
HELLOACK or ACK from A to B	$ID_A \parallel \alpha \parallel \lambda_A \parallel \omega_A$	$K'_{A,B}$
Unicast data or command frame from A to B	$ID_A \parallel \alpha \parallel \lambda_A \parallel \omega_B$	$K_{B,*}$
Acknowledgement frame from B to A	Same as the corresponding unicast frame except that $\alpha = 2$	Same as the corresponding unicast frame
Broadcast data, broadcast command, or HELLO frame from A	$ID_A \parallel \alpha \parallel 0 \parallel \omega_A + 1$	$K_{A,*}$

during session key establishment. The LB optimization is no different in this respect as it requires exchanging certain frame counters in full during session key establishment, too [15]. We also note that ILOS depends on that the back-off period for retransmissions is greater or equal than t_w so that ω_A increments in the meantime. Otherwise, such CCM nonces may reoccur.

As for a unicast data or command frame from a node A to a node B , ILOS generates the CCM nonce by concatenating ID_A , $\alpha = 1$, λ_A , and ω_B , where ω_B is B 's wake-up counter as B receives the unicast frame. Thus, A needs ω_B . However, using $t_{A,B}^*$ and $\omega_{A,B}^*$, A can predict ω_B as $\omega_{A,B}^* + \left\lceil \frac{t_{\text{sched}} - t_{A,B}^*}{t_w} \right\rceil$, where t_{sched} is the time when SPLO schedules the transmission of the unicast frame. This prediction is correct if SPLO keeps the absolute uncertainty about the wake-up time of B below $\frac{t_w}{2}$, which SPLO achieves by default.

Acknowledgement Frames. An acknowledgement frame uses the same CCM nonce as the unicast frame whose receipt is being acknowledged except that α is set to 2.

Broadcast Frames. The CCM nonce of a broadcast frame from a node A is generated by concatenating ID_A , $\alpha = 3$, 0, and $\omega_A + 1$. Here, ω_A is A 's wake-up counter as A begins to strobe. To aid receivers in restoring $\omega_A + 1$, $\omega_A + 1$ needs to be even and A must begin to strobe at $t - \frac{t_w}{2}$, where t is when A increments ω_A next. Thus, A may need to defer the transmission of the broadcast frame until both conditions are met. Yet, this way, a receiver B can restore $\omega_A + 1$ by rounding $\omega_{B,A}^* + \frac{t_{\text{awoke}} - t_{B,A}^*}{t_w}$ to the next even value, where t_{awoke} is when B awoke for doing ContikiMAC's two CCAs that led to receiving the broadcast frame. This restoration of $\omega_A + 1$ is correct (i) if SPLO keeps the absolute uncertainty about the wake-up time of A below $\frac{t_w}{2}$ and (ii) if B wakes up during the interval $[t - \frac{t_w}{2}, t + \frac{t_w}{2}]$. As a side effect of delaying consecutive broadcast transmissions by at least $2t_w$, CCM nonces of broadcast frames can not coincide.

3.3 Adapting the Practical On-the-Fly Rejection (POTR)

As POTR derives OTPs of HELLOACKs and ACKs without frame counters already, ILOS leaves these OTPs unchanged. By contrast, POTR derives the OTPs of

data, command, and HELLO frames from K_n , the sender's group session key, the receiver's address, and the sender's frame counter. To avoid frame counters there too, ILOS calculates the OTP of a unicast data or command frame from A to B as $\text{KDF}(K_n \oplus K_{B,*}, ID_A || \alpha || \omega_B)$, where $\alpha = 1$ and ω_B is B 's wake-up counter when receiving the frame. The purpose of XORing K_n and $K_{B,*}$ is to prevent related-key attacks [17]. Likewise, ILOS calculates the OTP of a broadcast data, broadcast command, or HELLO frame from A as $\text{KDF}(K_n \oplus K_{A,*}, ID_A || \alpha || \omega_A + 1)$, where $\alpha = 3$ and ω_A is A 's wake-up counter as A begins to strobe. As the inputs to KDF resemble the CCM nonces of ILOS, the same methods for predicting, or rather restoring wake-up counters apply.

3.4 Adapting Anti-replay Protection

ILOS provides anti-replay protection as follows.

Unicast Frames. To filter out replayed HELLAOCKs and ACKs, ILOS retains POTR's methods [17].

As for unicast data and command frames, anti-replay protection comes almost as a side effect of generating CCM nonces like ILOS does. This is because a receiver B increments ω_B when waking up. Thus, if B receives a replayed unicast frame, B will assume a CCM nonce that differs from the CCM nonce that was used to secure the replayed unicast frame. Hence, B will reject the replayed unicast frame due to an invalid OTP during reception. Also, even if the OTP of a replayed unicast frame is valid by chance, B will eventually reject the replayed unicast frame due to an inauthentic MIC. However, a subtlety is that the sender A of a unicast data or command frame may miss an acknowledgement frame. In this case, A may retransmit and hence, the receiver B , may accept the same frame twice. This issue also arises in TSCH and POTR, where it may be solved by adding 8-bit sequence numbers to frames [1]. ILOS adopts this solution and adds sequence numbers to unicast data and command frames, as shown in Fig. 3. These sequence numbers are incremented on a per neighbor basis. B discards a unicast data or command frame from A if the contained sequence number matches the one of the previously accepted unicast data or command frame from A . However, although duplicated unicast data and command frames could already be discarded during reception, we opted to fully receive and acknowledge them so as to avoid self-imposed collision attacks.

Acknowledgement Frames. As for acknowledgement frames, anti-replay protection comes indeed by itself. If a sender A receives a replayed acknowledgement frame, A will use a different CCM nonce to check if the MIC of the replayed acknowledgement frame is authentic. Consequently, A will consider the MIC of the replayed acknowledgement frame inauthentic and reject the replayed acknowledgement frame.

Broadcast Frames. As for broadcast frames, anti-replay protection does not come automatically in two occasions. First, it may happen that a receiver B

receives a broadcast frame from a sender A and, during the next wake up of B , an attacker replays that same broadcast frame. In this case, B may assume the same CCM nonce, thus causing B to consider both the OTP and the MIC of the replayed broadcast frame valid. Only if there is one wake up in between, B will definitely assume a different CCM nonce and hence reject the replayed broadcast frame due to an invalid OTP or an inauthentic MIC. This also holds true if B updates $t_{B,A}^*$ and $\omega_{B,A}^*$ in between since, in this case, $\omega_{B,A}^*$ is raised, which causes B to restore a different CCM nonce, too. Altogether, ILOS merely needs to take care of not accepting a broadcast frame if, during the last wake up, a broadcast frame from the same sender was accepted already. Like POTR, ILOS does so already during reception to counter ding-dong ditching. Second, when a sender retransmits a broadcast frame because of collision avoidance (CA), a receiver may receive that frame twice. This issue is specific to ContikiMAC since, in TSCH, CA is only done before transmitting, whereas ContikiMAC also does CA in between strobed frames. Neither does this issue arise in POTR since POTR simply does not increment the frame counter when retransmitting broadcast frames, causing duplicated frames to be considered as replayed and hence rejected. To solve this issue, ILOS requires ContikiMAC to only do CA before strobing a broadcast frame. While this only offers intra-network CA, it has the security benefit that jamming during broadcast transmissions no longer induces retransmissions, fixing an open denial-of-sleep vulnerability.

3.5 Adapting the Adaptive Key Establishment Scheme (AKES)

Let A and B be adjacent nodes. To initialize $\omega_{B,A}^*$ and $\omega_{A,B}^*$ while A and B establish session keys, ILOS adds additional data to ACKs, as well as to acknowledgement frames that are sent in response to ACKs, as shown in Fig. 2. Specifically, to initialize $\omega_{B,A}^*$, A reports back on its wake-up counter at time of receiving B 's HELLOACK since SPLO initializes $t_{B,A}^*$ to the time when A woke up for receiving B 's HELLOACK. Likewise, B reports back on its wake-up counter when receiving A 's ACK since SPLO initializes $t_{A,B}^*$ to when B woke up for receiving A 's ACK.

In addition, ILOS applies two tweaks to AKES' transmission of UPDATES. First, rather than sending dedicated UPDATEACKs, ILOS relies on SPLO's authenticated acknowledgement frames, which are sent in response to UPDATES anyway. Second, it is also somewhat redundant that SPLO stores the last known wake-up time of each neighboring node while AKES additionally keeps track of each neighbor's expiration time. Hence, ILOS solely relies on SPLO to schedule the transmission of UPDATES, thereby freeing AKES from storing expiration times.

4 Implementation

Our implementation of ILOS advances Krentz et al.'s implementation of 802.15.4 security, AKES, POTR, ContikiMAC, as well as SPLO for the Contiki operating system [15–17]. We preserved their design and inserted the changes of ILOS surrounded by conditional preprocessor directives. This enables us to switch

between frame counters and wake-up counters at compilation time. Within our conditional code, we only let information flow downwards, following the terminology introduced in [21]. That is, upper layers retrieve additional information, such as the current wake-up counter, from lower layers. Nevertheless, ILOS is not a real cross-layer optimization because ILOS only affects the MAC layer.

5 Evaluation

Below, we (i) argue that ILOS achieves strong freshness, (ii) quantify the reduction of the security-related per-frame overhead due to ILOS, (iii) demonstrate the resulting reduction in energy consumption, (iv) show that ILOS accelerates POTR’s rejection speed, and (v) give insight to the RAM footprint of ILOS.

5.1 Freshness Guarantees

Recall that strong freshness is provided if a receiver can ensure that an incoming 802.15.4 frame was sent within a limited time span prior to its reception [20]. In the case of unicast data and command frames, an upper bound that ILOS achieves is t_w . This is because, if a unicast data or command frame is delayed by $\geq t_w$, receivers will use a different CCM nonce to verify its MIC, which results in the rejection of the frame. In the case of acknowledgement frames, this upper bound is even lower because an acknowledgement frame is only considered timely if it belongs to the unicast frame that was just strobed. Additionally, SPLO only accepts acknowledgement frames within a short window after sending a unicast frame. This way, SPLO ensures that acknowledgement frames are not delayed by more than 0.122 ms by default [16]. In the case of broadcast frames, ILOS achieves an upper bound of $2t_w$ since receivers will definitely assume a different CCM nonce when a broadcast frame is delayed by $\geq 2t_w$. On the other hand, delayed HELLOACKs and ACKs get accepted, but session key establishment ultimately fails as a result since SPLO aborts session key establishment if HELLOACKs and ACKs are not being acknowledged in a timely manner [16].

5.2 Security-Related Per-Frame Overhead

Table 3 compares the security-related per-frame overhead of various frame formats. According to the original frame format of 802.15.4, a secured 802.15.4 frame comprises a 1-byte Security Control field, a 4-byte Frame Counter field, an optional 1-byte Key Index field, an optional Key Source field of up to 8 bytes, as well as an m -bit CCM-MIC [1]. In TSCH networks, the Frame Counter field becomes unnecessary, as described in the introduction [1]. In ContikiMAC networks, using AKES obviates the need for the Key Index, as well as the Key Source field [15]. In addition, the LB optimization reduces the overhead of frame counters to 1 byte. On the other hand, POTR requires adding an l -bit OTP field to non-acknowledgement frames and a 1-byte sequence number to unicast frames. Moreover, SPLO requires adding the 1-byte Strobe Index field to unicast frames. ILOS reduces the security-related per-frame overhead again by dispensing with frame counters altogether.

Table 3. Security-related per-frame overhead

Frame format	Overhead (in bytes)
802.15.4 [1]	$[5, 13] + \frac{m}{8}$
TSCH [1]	$[1, 9] + \frac{m}{8}$
802.15.4 + AKES + LB [15]	$1 + \frac{m}{8}$
POTR + AKES [17]	$[4, 5] + \frac{l+m}{8}$
POTR + AKES + LB [17]	$[1, 2] + \frac{l+m}{8}$
POTR + AKES + SPLO + LB [16]	$[1, 3] + \frac{l+m}{8}$
POTR + AKES + SPLO + ILOS	$[0, 2] + \frac{l+m}{8}$

5.3 Energy Efficiency

To demonstrate that ILOS reduces the energy consumption of sending unicast data frames, the following experiment was conducted. An OpenMote *A* sent a unicast data frame with 50 bytes of payload to another OpenMote *B* [22]. *A* and *B* were recently synchronized so that *A* only strobed twice. While *A* strobed and subsequently received *B*'s acknowledgement frame, the current draw of *A* was measured by connecting *A*, a μ Current Gold, and a Rigol DS1000E oscilloscope in series, as is further detailed in [22]. The current draw over time was then converted into the actual energy consumption under the assumption of a constant supply voltage of 3 V. This was repeated using three different configurations, namely (i) with the LB optimization, as well as ILOS disabled, (ii) with the LB optimization enabled, and (iii) with ILOS enabled. For each of the three configurations, 100 samples were obtained. POTR, SPLO, and 8-byte addresses were used throughout.

Figure 4a depicts the results as boxplots. Expectably, the most energy-consuming configuration is to use neither the LB optimization nor ILOS. This is because, in this configuration, frame counters are transmitted uncompressed. Enabling the LB optimization saves energy since the security-related per-frame overhead decreases by 3 bytes. Another byte can be saved by enabling ILOS instead, yielding a slightly lower energy consumption compared to using the LB optimization.

To also demonstrate that ILOS reduces the energy consumption of receiving unicast data frames, the above experiment was repeated with two differences. First, *B*'s energy consumption while receiving *A*'s unicast data frames and acknowledging them was measured. Second, to avoid bias, *B* woke up at randomized times. Throughout, the dozing optimization for ContikiMAC was switched on [16].

Figure 4b shows the results. This time, the variation in the data is much higher because the energy consumption per frame reception highly depends on how long a receiver waits until the next unicast frame is being strobed. Apart from that, the results are similar. While ILOS constitutes the most energy-efficient configuration, using neither the LB optimization nor ILOS constitutes the least energy-efficient configuration.

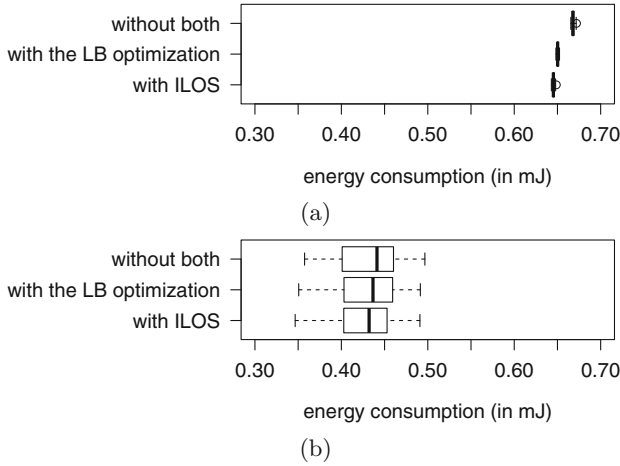


Fig. 4. Energy consumption per (a) transmission and (b) reception of a unicast data frame with 50 bytes of payload

The above results apply to unicast command and data frames alike since they are treated equally. Broadcast receptions should also consume less energy when using the LB optimization or ILOS. Broadcast transmissions, by contrast, will not become more energy efficient since ContikiMAC strobcs broadcast frames for a whole wake-up interval anyway. Additionally, we note that ILOS may also reduce the frequency of fragmentation, which then saves further energy.

5.4 Rejection Speed

To compare the rejection speed of POTR in different configurations, the following experiment was conducted. An OpenMote *A* sent a unicast data frame with an invalid OTP to another OpenMote *B* [22]. Upon receipt, *B* stopped the time between detecting the frame’s SHR and the rejection of the frame. This was repeated using three different configurations, namely (i) with the LB optimization, as well as ILOS disabled, (ii) with the LB optimization enabled, and (iii) with ILOS enabled. For each of the three configurations, 100 samples were obtained. Throughout, 8-byte addresses were used.

Figure 5 shows that the LB optimization accelerates POTR’s rejection speed noticeably. This was expected, as explained in Sect. 2. ILOS, by comparison,

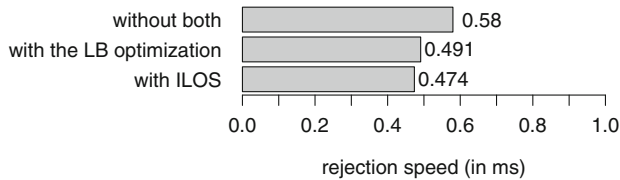


Fig. 5. Mean rejection speed of POTR

accelerates POTR’s rejection speed a bit more since the validation of OTP begins even earlier. In effect, ILOS further reduces the time spent in receive mode under ding-dong ditching.

5.5 RAM Footprint

To measure the RAM footprint, the tool `arm-none-eabi-size` was used. As a baseline for comparison, the RAM footprint when disabling 802.15.4 security altogether was measured. Based on this measure, the overhead in RAM was then determined when (i) using neither the LB optimization nor ILOS, (ii) enabling the LB optimization, and (iii) enabling ILOS. Also, the RAM footprint was determined when configuring the Contiki operating system to use 0, 5, 10, 15, 20, or 25 neighbor slots.

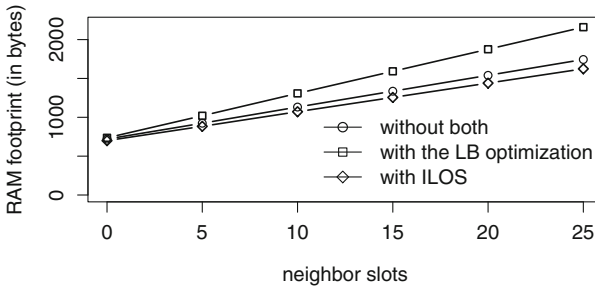


Fig. 6. RAM footprint

Figure 6 shows the results. Surprisingly, ILOS consumes even less RAM than if using neither the LB optimization nor ILOS. There are two main reasons for this. First, ILOS reuses the wake-up times that are stored by SPLO anyway. Second, ILOS frees AKES from storing expiration times. Conversely, the LB optimization incurs a high RAM overhead, as conjectured in the introduction. This is problematic since 802.15.4 nodes usually have just a few kilobytes of RAM.

6 Conclusions and Future Work

Using frame counters incurs drawbacks in terms of energy efficiency and freshness guarantees. TSCH avoids these drawbacks by replacing frame counters with timeslot indices. However, TSCH’s mechanisms for time synchronization are vulnerable to a range of attacks. ContikiMAC, on the other hand, avoids many of TSCH’s vulnerabilities in the first place since ContikiMAC works asynchronously. Yet, as far as ContikiMAC is concerned, only the LB optimization is currently available for alleviating the drawbacks of frame counters. To address

the limitations of the LB optimization, we have proposed ILOS. According to our evaluation, ILOS outperforms the LB optimization in terms of security-related per-frame overhead, energy efficiency, rejection speed, as well as RAM footprint. Additionally, ILOS has three major advantages over the LB optimization. First, ILOS achieves strong freshness. Second, ILOS avoids resynchronization actions. Third, ILOS simplifies AKES. The only drawback of ILOS seems to be that ILOS intertwines ContikiMAC and 802.15.4 security. From a software engineer's perspective, we would actually like to decouple the implementation of 802.15.4 security and ContikiMAC so that we can change one without affecting the other. Future work may generalize ILOS to other MAC protocols or formally assess the correctness of ILOS, e.g., with a protocol verification tool.

References

1. IEEE Standard 802.15.4 (2015)
2. Al Nahas, B., Duquennoy, S., Iyer, V., Voigt, T.: Low-power listening goes multi-channel. In: Proceedings of the 2014 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), pp. 2–9. IEEE (2014)
3. Aljareh, S., Kavoukis, A.: Efficient time synchronized one-time password scheme to provide secure wake-up authentication on wireless sensor networks. *Int. J. Adv. Smart Sens. Netw. Syst. (IJASSN)* **3**, 1–11 (2013)
4. Brownfield, M., Gupta, Y., Davis, N.: Wireless sensor network denial of sleep attack. In: Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop (IAW 2005), pp. 356–364. IEEE (2005)
5. Capossele, A.T., Cervo, V., Petrioli, C., Spenza, D.: Counteracting denial-of-sleep attacks in wake-up-based sensing systems. In: Proceedings of the IEEE International Conference on Sensing, Communication and Networking (SECON 2016), pp. 1–9. IEEE (2016)
6. Dunkels, A.: The ContikiMAC radio duty cycling protocol. Technical report T2011:13. Swedish Institute of Computer Science (2011)
7. Duquennoy, S., Landsiedel, O., Voigt, T.: Let the tree bloom: scalable opportunistic routing with ORPL. In: Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems (SenSys 2013), pp. 2:1–2:14. ACM (2013)
8. Duquennoy, S., Österlind, F., Dunkels, A.: Lossy links, low power, high throughput. In: Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys 2011), pp. 12–25. ACM (2011)
9. Falk, R., Hof, H.J.: Fighting insomnia: a secure wake-up scheme for wireless sensor networks. In: Proceedings of the Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2009), pp. 191–196 (2009)
10. Gouda, M.G., Choi, Y.R., Arora, A.: Antireplay protocols for sensor networks. In: Wu, J. (ed.) *Handbook on Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wireless, and Peer-to-Peer Networks*, pp. 561–574. CRC (2005)
11. He, Z., Voigt, T.: Droplet: a new denial-of-service attack on low power wireless sensor networks. In: Proceedings of the 2013 IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS 2013), pp. 542–550. IEEE (2013)
12. Hsueh, C.T., Wen, C.Y., Ouyang, Y.C.: A secure scheme against power exhausting attacks in hierarchical wireless sensor networks. *IEEE Sens. J.* **15**(6), 3590–3602 (2015)

13. Huang, P., Xiao, L., Soltani, S., Mutka, M., Xi, N.: The evolution of MAC protocols in wireless sensor networks: a survey. *IEEE Commun. Surv. Tutor.* **15**(1), 101–120 (2013)
14. Hui, J., Thubert, P.: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks. RFC 6282 (2011). Updates RFC 4944
15. Krentz, K.F., Meinel, C.: Handling reboots and mobility in 802.15.4 security. In: *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC 2015)*, pp. 121–130. ACM (2015)
16. Krentz, K.F., Meinel, C., Graupner, H.: Countering three denial-of-sleep attacks on ContikiMAC. In: *Proceedings of the International Conference on Embedded Wireless Systems and Networks (EWSN 2017)*, pp. 108–119. Junction (2017)
17. Krentz, K.F., Meinel, C., Schnjakin, M.: POTR: practical on-the-fly rejection of injected and replayed 802.15.4 frames. In: *Proceedings of the International Conference on Availability, Reliability and Security (ARES 2016)*, pp. 59–68. IEEE (2016)
18. Luk, M., Mezzour, G., Perrig, A., Gligor, V.: MiniSec: a secure sensor network communication architecture. In: *Proceedings of the 6th International Conference on Information Processing in Sensor Networks (IPSN 2007)*, pp. 479–488. ACM (2007)
19. Michel, M., Voigt, T., Mottola, L., Tsiftes, N., Quoitin, B.: Predictable MAC-level performance in low-power wireless under interference. In: *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks (EWSN 2016)*, pp. 13–22. Junction (2016)
20. Raymond, D.R., Marchany, R.C., Midkiff, S.F.: Scalable, cluster-based anti-replay protection for wireless sensor networks. In: *Proceedings of the IEEE SMC Information Assurance and Security Workshop (IAW 2007)*, pp. 127–134. IEEE (2007)
21. Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. *IEEE Commun. Mag.* **43**(12), 112–119 (2005)
22. Vilajosana, X., Tuset, P., Watteyne, T., Pister, K.: OpenMote: open-source prototyping platform for the industrial IoT. In: Mitton, N., Kantarci, M.E., Gallais, A., Papavassiliou, S. (eds.) *ADHOCNETS 2015*. LNICSSITE, vol. 155, pp. 211–222. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25067-0_17
23. Whiting, D., Housley, R., Ferguson, N.: Counter with CBC-MAC (CCM). RFC 3610 (2003)
24. Wood, A., Stankovic, J., Zhou, G.: DEEJAM: defeating energy-efficient jamming in IEEE 802.15.4-based wireless networks. In: *Proceedings of the 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON 2007)*, pp. 60–69. IEEE (2007)
25. Yang, W., Wang, Q., Qi, Y., Sun, S.: Time synchronization attacks in IEEE802.15.4e networks. In: *Proceedings of the International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI 2014)*, pp. 166–169. IEEE (2014)



ObliviousDB: Practical and Efficient Searchable Encryption with Controllable Leakage

Shujie Cui^(✉), Muhammad Rizwan Asghar, Steven D. Galbraith,
and Giovanni Russello

The University of Auckland, Auckland, New Zealand
scui379@aucklanduni.ac.nz

Abstract. Searchable encryption allows users to execute encrypted queries over encrypted databases. Several encryption schemes have been proposed in the literature but they leak sensitive information that could lead to inference attacks. We propose *ObliviousDB*, a searchable encryption scheme for an outsourced database that limits information leakage. Moreover, our scheme allows users to execute SQL-like queries on encrypted data and efficiently supports multi-user access without requiring key sharing. We have implemented *ObliviousDB* and show its practical efficiency.

1 Introduction

Cloud computing is a successful paradigm offering companies virtually unlimited data storage and computational power at very attractive costs. Despite its benefits, cloud computing raises new challenges for protecting data.

Motivation. Once the data is outsourced to the cloud environment, the data owner lacks a valid mechanism for protecting the data from unauthorised access. This poses serious confidentiality and privacy concerns to the outsourced data. To mitigate this problem, the hybrid cloud computing approach is getting more popular among large enterprises [1,2]. In a hybrid cloud approach, the organisation maintains sensitive data and services within their infrastructure and outsources the rest to a public cloud. However, identifying sensitive assets is not an easy task and once the data and services leave the internal infrastructure, there is a risk of compromising the confidentiality of the assets if no proper security mechanisms have been put in place.

Problem. In recent years, Searchable Encryption (SE) schemes have been proposed to partially overcome the confidentiality issue in cloud computing. These schemes allow the cloud to perform encrypted search operations on encrypted data. Most of them focus on improving the search efficiency and functionality. A thorough survey with a comparative analysis of existing SE schemes can be found in our recent work [3].

Unfortunately, researchers have paid little attention to the information the cloud provider can learn during search and match operations even if performed on encrypted data. Some recent works [4–6] have shown that even a minor leakage can be exploited to learn sensitive information and break the scheme.

In [5], Naveed *et al.* recover a vast majority of data in CryptDB [7] by using frequency analysis. Zhang *et al.* [6] further investigate the consequences of leakage by injecting chosen files or records into the encrypted database. Based on the information learned by looking at which encrypted data is accessed by a given query, referred to as *access pattern* leakage, they could recover a very high fraction of the searched keywords by injecting a small number of known files or records into the database. The cloud provider can also infer if two or more queries are equivalent or not, referred to as *search pattern* leakage. A recent study by Cash *et al.* [4] also shows that given small leakage a determined attacker (including a malicious cloud provider) could break the encryption scheme.

Matters are even worse for *dynamic* SE schemes where insert and delete operations are also supported. Most of the dynamic SE schemes do not support *forward privacy* and *backward privacy* properties. Lacking forward privacy means that the cloud provider can learn if newly inserted data or updated data matches previously executed queries; lack of support for backward privacy means that the cloud provider can learn if deleted data matches new queries. Basically without forward and backward privacy, a cloud provider executing a dynamic SE scheme is able to learn the evolution of the data over time. Only a few of the existing dynamic schemes [8–10] support forward privacy, but no scheme is able to support both properties simultaneously.

A possible solution could be to employ Oblivious Random Access Memory (ORAM) or Private Information Retrieval (PIR) schemes. However, current ORAM and PIR schemes are prohibitively costly and impractical.

Our Solution. In this paper, we present *ObliviousDB*, an SE scheme for databases for hybrid cloud environments, that is able to overcome all the issues discussed above.

Based on proxy-encryption given in [11], *ObliviousDB* is an encrypted search scheme that supports the full-fledged multiple user management. Moreover, *ObliviousDB* exploits the hybrid cloud computing approach and minimises information leakage. In our approach, the organisation is not required to make decisions on how to split its data between the private and public infrastructure. The public infrastructure is used for storing all the data while the private infrastructure is used mainly for running our *Oblivious Proxy Service (OPS)*, a proxy service for maintaining metadata information about the data stored in the public infrastructure.

The OPS plays a major role in ensuring the confidentiality of the data and manages the data structures for achieving search efficiency. In terms of its functionality, the OPS is similar to the proxy server used in CryptDB [7]. However, unlike CryptDB, we have designed the OPS to be robust against attacks *i.e.*, a compromised OPS will not reveal sensitive data to adversaries.

Contributions. This paper makes the following novel contributions:

1. *ObliviousDB* minimises the information leaked to the cloud provider when executing queries by (i) dynamically re-randomising the encrypted data, (ii) shuffling the locations of records within the database, and (iii) introducing and varying a random number of dummy records, necessary for achieving *search and access pattern privacy*.
2. To achieve *operation pattern privacy*, where the cloud server is not able to distinguish between select, insert, delete and update queries, the OPS obfuscates the actual operations executed by the users by inserting additional queries and combining queries into the shuffle operation.
3. *ObliviousDB* supports both forward and backward privacy by randomising data and query through the use of fresh nonces. In this way, even if the cloud provider stores a search query, it cannot be matched with new data. Likewise, new queries cannot be executed over deleted records.

To show the feasibility of our approach, we have implemented *ObliviousDB* and measured its performance.

2 Overview of *ObliviousDB*

In the remainder of the paper, we set the context and informally describe the properties used in our categorisation. **Search Pattern Privacy (SPP)** refers to the property where the cloud provider is not able to distinguish if two (or more) queries are the same or not. **Access Pattern Privacy (APP)** means the cloud provider is unable to infer if two (or more) result sets contain the same records or not. **Size Pattern Privacy (SzPP)** is achieved if the cloud provider is unable to learn the size of returned (real) records. **Operation Pattern Privacy (OPP)** is achieved if the cloud provider is unable to discover if the issued query is select, insert, delete or update. **Forward Privacy** means the cloud provider does not learn if a new or updated record matches a query executed in the past. **Backward Privacy** means the cloud provider is unable to executed queries on records that have been deleted or modified.

2.1 System Model

The system involves five main entities shown in Fig. 1:

- **Database Administrator (DBA):** A DBA is responsible for management of the database, its users and access control policies for regulating access to tables.
- **Database User (DBU):** It represents an authorised user who can execute select, insert, update and delete queries over encrypted data. After executing encrypted queries, a DBU can retrieve the result set, if any, and decrypt it.

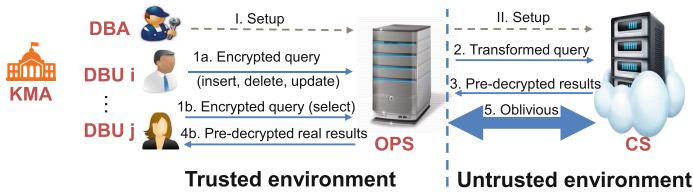


Fig. 1. Overview of *ObliviousDB*: A DBA is responsible for running setup (Step I then Step II). A DBU can insert, delete and update the data (Step 1a) or execute a select query (Step 1b) to receive matching records (Step 4b). Regardless of the query type, to control information disclosure, the OPS transforms the query (Step 2) to perform the search (Step 3) followed by an oblivious algorithm (Step 5).

- **Oblivious Proxy Server (OPS):** It provides greater security and search efficiency. It serves as a proxy between DBUs and the cloud server. In order to hide sensitive information about queries, it pre-processes the queries submitted by the DBU. It also filters out dummy records from the result set returned to the DBU. To improve performance, it manages some indexing information. Technically, the OPS is part of the private cloud in the hybrid cloud environment, which is linked with a more powerful public cloud infrastructure.
- **Key Management Authority (KMA):** This entity is responsible for generating keying material once a new DBU joins the system. Furthermore, the KMA removes the DBU, when she is compromised or no longer part of the database.
- **Cloud Server (CS):** A CS is part of the public cloud infrastructure provided by a cloud service provider. It stores the encrypted data and access control policies and enforces those policies to regulate access to the data.

Threat Model. We assume that the KMA is fully trusted. The KMA does not need to be online all the times. In particular, it has to be online only when the system is initialised, a new DBU is created or an existing one is removed from the system. In this way, the organisation can easily secure the KMA from external attacks. DBUs are only considered to keep their keys (and decrypted data) securely.

We consider that the CS is honest-but-curious. More specifically, the CS would honestly perform the operations requested by the DBA and DBUs according to the designated protocol specification; however, it is curious to analyse the stored and exchanged data so as to learn additional information. We assume that the CS will not mount active attacks, such as modifying the message flow or denying access to the database.

The OPS is deployed in the private cloud, which is owned by the organisation. Hence, we assume the OPS is trusted. However, it is responsible for communicating with the external world. Thus, it could be the target of attackers and get compromised, which means the data stored on the OPS could possibly be exposed to attackers.

In this work, we assume that there are mechanisms in place for data integrity and availability. Last but not least, access policy specification is out of the scope of this paper, but the approach introduced in [11,12] can be utilised in *ObliviousDB*.

3 Solution Details

3.1 Data Representation

Table 1 illustrates an example of how we represent and store the data on the OPS and CS. Let us assume that we have a table Staff (Table 1(a)) containing *Name* and *Age* fields. The CS stores an encrypted version of this, which is *EDB* and illustrated in Table 1(c), where each data element is encrypted under Data Encryption (DE) and Searchable Encryption (SE), where DE ensures the confidentiality of the retrievable data, and SE makes the data searchable (the implementation details are given in Algorithm 1). Similarly, we encrypt each value in the table.

To improve search efficiency and reduce the communication overhead, we support indexing. Technically, we divide the data into groups and build an index for each group maintained by the OPS. When a query is received, the OPS sends to the CS the corresponding list of indices to be searched. Table 1(b), called *GDB*, shows an example of the group information. Note that each field of the database corresponds to a different group, so that in a complex query the OPS would identify all the groups corresponding to fields in the query and send to the CS the union or intersection of the indices (depending on whether the query is a disjunction or conjunction). The group identifiers are concealed with *GE* (the detail is given in Algorithm 1, Sect. 3.3). The secret key for *GE* is only known to DBUs. This means that if the OPS gets compromised then an attacker is unable to learn the actual data items that correspond to a group.

SE and DE representations do not leak information about encrypted values. However, the CS can easily learn the number of matching records during the search process. In *ObliviousDB*, the OPS adds dummy records. Note that the

Table 1. Data representation on each entity.

(a) Staff		(b) GDB			(c) EDB				
Name	Age	GID	Nonce	Index List	ID	{Name}_{SE}	{Name}_{DE}	{Age}_{SE}	{Age}_{DE}
Alice	25	g_1	n_{20}	{1, 3, 4}	1	$SE_{n_a}(Alice)$	$DE(Alice)$	$SE_{n_{20}}(25)$	$DE(25)$
Anna	30	g_2	n_{30}	{2}	2	$SE_{n_a}(Anna)$	$DE(Anna)$	$SE_{n_{30}}(30)$	$DE(30)$
Bob	27	g_3	n_a	{1, 2, 4}	3	$SE_{n_b}(Bob)$	$DE(Bob)$	$SE_{n_{20}}(27)$	$DE(27)$
		g_4	n_b	{3}	4	$SE_{n_a}(Alice)$	$DE(xyz)$	$SE_{n_{20}}(25)$	$DE(13)$

Table (a) is a sample table viewed by DBUs. Table (b) is the group information stored on the OPS. We have $g_1 = GE(25) = GE(27)$, $g_2 = GE(30)$, $g_3 = GE(Alice) = GE(Anna)$ and $g_4 = GE(Bob)$. The group ID is encrypted, since the OPS could be compromised. Each group has a nonce to ensure forward and backward privacy and a list of IDs indicating the records in the group. The CS stores Table (c), where each value is encrypted with *SE* and *DE* for data search and retrieval, respectively. Each *SE* value is bound with the nonce of its group. The last record, consisting of normal *SE* and fake *DE* parts, is dummy.

CS is not able to distinguish between a dummy and a real record. To make sure that the dummy records match with the queries generated by DBUs, the OPS generates dummy records such that the SE fields correspond to real data values. Specifically, the OPS generates dummy records by sampling SE terms from the set of real records in the corresponding group. The OPS also maintains a list of indices that contain dummy records (not shown in Table 1(b)); this could be an N -bit string $flags$, where N is the total size of the database on the CS. We have $flags[id] = 0$ or 1 if the record is a dummy or real record, respectively.

To ensure that dummy records are not delivered to the DBU, the OPS filters them out from the search result. A large number of dummy records can ensure a high level of privacy but at the cost of poor performance since the CS has to search over more records and the OPS has to filter out more dummy records before returning the result to the DBU. For controlling dummy records, the DBA sets a threshold t at the initialisation time, which is the ratio between dummy and real records. In reality, the value of t can be set according to the practical requirements for security and performance, and depending on the type of data being stored in the database. For example, in a different context, Cash *et al.* [4] suggested taking $t = 0.6$ to resist against size pattern based attacks on the Enron emails dataset.

ObliviousDB achieves both forward and backward privacy. That is, even if the CS holds old queries, they cannot be matched with new records. Similarly, if the CS holds deleted records, they cannot be matched with new queries. To achieve both properties, the OPS re-encrypts each record and query with nonces. Nonces are generated and maintained on groups, meaning all the records with the same group are under the same nonce. When a query is executed over a given group, the OPS will generate a new nonce and the data will be updated accordingly. The nonces are not revealed to the CS. The OPS maintains the information between groups and nonces using the *GDB* table, as shown in Table 1(b).

3.2 Setup

The system is set up by the KMA by taking as input a security parameter λ . The output is a prime number p , three multiplicative cyclic groups \mathbb{G}_1 , \mathbb{G}_2 and \mathbb{G}_T of order p , such that there is a “Type 3” bilinear map [13] $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, which has the properties of *bilinearity*, *computability* and *non-degeneracy*, but there is no symmetric bilinear map defined on \mathbb{G}_1 alone or \mathbb{G}_2 alone. Let g_1 and g_2 be the generators of \mathbb{G}_1 and \mathbb{G}_2 , respectively. The KMA chooses a random x from \mathbb{Z}_p and returns $h = g_1^x$. Next, it chooses a collision-resistant keyed hash function H , a pseudorandom functions f and a random key s for f . It also initialises the key store managed by the CS. That is, $K_S \leftarrow \phi$. Finally, it publishes the public parameters $Params = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, h, H, f)$ and keeps securely the master secret key $MSK = (x, s)$.

Building on top of proxy encryption [11, 14], *ObliviousDB* supports multi-user access with efficient DBU registration and revocation. Specifically, when the DBU i joining the system, the KMA splits MSK into two values x_{i1} and x_{i2} , where $x = x_{i1} + x_{i2} \pmod p$ and $x_{i1}, x_{i2} \in \mathbb{Z}_p$. Then, the KMA transmits

$K_{U_i} = (x_{i1}, s)$ and $K_{S_i} = (i, x_{i2})$ securely to the DBU i and the CS, respectively. The CS adds K_{S_i} to its key store: $K_S \leftarrow K_S \cup K_{S_i}$. With K_{U_i} , DBU i could issue a query. For revoking a DBU, we just need to remove K_{S_i} on the CS.

Algorithm 1 *Query*(Q)

```

1:  $DBU_i(Q)$ :
2: for each data element  $d$  in Query  $Q$  do
3:    $\sigma \leftarrow f_s(d)$ 
4:    $GE(d) \leftarrow LSB_k(\sigma)$  {the least significant  $k$  bits of  $\sigma$ }

5:    $r \leftarrow Z_p^*, SE(d) \leftarrow (c_1 = g_1^r, c_2 = g_2^{sr})$ 
6:    $r \leftarrow Z_p^*, DE(d) \leftarrow (e_1 = g_2^r, e_2 = h^r d)$ 
7: for each keyword  $k$  in WHERE-clause of  $Q$  do
8:    $\sigma \leftarrow f_s(k)$ 
9:    $GE(k) \leftarrow LSB_k(\sigma)$ 
10:   $r \leftarrow Z_p^*, SE(k) \leftarrow (t_1 = g_2^r, t_2 = g_2^{sr})$ 
11: Send the encrypted query  $EQ = (SE(Q), DE(Q))$  and
    its group information  $GQ = GE(Q)$  to the OPS

12:  $OPS(EQ, GQ)$ :
13:  $IL \leftarrow \emptyset$ 
14: Cache a copy of  $EQ$ 
15: if  $EQ$  is an insert query then
16:   Generate a fake select query as  $EQ$ 
17: if  $EQ$  is a delete or update query then
18:   Change the type of  $EQ$  into select
19: for each  $GE(k) \in GQ$  and  $SE(k) \in EQ$  do
20:    $(n, il) \leftarrow GDB(GE(k))$ 

21:    $IL \leftarrow IL * il$ , where  $*$  is the conjunction in  $GQ$ 
22:    $SE_n(k) \leftarrow (t_1 \leftarrow t_1^n = g_2^{rn}, t_2)$ 
23: Send  $(IL, SE_n(Q), i)$  to the CS, where  $i$  is the identifier
    of the  $DBU$ 

24:  $CS(IL, SE_n(Q), i)$ :
25:  $SR \leftarrow \emptyset$ 
26: for each  $id \in IL$  do
27:   if  $Match(EDB(id), SE_n(Q)) = \text{true}$  then
28:     Add all the required  $DE(d)$  in  $EDB(id)$  into  $SR$ 
29: for each  $DE(d) = (e_1 = g_1^r, e_2 = h^r d)$  in  $SR$  do
30:    $DE'(d) \leftarrow (e_1, e_2' = e_2 * e_1^{-x_2} = g_1^{r-x_2} d = g_1^{x_1 r} d)$ ,
    where  $x_2$  is the CS side key for  $DBU_i$ 
31: Send  $SR$  to the OPS

32:  $OPS(SR)$ :
33: Remove all dummy records from  $SR$  by checking flags
34: Send  $SR$  to the DBU

35:  $DBU_i(SR)$ :
36: for each  $DE'(d) = (e_1 = g_1^r, e_2' = g_1^{x_1 r} d) \in SR$  do
37:    $d \leftarrow e_2' * e_1^{-x_1} = g_1^{x_1 r} d * g_1^{-x_1 r}$ 
    
```

3.3 Query Execution

ObliviousDB supports SQL-like queries consisting of a set of equalities and inequalities, which are connected with conjunctions (*i.e.*, and) and disjunctions (*i.e.*, or), such as ‘select * from staff where name = Alice and age > 30’. To support range queries, we use the same approach presented in [11].

Every query executed in *ObliviousDB* is performed with the cooperation of the DBU, the OPS and the CS. The details of the steps performed by each entity are described in Algorithm 1.

The DBU encrypts the query with key K_{U_i} (Lines 1–11, Algorithm 1). For an insert or update query, each data element d is encrypted under GE, SE and DE. For a select query, a keyword k in the WHERE-clause is encrypted only under GE and SE. The encrypted query EQ and group information GQ are sent to the OPS. Both DE and SE are semantically secure because of the random numbers r , which prevents the CS to infer the search pattern from EQ , or learn any frequency information in EDB .

Algorithm 2 *Match*($rcd, SE_n(Q)$)

```

1: for each  $SE_n(k) = (t_1, t_2) \in SE_n(Q)$  do
2:   Get  $SE_n(d)$  in the same field from  $rcd$ 
3:   if  $e(c_1, t_2) \neq (c_2, t_1)$  and  $*$  = ‘and’ then
4:     return false

5:   if  $e(c_1 t_2) = e(c_2, t_1)$  and  $(* = \text{‘or’})$  or  $k$ 
    is the last keyword then
6:     return true
7: return false
    
```

On the OPS, the original query EQ is cached temporarily (Line 14), and the real insert, delete and update operations will be performed by the oblivious algorithm later. To hide the operation pattern, the OPS always sends an encrypted select query to the CS. So, the OPS first transforms EQ into a select query (Lines 15–17). Specifically, if EQ is update or delete query, the OPS just changes the terms ‘delete’ and ‘update’ into ‘select *’. If EQ is insert, a random number of values in $SE(Q)$ are used to assemble the WHERE-clause of the fake select query.

Second, to improve search efficiency, the OPS gets the search range IL for the CS, which is populated by merging the index lists of involved groups according to the conjunctions and disjunctions in GQ (Lines 19–21). Meanwhile, to ensure forward and backward privacy, each SE in EDB is bound to the nonce of this group (Line 22). Only the query bound with the same nonce could match the record. Both the index list IL and the transformed query $SE_n(Q)$ are sent to the CS.

Finally, the CS checks each record in IL with $SE_n(Q)$ by performing the pairing map operation (Lines 26–28). The match operation is described in detail in Algorithm 2. Assume the searched data element is $SE_n(d) = (c_1 = g_1^{r'n'}, c_2 = g_1^{\sigma'r'})$ and the encrypted keyword in $SE_n(Q)$ is $SE_n(k) = (t_1 = g_2^{rn}, t_2 = g_2^{\sigma r})$. The equality check between them is performed by checking whether $e(c_1, t_2) = e(c_2, t_1)$. Note that

$$\begin{aligned} e(c_1, t_2) = e(c_2, t_1) &\iff e(g_1^{r'n'}, g_2^{\sigma r}) = e(g_1^{\sigma'r'}, g_2^{rn}) \\ &\iff e(g_1, g_2)^{r'n'r\sigma} = e(g_1, g_2)^{rnr'\sigma'} \end{aligned}$$

and so if $\sigma = \sigma'$ and $n = n'$ then equality holds, while inequality holds with negligible probability if $\sigma \neq \sigma'$ and $n \neq n'$. That is, the record matches the query only when $k = d$ and they are bound to the same nonce. Finally, the search result SR is sent to the OPS. Yang *et al.* introduce a similar method to perform the equality check for SE schemes in [15]. However, their method leaks the search pattern and frequency information of records to the CS, since the pairing map they use is symmetric. That is, the CS could still infer if they are the same or not by running the bilinear map operation between two records or two queries, although they are encrypted with a probabilistic algorithm.

The search result is recovered in two rounds of decryption (Lines 29–37, Algorithm 1). Before sending SR to the OPS, the CS first pre-decrypts the DE parts of each matched record with DBU_i 's CS side key x_2 (Lines 29–30), and sends the pre-decrypted SR to the OPS. Next, the OPS filters the dummy records out (Lines 33–35). Finally, with the pre-decrypted real search result get from the OPS, the DBU can recover the plaintext with its DBU side key x_1 (Lines 36–37).

Algorithm 3 *Oblivious*(EQ, GQ, t)

```

1:  $Rcds \leftarrow \emptyset$ 
2: for each  $GE(k) \in GQ$  do
3:    $(n, il) \leftarrow GDB(GE(k))$ 
4:    $rcds \leftarrow EDB(il)$  {Get from CS all the records indexed by  $il$ }
5:    $n \leftarrow mn'$ , where  $n' \xleftarrow{\$} Z_p^*$ 
6:   for each record  $rcd \in rcds$  do
7:      $SE_n(d) = (c_1 \leftarrow c_1', c_2)$  { $d$  is in the same field as  $k$ }
8:     for each  $(SE_n(d), DE(d))$  pair in  $rcd$  do
9:        $r \xleftarrow{\$} Z_p, SE_n(d) = (c_1 \leftarrow c_1', c_2 \leftarrow c_2')$ 
10:       $r \xleftarrow{\$} Z_p, DE(d) = (e_1 \leftarrow e_1', e_2 \leftarrow e_2')$ 
11:      for each dummy record  $rcd \in rcds$  do
12:        if  $du/re > t$  then
13:          delete it,  $du --$ 
14:        else
15:           $SE(d') = (c_1, c_2) \xleftarrow{\$} rcds$ 
16:           $SE_n(d) \leftarrow (c_1^{nr}, c_2'), r \xleftarrow{\$} Z_p^*$  { $d$  is in the same field as  $k$ }
17:       $Rcds \leftarrow Rcds \cup rcds$ 
18:      for each matched real record  $rcd \in Rcds$  do
19:        if  $EQ$  is an update query then
20:          for each  $SE(d) \in SE(Q)$  do
21:             $SE_n(d) \leftarrow (c_1', c_2), n \leftarrow GDB(GE(d))$ 
22:            Update  $rcd$  with  $SE_n(Q)$  and  $DE(Q)$ 
23:          if  $EQ$  is a delete query then
24:            Invert its flag
25:           $du ++, re --$ 
26:          if  $EQ$  is an insert query then
27:            Assign an  $id$  to the new record
28:          for each  $SE(d) \in SE(Q)$  do
29:            if  $\emptyset \leftarrow GDB(GE(d))$  then
30:               $n \xleftarrow{\$} Z_p^*$ 
31:               $GDB(GE(d)) \leftarrow (n, id)$ 
32:            else
33:               $(n, il) \leftarrow GDB(GE(d))$ 
34:               $il \leftarrow il \cup id$ 
35:               $SE_n(d) \leftarrow (c_1 \leftarrow c_1', c_2)$ 
36:             $Rcds \leftarrow Rcds \cup (SE_n(Q), DE(Q))$ 
37:             $re ++$ 
38:             $flags[id] = 1$ 
39:          else
40:            Assign the  $id$  to a new dummy record
41:          for each field in  $rcd$  do
42:             $SE(d') = (c_1, c_2) \xleftarrow{\$} Rcds$ 
43:             $r \xleftarrow{\$} Z_p, n \leftarrow GDB(g), SE_n(d) \leftarrow (c_1^{nr}, c_2')$ 
44:             $e_1, e_2 \xleftarrow{\$} G_1, DE(d) \leftarrow (e_1, e_2)$ 
45:             $il' \leftarrow GDB(GE(d'))$ 
46:             $il' \leftarrow il' \cup id$ 
47:             $Rcds \leftarrow Rcds \cup rcd$ 
48:             $du ++$ 
49:             $flags[id] = 0$ 
50:          Shuffle  $Rcds$  and upload to CS
51:          Update the index list in  $GDB$  and update  $flags$ 

```

3.4 Oblivious Algorithm

To hide the access, size and operation patterns and ensure forward and backward privacy, in the oblivious algorithm, the OPS shuffles and re-randomises all the records included in the searched the groups every time a query is executed.

To ensure a high level of security, it is possible to shuffle all the records in the database. However, this degrades the system performance. The more records are shuffled, the more difficult it is for the CSPs to infer the access pattern, but it is worse in terms of the system performance. In this work, we shuffle all the records in the searched groups. In this case, the CS can only recognise if two queries are performed within the same group or not. In practice, the number of records to be shuffled can be set according to the performance and security requirements. Note that, at this stage, the user has already obtained the search results from the CS and does not need to wait for the shuffle operation to be completed.

There are four main steps in the oblivious algorithm. In the first step (Lines 3–10, Algorithm 3), for each group involved in the query, the OPS updates its nonce and re-encrypts the associated SE parts of the records in this group with the new nonce (Line 7). Consequently, the queries bound to previous nonces cannot match the re-encrypted records, as illustrated in (Sect. 3.3). Similarly, a new query bound to the latest nonce can not match stale records. That is, both forward and backward privacy are ensured. Meanwhile, the OPS re-randomises both the SE and DE parts of all the records to be shuffled to make them untraceable (Lines 8–10).

In the second step (Lines 11–16), all the dummy records in each searched group are updated. The OPS controls the number of dummy records to ensure the performance of the system (Lines 12–13). To this end, the OPS counts the total number of real records re in the index list IL and the total number of dummy records du in IL . When the ratio of dummy records exceeds the threshold t , some of them are deleted. The OPS updates the SE parts of the remained dummy records (Lines 14–16). Although the dummy record protects the real size pattern, if the fake size pattern for the same query never changes, the CS could make some assumptions on whether two queries are equivalent or not by checking the number of the records in the result set. To protect the search pattern, it is necessary to make the size pattern for all queries variable. Note that considering the correctness of the system it is impossible to change the number of matched real records when there is no insert, delete or update operation. In our work, the OPS replaces the SE parts of dummy records with randomly chosen ones from non-dummy records in the group (lines 15–16).

Recall that to protect the operation pattern the OPS only sends select queries to the CS. The real insert, update and delete operations are executed in the third step (Lines 18–48). If the type of the original query is update (Lines 19–22), the OPS re-encrypts the cached SE parts with the latest nonce stored in GDB , and replaces both the SE and DE parts of the matched records with new values. If the type of the original query is delete (Lines 23–25), in order to protect the operation pattern, the OPS converts them into dummy records by inverting the flag into dummy instead of deleting them directly. In this way, no matter what type the original query is, the number of records returned by the oblivious algorithm is only affected by de , re and t . On the contrary, if we delete them, fewer records will be sent to the CS. For other types of queries, a number of dummy records are probably deleted. However, the CS could learn the type of the original query must not be delete when it gets more records or the same number of records after the oblivious algorithm. If the type of the original query is insert (Lines 26–38), the OPS re-encrypts the cached SE parts with the latest nonces and adds it to the records set. In case if the original query is not insert, the OPS generates a dummy record (Lines 39–49) by re-randomising the SE parts that picked from $Rcds$ randomly. Otherwise, the CS could infer the type of the original query must be insert when receiving one more record after the oblivious algorithm.

Finally, the OPS shuffles all the updated records set and sends them to the CS (Line 50). Note that their flags and ids stored in the index list are updated at the same time. Because of the shuffling and re-randomising, if the same query is executed again, the search result will be totally different from the previous ones in terms of the store locations on the CS, size and appearance, which means the CS is unable to infer if different search results contain the same records or not. That is, the access pattern is protected.

4 Security Analysis

In this section, we formally define and prove SPP, APP, SzPP, OPP, forward and backward.

Leakage. *ObliviousDB* aims at minimising information leakage. At the same time, we require *ObliviousDB* to be efficient. To meet these two conflicting requirements, we consider a trade-off between security and performance. In this work, we achieve SPP and APP for those queries involved in the same groups. To optimise the performance of the system, we divide the data into groups and only perform the search and oblivious operations within groups. Consequently, the CS could learn if some records and interested keywords are in the same groups or not. Second, considering we do not encrypt the conjunctions between predicates, the CS could learn the number of predicates and their conjunctions in complex queries. In addition, the CS could learn if two queries are searched over the same fields or not, since we do not re-randomise the field names and shuffle the columns. Given these leakages, we formalise our security definition below.

Definition. In our security definition, we only consider the queries with the same structure. Any two queries Q_0 and Q_1 have the ‘same structure’ if they satisfy the following criteria:

- Both SQL queries have the same logical structure (all WHERE-clauses are the same equalities and inequalities with respect to the same data fields, and the sentences are formed from the clauses using the same conjunctions or disjunctions in the same order). This can be achieved by padding and reordering the WHERE clause. Note that the queries could be insert, select, update or delete.
- The groups involved by each equality/inequality in Q_0 should be same to the one involved in the corresponding equality/inequality in Q_1 . This will ensure that both index lists are the same, *i.e.*, $IL_0 = IL_1$.
- The number of matched real records $|RR|$ for each equality/inequality in Q_0 and Q_1 should be similar in size, namely $||RR_0| - |RR_1|| \leq \theta * \min\{|RR_0|, |RR_1|\}$, where θ is a parameter specified when the scheme is set up.

The CS is modelled as the Probabilistic Polynomial-Time (PPT) adversary \mathcal{A} , which means \mathcal{A} honestly follows the protocols and gets all the messages the CS sees.

The scheme is considered to be secure if an adversary could break it with not more than a negligible probability. Formally, it could be defined as follows:

Definition 1 (Negligible Function). A function f is negligible if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

Definition 2. Let $\Pi = (\text{Setup}, \text{Query}, \text{Oblivious})$ be ObliviousDB, λ be the security parameter, and t be the threshold indicating the ratio between dummy and real records. \mathcal{A} is a PPT adversary, and \mathcal{C} is a challenger. The game between \mathcal{A} and \mathcal{C} in Π is described as below:

- **Setup.** The challenger \mathcal{C} first initialises the system by generating Params and MSK. Then, she generates the secret key pair (K_U, K_S) . The adversary \mathcal{A} is given Params and K_S .
- **Bootstrap.** \mathcal{A} submits a database Δ ¹. Assume Δ contains n records with a certain number of fields. \mathcal{C} encrypts Δ and divides the data in each field into groups. Moreover, \mathcal{C} generates a number of dummy records for each group, such that the total number of dummy records is $t \cdot n$. The encrypted database EDB is sent to \mathcal{A} . The encrypted groups information GDB is securely kept by \mathcal{C} .
- **Phase 1.** \mathcal{A} can make polynomially many SQL queries Q in plaintext. All the queries are in the same structure but could be of different types. \mathcal{C} encrypts and transforms each query Q to $SE_n(Q)$, and generates the index list IL , as would be done by the DBU and the OPS. With $SE_n(Q)$ and IL , \mathcal{A} searches over EDB to get the search result SR . After that, \mathcal{C} and \mathcal{A} engage in the oblivious algorithm to update EDB. So, for each query, \mathcal{A} sees $SE_n(Q)$, IL , SR and the records set $Rc ds$ returned by the oblivious algorithm. Note that, the \mathcal{A} could cache $SE_n(Q)$ and execute it again independently at any time.
- **Challenge.** \mathcal{A} sends two queries Q_0 and Q_1 to \mathcal{C} that have the same structure, which can be those already issued in phase 1. Note that if Q_0 or Q_1 is insert or update query, the data elements included in them should already exist in Δ . \mathcal{C} responds the request as follows: it chooses a random bit $b \in \{0, 1\}$ and transforms query Q_b , as done by the DBU and OPS, to $SE_n(Q)$ and IL . Then, \mathcal{C} and \mathcal{A} perform the full protocol, so that \mathcal{A} learns SR and $Rc ds$.
- **Phase 2.** \mathcal{A} continues to adaptively request polynomially many queries, which could include the challenged queries Q_0 and Q_1 .
- **Guess.** \mathcal{A} submits her guess b' .

The advantage of \mathcal{A} in this game is defined as:

$$Adv_{\mathcal{A}, \Pi}(1^\lambda) = Pr[b' = b] - \frac{1}{2}.$$

We say ObliviousDB achieves SPP, APP, SzPP, OPP, forward and backward privacy, if all PPT adversaries have negligible advantage in the above game.

In this game, \mathcal{A} is very powerful. She knows the plaintext of all the real records and queries, could arbitrarily generate and issue any kind of queries as long as they are in the same structure, and has the full access to EDB. That is, she could learn the real search result of all the issued queries, and

¹ For simplicity, we assume there is only a single table in Δ and regard Δ as a table. Without loss of generality, our proofs will hold for a database containing a set of tables.

could adaptively run the encrypted queries over EDB to get the encrypted search results at any time. If one of the search, access, size, operation patterns and forward and backward privacy is not protected, \mathcal{A} could infer b easily. For example, if the search pattern is not protected (*i.e.*, \mathcal{A} can learn if the terms involved in two queries are the same or not), she could select one of the queries issued in phrase 1 as either Q_0 or Q_1 and win the game by checking if $SE_n(Q)$ is same to one of those get in phase 1; if the size pattern is not protected (*i.e.*, \mathcal{A} can learn the number of real records in SR), she could win the game by setting Q_0 and Q_1 with different numbers of matched records; if the operation pattern is not protected (*i.e.*, \mathcal{A} can learn the type of Q_b), she could set two different types of query as Q_0 and Q_1 and win the game from the type of $SE_n(Q)$. In other words, if with these abilities, \mathcal{A} still can not win the game with non-negligible advantage, it means *ObliviousDB* achieves all the properties.

Theorem 1. *Let the SE and DE schemes have semantic security. Let t (the proportion of dummy records) be chosen sufficiently large relative to θ . If the SE and DE schemes have semantic security, ObliviousDB achieves SPP, APP, OPP, SzPP, forward and backward privacy.*

Proof (Sketch). We show that the bit b chosen by \mathcal{C} is information-theoretically hidden from the view of \mathcal{A} , assuming that both SE and DE are semantically secure.

Consider the view of \mathcal{A} in the game. \mathcal{A} chooses an arbitrary database Δ and uploads this to \mathcal{C} . In Phase 1, \mathcal{A} makes queries that are answered correctly by \mathcal{C} by following the protocols.

In the challenge round, \mathcal{A} sends two queries Q_0 and Q_1 . \mathcal{A} receives a list of $SE_n(k)$ terms corresponding to the literals in the predicate defining the query. By definition, the two queries have the same structure. Hence, the same number of literals, each of the same type, will be received by \mathcal{A} for either query. Since *SE* is semantically secure, \mathcal{A} cannot distinguish the query terms given the ciphertexts $SE_n(k)$.

\mathcal{A} also receives a list IL of database indexes to be searched by the CS, and by definition, this is the same list for all queries. Hence, no information about the queries can be leaked by IL .

Each group involved in IL is accompanied by a nonce n . With overwhelming probability, these nonces are distinct and unrelated to the values used in previous queries. Previously encrypted search keywords can no longer be used to query these indexes, and $SE_n(Q)$ can not be executed over stale records, since the nonces do not match. Hence, there is no way to link information from previous search queries to these records, indicating forward and backward privacy is achieved.

The adversary \mathcal{A} may try to guess b from the search result SR . Although the numbers of real records matched with Q_0 and Q_1 are known to \mathcal{A} since all the queries and real records in plaintext are set by her, a number of dummy records are inserted into EDB in order to hide the number of real records that are matched. Since SE and DE are semantically secure, the dummy records

are indistinguishable from the real ones, if t is sufficiently large compared to θ then the probability distributions of the result set sizes $|SR_0|$ and $|SR_1|$ are statistically close and \mathcal{A} cannot distinguish them from a single query. Therefore, the CS is unable to distinguish the two queries from the size of SR , indicating SzPP is achieved.

Even if the queries Q_0 and/or Q_1 have previously been executed by \mathcal{A} , the refreshing of dummy records, together with the shuffling and re-randomising performed in the oblivious algorithm, imply that \mathcal{A} cannot distinguish the two queries by comparing SR with previous search results, indicating APP is achieved.

Finally, \mathcal{A} gets $Rclds$ from the oblivious algorithm. Some records in $Rclds$ may be updated with new values or turned into dummy records, and one of them is newly added. Due to the semantic security of SE and DE , $Rclds$ leaks nothing to \mathcal{A} , indicating the operation pattern is concealed, *i.e.*, OPP is achieved.

The game continues in Phase 2. \mathcal{A} may repeat Q_0 and/or Q_1 . If Q_0 and Q_1 are different types of queries, *e.g.*, insert and delete. \mathcal{A} may run a related select query to test the search result. Again, due to the refreshing of dummy records, the shuffling and re-randomising operations, the number, the ciphertext and store locations of matched records for Q_0 and/or Q_1 will be different from SR . Similarly, the nonce updating does not allow records to be linked to records found in previous search queries. Hence, the future state of the database and the queries in Phase 2 are independent of the query made in the challenge round.

Since \mathcal{A} has no information to distinguish the bit b , the scheme satisfies the definition. ■

5 Performance Analysis

We implemented the scheme in C using the MIRACL 7.0.0 library, necessary for cryptographic primitives. The implementation of the overall system including the functions on the DBU, the OPS and the CS was tested on a single machine with 64 Intel i5 3.3 GHz processor and 8 GB RAM running Ubuntu 14.08 Linux system. In our testing scenario, we ignored network latency that could occur in a real deployment. In the following, all the results are averaged over 10 trials.

The tested database contains one table with 3 fields. Considering the search operation can be performed in each field, we encrypted each field with SE separately. However, each record was encrypted by DE as a whole since we only tested ‘select *’ queries. In the following, we tested how the three controllable parameters namely the number of dummy records, the number of groups and the number of shuffled records, affect the performance of the system.

We first present the results of end-to-end latency measured at the DBU when performing a search operation on a database consisting of 100,000 real records with a result set of 1,000 real records. Note that, in the DBU latency experiment, we did not measure the time the OPS spends in executing the oblivious algorithm. The reason is that the OPS will forward to the DBU the result sets before initiating the oblivious algorithm.

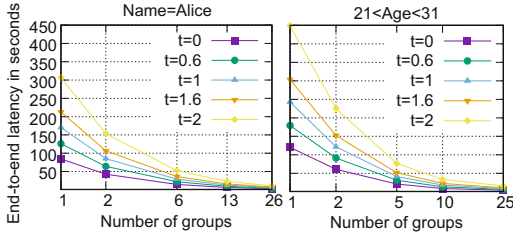


Fig. 2. End-to-end latency on the DBU for getting 1,000 real records from the database consisting of 100,000 real records. The database size goes up to 300,000 with the increase of t , the ratio between dummy and real records.

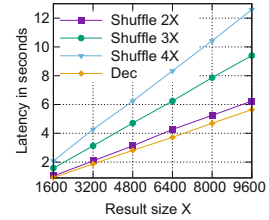


Fig. 3. Oblivious latency with $t = 0.6$. *Shuffle 2X*, *3X*, *4X* mean the number of records to be shuffled are 2, 3, 4 times of the result size, respectively.

The graphs in Fig. 2 illustrate latency in seconds. In particular, Fig. 2(a) shows the results for a simple select query. Figure 2(b) reports latency for performing a range query on a numerical field. In both graphs, the X-axis shows the number of groups: that is, we change the granularity of the indexing going from no indexing (where all the records are part of one group) to a more fine-grained indexing. For a given number of groups, the same experiment was executed 5 times, each time changing the ratio t , represented by different lines in both graphs.

As we expected, for both queries, increasing the number of groups reduces the DBU latency. For a given size of a database, more groups mean fewer records within a given group. This reduces searching time on the CS and in turn reduces latency on the DBU.

On the other hand, increasing the value of t degrades the performance. For both queries, for a given group size, there is a slight latency increase when we go from $t = 0$ dummy records (*i.e.*, only real records) to a ratio of $t = 2$ (*i.e.*, 2 times dummy records of real ones). This is explained mainly by two facts: (a) with more dummy records, the CS has to retrieve more records (both real and dummy ones), and (b) the OPS needs to filter out the dummy records before sending the real records to the DBU: the higher the percentage of dummy records, the longer it takes for the OPS to remove them from the result set (to be returned to the DBU). Recall that $t = 0.6$ is the minimum value to ensure security reported by Cash *et al.*

Second, we measured the effect on the oblivious algorithm when varying the number of records to be shuffled. As shown in Fig. 3, the latency of the oblivious algorithm goes up linearly with the increase in result size, which affects the size of shuffled records in the test setting. Fortunately, the oblivious algorithm can be executed in parallel with decryption operations since they are independently executed by different entities. From Fig. 3, we can observe that, if we shuffle the search result with the unmatched records (where the number of unmatched records is same as the number of records in the search result), the latency of the oblivious algorithm is close to the decryption time, indicating the oblivious algorithm does not severely degrade the throughput of *ObliviousDB*.

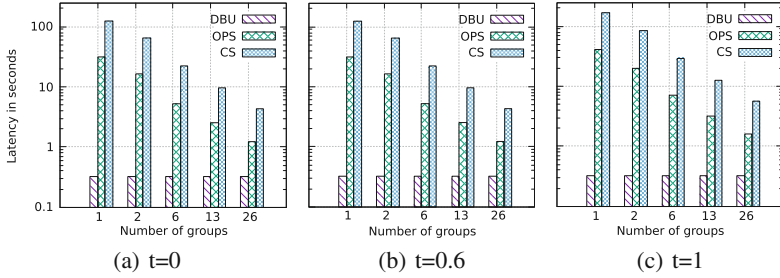


Fig. 4. Latency on the DBU, the OPS and the CS for executing ‘select * from *staff* where name = Alice’ with three different ratios of dummy records.

Next, we want to provide some details on the time each entity, namely the DBU, the OPS and the CS, spends for executing a query. Figure 4 shows the graphs for the execution of the select query (same as the one for the graph in Fig. 2(a)). In this experiment, for each graph, we shuffled all the records in the searched group and kept the ratio constant while we changed only the number of groups (shown on the X-axis). As we can see, the more groups we introduce, the better performance we achieve on the CS (while for the DBU and the OPS, there is no big variation). Increasing the ratio between dummy and real records slightly increases latency on the OPS and the CS.

From our experiments, we can see that latency, although substantially higher than a less secure scheme like CryptDB, for DBUs using *ObliviousDB* is still usable especially when introducing more groups. Also, by comparing Figs. 2(a) and (b), we can see that the performance of numerical range queries is not much different from simpler single keyword queries.

At the same time, to ensure data confidentiality, it is necessary to maintain some dummy records in the data set. However, our experiments show that the burden of maintaining dummy records does not impact latency on the DBU, in particular when a large number of groups are used. The cost of maintaining the dummy records is offloaded to the OPS and the CS, which are likely to be deployed on more powerful machines than the one used by the DBU.

6 Conclusions and Future Work

In this work, we propose *ObliviousDB*, a searchable scheme for hybrid outsourced databases. *ObliviousDB* is the first full-fledged multi-user scheme that does not leak information about search pattern, access pattern, size pattern and operation pattern. It is also the first scheme that achieves both forward and backward privacy, where the CS cannot reuse cached queries for checking if new records have been inserted or if records have been deleted. We have implemented *ObliviousDB* and shown that it is capable of performing numerical range queries with 1000 results on a database of 200,000 records in around 4 s.

As future work, we plan to carry out a thorough security analysis for identifying a right balance between real and dummy records for achieving a sustainable

level of security without degrading performance. Another area we want to explore is to investigate sub-linear data structure to achieve more efficiency.

References

1. Gartner expects five years for hybrid cloud to reach productivity. <http://www.cloudcomputing-news.net/news/2015/aug/18/gartner-expects-hybrid-cloud-reach-productivity-five-years-are-they-right/>. Accessed 19 Feb 2016
2. Rightscale 2016 state of the cloud report. <https://www.rightscale.com/lp/state-of-the-cloud>. Accessed 3 July 2016
3. Cui, S., Asghar, M.R., Galbraith, S.D., Russello, G.: Secure and practical searchable encryption: a position paper. In: Pieprzyk, J., Suriadi, S. (eds.) ACISP 2017. LNCS, vol. 10342, pp. 266–281. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60055-0_14
4. Cash, D., Grubbs, P., Perry, J., Ristenpart, T.: Leakage-abuse attacks against searchable encryption. In: Ray, I., Li, N., Kruegel, C. (eds.) SIGSAC 2015, pp. 668–679. ACM, New York (2015)
5. Naveed, M., Kamara, S., Wright, C.V.: Inference attacks on property-preserving encrypted databases. In: Ray, I., Li, N., Kruegel, C. (eds.) SIGSAC 2015, pp. 644–655. ACM, New York (2015)
6. Zhang, Y., Katz, J., Papamanthou, C.: All your queries are belong to us: the power of file-injection attacks on searchable encryption. In: USENIX Security 2016, pp. 707–720. USENIX Association (2016)
7. Popa, R.A., Redfield, C.M.S., Zeldovich, N., Balakrishnan, H.: CryptDB: protecting confidentiality with encrypted query processing. In: Wobber, T., Druschel, P. (eds.) SOSP 2011, pp. 85–100. ACM, New York (2011)
8. Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: NDSS 2013, vol. 71, pp. 72–75 (2014)
9. Bost, R.: $\sum\phi\phi\phi$: forward secure searchable encryption. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) SIGSAC 2016, pp. 1143–1154. ACM, New York (2016)
10. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
11. Asghar, M.R., Russello, G., Crispo, B., Ion, M.: Supporting complex queries and access policies for multi-user encrypted databases. In: Juels, A., Parno, B. (eds.) CCSW 2013, pp. 77–88. ACM, New York (2013)
12. Asghar, M.R.: Privacy preserving enforcement of sensitive policies in outsourced and distributed environments. Ph.D. dissertation, University of Trento, Trento, Italy, December 2013. <http://eprints-phd.biblio.unitn.it/1124/>
13. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. *Discret. Appl. Math.* **156**(16), 3113–3121 (2008)
14. Dong, C., Russello, G., Dulay, N.: Shared and searchable encrypted data for untrusted servers. In: Atluri, V. (ed.) DBSec 2008. LNCS, vol. 5094, pp. 127–143. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70567-3_10
15. Yang, G., Tan, C.H., Huang, Q., Wong, D.S.: Probabilistic public key encryption with equality test. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 119–131. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11925-5_9



Ethereum: State of Knowledge and Research Perspectives

Sergei Tikhomirov(✉)

SnT, University of Luxembourg, Esch-sur-Alzette, Luxembourg
sergey.s.tikhomirov@gmail.com

Abstract. Ethereum is a major blockchain-based platform for smart contracts – Turing complete programs that are executed in a decentralized network and usually manipulate digital units of value. A peer-to-peer network of mutually distrusting nodes maintains a common view of the global state and executes code upon request. The stated is stored in a blockchain secured by a proof-of-work consensus mechanism similar to that in Bitcoin. The core value proposition of Ethereum is a full-featured programming language suitable for implementing complex business logic.

Decentralized applications without a trusted third party are appealing in areas like crowdfunding, financial services, identity management, and gambling. Smart contracts are a challenging research topic that spans over areas ranging from cryptography, consensus algorithms, and programming languages to governance, finance, and law.

This paper summarizes the state of knowledge in this field. We provide a technical overview of Ethereum, outline open challenges, and review proposed solutions. We also mention alternative smart contract blockchains.

Keywords: Blockchain · Ethereum · Smart contracts
State of knowledge

1 Introduction

Bitcoin [Nak08] is the first fully decentralized digital currency introduced in 2008 and launched in 2009. It innovatively combines cryptographic techniques with economic incentives to make rational participants likely to play by the rules. Bitcoin gained significant traction, reaching \$80 billion market capitalization in September 2017. Hundreds of alternative cryptocurrencies based on similar general design have appeared since Bitcoin's launch. Programming languages in early blockchains, e.g., the Bitcoin scripting language, were deliberately limited to reduce complexity for the sake of security.

Ethereum [VB+14, Woo14], announced in 2014 and launched in 2015, aims at creating a universal blockchain-based application platform. It incorporates a Turing complete language, making it theoretically possible to express all practical computations in *smart contracts* – pieces of code permanently stored on the

blockchain and capable of responding to users' requests. This enhanced functionality introduces new security challenges related to language design and secure programming practices.

Ethereum is not the only smart contract blockchain system [BP17]. Ethereum Classic [Eth17c] is an alternative blockchain originating from a controversial Ethereum update. Rootstock [Roo17] and Qtum [Qtu17] aim at implementing smart contracts in combination with the Bitcoin blockchain. Chain [Cha17a], Corda [Cor17], and Hyperledger [Hyp17] propose permissioned (i.e., with a fixed set of approved participants) smart contract blockchains, designed to simplify transactions between corporate entities.

This paper focuses on Ethereum as the most mature open blockchain with Turing complete programming capabilities. We summarize the state of knowledge and outline the research perspectives in this rapidly developing field. We assume familiarity with the basic blockchain concepts; [BMC+15, TS15] provide the necessary background.

2 Technical Overview

2.1 State and Accounts

Ethereum can be thought of as a state machine. Nodes of the Ethereum peer-to-peer network maintain a shared view of the global state. A user interacts with the network by issuing a transaction representing a valid state transition. Nodes pick transactions from the *mempool* (the set of unconfirmed transactions), verify their validity, perform the corresponding computation (possibly changing ownership of units of the Ethereum native cryptocurrency *ether*), and update the state. There are two types of accounts in Ethereum: *externally owned accounts* and *contract accounts* controlled by a private key or by a smart contract – a piece of code deployed on the blockchain – respectively.

The account state consists of the following fields:

- *nonce* – the number of transactions sent by this account (for externally controlled accounts) or the number of contract creations made by this account (for contract accounts);
- *balance* – the number of *wei*¹ owned by this account;
- *storageRoot* – Merkle Patricia tree root of this account's storage;
- *codeHash* – hash of this account's contract bytecode.

Accounts' 160-bit addresses² are derived from its public key or, in case of contract accounts, from the address of the contract's creator and its nonce [eth16]. The global state maps addresses to account states. The primary data structure in Ethereum is the Merkle Patricia tree – a radix tree optimized for key-value mappings with 256 bit keys [VBR+17, Buc14]. The root hash authenticates the whole data structure. Values pairs are editable in logarithmic time.

¹ Smallest denomination of ether: 1 ether = 10^{18} wei.

² Addresses are usually written in hex with a 0x prefix.

The Ethereum state model (accounts and states) differs from than in Bitcoin. The Bitcoin blockchain stores unspent transaction output (UTXO); balances of addresses are calculated off-chain by wallet software.

2.2 Transactions and Gas

The halting problem – determining if a given program will ever halt – is unsolvable in the general case [Chu36]. This poses a challenge: nodes running the Ethereum virtual machine (EVM) cannot foresee the amount of resources required for validating a transaction, which enables denial-of-service attacks.

To overcome the issue, the Ethereum protocol incorporates a pricing mechanism. It makes resource-intensive computations in smart contracts economically infeasible. Every computational step in EVM is priced in units of *gas*. EVM opcodes and their gas costs are defined in the Yellow paper [Woo14]. The price of a gas unit in ether is determined by the market. For every transaction, the sender specifies the maximum amount of gas that the intended computation is expected to consume (the *gas limit*) and the price the user wishes to pay per unit of gas (the *gas price*). The transaction fee equals the gas limit multiplied by the gas price. If the execution is successful, the remaining ether is refunded. If an error occurs, the transaction has no effect on the state, but all provided gas is consumed. Miners can vote to gradually change the limit on the total amount of gas consumed in a block [jmm15].

A transaction is a signed data structure comprising a set of instructions to be atomically executed by the EVM. It consists of the following fields:

- *nonce* – the number of transactions sent by the sender;
- *gasPrice* – the number of wei per gas unit that the sender is paying;
- *gasLimit* – the maximum amount of gas to be spent during execution;
- *to* – the destination address (0x0 for contract creation transactions);
- *value* – the number of wei transferred along with the transaction;
- *v, r, s* – signature data.

There are two types of transactions in Ethereum. A *contract creation transaction* is used to deploy a new contract. It contains an additional *init* field that specifies the EVM code to be run on contract creation, as well as the EVM code of the new contract. A *message call transaction* is used to execute a function of an existing contract (with arguments specified by the an optional *data* field) or to transfer ether.

2.3 Block Structure and Mining

Ethereum uses proof-of-work (PoW): nodes compete to find a partial collision of a cryptographic hash function and produce the next block³. Both Bitcoin [Wui17] and Ethereum [Joh17] chose the *heaviest* chain as a valid one in case of forks, where a chain's *weight* is defined as the sum of its blocks' difficulties.

³ See [ato16] for a visual interpretation of the block structure in Ethereum.

Good connectivity is crucial for Bitcoin mining operation: the resources spent mining on a block other than the latest one are essentially wasted. Good connectivity puts big pools at an advantage, while blocks from worse connected miners propagate slowly and increase the orphan rate. Thus Bitcoin mining is prone to centralization. To be able to operate with block times much shorter than Bitcoin’s 10 min (about 30 s in September 2017), Ethereum uses a mining protocol [doc17] similar to GHOST [SZ13]. Ethereum considers *uncles* – valid orphan blocks that are ancestors of the current block (no more than 6 generations deep). For each block, the miner receives a static reward of 5 ether, payments for the gas consumed by transactions in the block, and 1/32 of the static reward (0.15625 ether) per uncle, whose hash is included in the block header (no more than 2 uncles per block). Miners of uncles whose headers get included in the main chain receive 7/8 of the static reward (4.375 ether). Due to uncles, the energy spent on orphan blocks contributes to security, increasing the amount of work required for a double-spend.

Contrary to Bitcoin, where coins are issued on a diminishing rate with a total cap of 21 million, Ethereum issues ethers at a constant rate with no total cap. Ethereum’s issuance parameters may change after switching to proof-of-stake (see Sect. 3.1).

Bitcoin PoW uses a general purpose cryptographic hash function SHA-256, which can be efficiently implemented in hardware. Specialized mining equipment (application-specific integrated circuits, ASIC) is orders of magnitude more efficient than commodity hardware, which puts small miners at a disadvantage. Ethereum uses a memory hard hash function Ethash and targets GPUs as the primary mining equipment. It helps prevent mining centralization akin to Bitcoin’s and throttles CPU mining (botnets or cloud VM instances can be rented for a short time to perform an attack).

Table 1 compares some properties of Bitcoin and Ethereum. Note that the practical requirements regarding the disk space for an Ethereum node can be greatly reduced due to the explicit storage of account balances and data as opposed to Bitcoin’s UTXO [Dom17].

Table 1. Bitcoin and Ethereum, September 2017 [Eth17d, Bit17c, Eth17e, Bit17b, Coi17a]

Metric	Bitcoin	Ethereum
Number of nodes	9428	22007
Blockchain size	158 GB	52 GB
Transactions per hour	8509	12406
Market capitalization (\$ million)	62812	27200
Daily trading volume (\$ million)	997	420

2.4 Smart Contract Programming

EVM bytecode is a low-level Turing complete stack-based language operating on 256-bit words designed to be simple compared to general purpose VMs

like JVM, execute deterministically, and natively support cryptographic primitives [But17b]. Developers usually write contracts in high-level languages targeting EVM, the most popular one being Solidity [Sol17] – a statically typed language with a Javascript-like syntax. Others include Serpent [Ser17] (deprecated in 2017 [Cas17]) and LLL [Ell17] (Python- and Lisp-like syntax respectively).

```

1  pragma solidity 0.4.17;
2  contract StringStorageContract {
3      string private str = "Hello, world!";
4      function getString() public constant returns (string) {
5          return str;
6      }
7      function setString(string _str) public {
8          str = _str;
9      }
10 }
```

Listing 1.1. A simple contract in Solidity

2.5 Applications

Among many potential applications of smart contracts [McA17], crowdfunding is arguably the first widely successful one. The first wide-scale Ethereum-based crowdfunding project was a decentralized investment fund called The DAO, launched on 30 April 2016⁴. In 2017, the amount of money collected during so-called initial coin offerings (ICO) skyrocketed, reaching \$1.8 bn [Coi17b] and surpassing early stage venture capital funding [Sun17]. ICO is usually based around a token – a smart contract that maintains a list of users’ balances and allows them to transfer tokens or buy and sell them for ether. Tokens are usually implemented with respect to the API defined in the ERC20 standard [Vog17]. The ICO organizers often promise that the tokens will be required to use the to-be developed product or service. Prominent Ethereum applications include decentralized file storage [Fil17, Sia17, Sto17] and computation [Gol17, Son17], name systems [ENS17], and prediction markets [Aug17, Gno17].

3 Open Problems

3.1 Core Protocol

Cryptographic Primitives. Ethereum uses ECDSA for signatures⁵, Keccak256 for generating unique identifiers⁶, and Ethash [Eth17a] for proof-of-work.

⁴ In June 2016, an unknown hacker exploited a vulnerability in the DAO code and withdrew around \$50 million, leading to a controversial [ETC16] hard fork.

⁵ See [May16] for a study of ECDSA security in Bitcoin and Ethereum.

⁶ Though Keccak256 is the winning proposal in the SHA3 competition, it differs from the officially standardized SHA3. SHA3 in the Ethereum documentation and source code refers to Keccak256.

Based on Dagger [But13] and Hashimoto [Dry14], Ethash is a memory intensive, GPU-friendly and ASIC-resistant hash function⁷.

The algorithm is composed of four steps. In the first step, a seed is created from the blockchain by hashing the headers of each block together with the current epoch using Keccak. An epoch consists of 30 thousand blocks. In the second step, a 16 MB pseudorandom cache is generated from the seed using a memory-hard hash function. In the third step, done once per epoch, a linearly growing dataset (approximately 2 GB in 2017 [DAG17]) consisting of 64 byte elements is generated from the cache using a non-cryptographic hash function Fowler-Noll-Vo [Nol17]. In the fourth step, the dataset, a header, and a nonce are repeatedly hashed until the result satisfies the difficulty target.

Both Dagger and Hashimoto, in contrast to standardization attempts like the SHA-3 competition [SHA17] or the Password hashing competition [PHC15], were announced shortly before the Ethereum launch and did not undergo significant cryptanalysis in the academic community. The Ethash design rationale [Eth17b] lacks details on why established and well-tested memory-hard hash functions do not serve the purpose. [Ler14] claims that an earlier version of Dagger (as of 2014) was flawed. Rigorous cryptanalysis of Ethereum’s underlying cryptographic primitives is required to guarantee its long-term security.

Consensus Mechanism. Though some argue that PoW is the only viable blockchain consensus mechanism [And14, Szt15], Ethereum is planning to switch from proof-of-work to proof-of-stake (PoS) [Her17]. As of September 2017, the first step of a two-stage process is due October 2017, transitioning Ethereum to a hybrid PoW-PoS consensus mechanism. The second step will make Ethereum fully PoS. PoS aims to address the drawbacks of PoW:

- energy consumption comparable to a mid-sized country as of 2017 [Dig17];
- centralization risks: miners are incentivized to invest in specialized hardware, which pushes up the entry cost of participating and puts big miners at an advantage due to economies of scale;
- game-theoretic attacks like selfish mining [ES13].

PoS can be described as “virtual mining”: a miner purchases coins instead of hardware and electricity. The consensus mechanism distributes power proportionally to the amount of coins miners hold (*stake*), not computing power (see [BGM16] for a review of cryptocurrencies without PoW). Known issues with naive PoS implementations include:

- *Nothing-at-stake*. As producing new blocks incurs only a negligible cost, a rational PoS validator extends all known chains to get a reward regardless of which one wins. This opens the door to attacks that require far less than 51%

⁷ Ethash is also referred to as Dagger-Hashimoto. Official documentation [Eth17a] states that Ethash “is the latest version of Dagger-Hashimoto, although it can no longer appropriately be called that since many of the original features of both algorithms have been drastically changed”.

of the stake⁸: the attacker’s chain wins if the attacker supports it exclusively, whereas other validators behave rationally and support all chains.

- *Randomly choosing validators.* Using randomness from the blockchain itself (i.e., previous block hash) to determine the next validator is insecure, as it is determined by validators in previous rounds. A possible solution is to use verifiable secret sharing for randomness generation.
- *Transaction finality.* In PoW, a block header which has a hash less than the target simultaneously represents the choice of the next validator and the very act of validating the block. PoS separates choosing the next validators and producing the block. A PoS validator may create its own chain, plug in a constant instead of a pseudo-random number generator (PRNG) output, and produce blocks despite owning an arbitrarily small stake.

A rule of thumb in Bitcoin considers transactions older than six blocks final, as the chance of a minority attacker overtaking the main chain becomes negligible. By contrast, as PoS blocks cost nearly nothing to produce, an attacker can secretly create an alternative chain starting from the genesis block. To prevent this, a PoS blockchain must provide finality – i.e., guarantee that after a fixed number of blocks old transaction can not be reversed⁹.

The central concept of the proposed Ethereum PoS algorithm Casper [But16a] is “consensus by bet”: validators bet on the future blockchain state [PoS16, But17c]. Casper addresses the nothing-at-stake problem by introducing validator punishments for incorrect behavior, e.g., extending multiple chains, in addition to rewards, which makes the game-theoretic analysis of the protocol more complex. Long range attacks are addressed with the concept of *finality* [But17a].

Recent PoS designs also include 2-hop blockchain [DFZ16], Algorand [Mic16], Ouroboros [KRDO16], SnowWhite [DPS16], Proof of luck [MHWK17]. Blockchain networks Ripple [SYB14] and Stellar [Maz14] use consensus mechanisms inspired by Byzantine fault tolerant consensus protocols like PBFT [CL02]. Developing an efficient, secure and incentive compatible PoS algorithm is an important task in blockchain research.

Scalability. Open blockchains deliberately sacrifice performance for what a smart contracts pioneer Nick Szabo describes as *social scalability* [Sza17] – “the ability of an institution [...] to overcome shortcomings in human minds [...] that limit who or how many can successfully participate”. Both Bitcoin and Ethereum have been facing scalability problems [Sil16, Bit17a]. Improving blockchain scalability while minimally sacrificing security is an important research direction. Blockchain scalability can be defined as two goals: increasing

⁸ A commonly used term “51% attack” is not precisely correct: “51%” here means “strictly greater than 50%”.

⁹ Interestingly, the reference Bitcoin implementation uses checkpoints to skip validation of very old blocks for efficiency, effectively providing finality for transactions older than the latest checkpoint [Bit16].

transaction throughput and decreasing the requirements on bandwidth, storage, and processing power for nodes (thus preserving decentralization).

The first goal can be addressed by payment channel networks and sharding. A bidirectional payment channel is a protocol that lets users exchange signed transactions before publishing of them on-chain as settlement. A network of payment channels is a protocol that finds a sequence of payment channels across the network, a mechanism similar to the IP packet routing [McC15]. Payment channel networks for Bitcoin [Lig16] and Ethereum [Rai17] are in development.

In open blockchains, every node is usually required to process every transaction. This provides strong security, but severely limits scalability. Sharding [GvRS16, LNZ+16] might alleviate this problem by spreading transactions across groups of nodes (*shards*), which should be large enough to provide a sufficient level of security and a significantly better throughput [Sha16].

The second goal can be addressed by skipping the validation of old blocks [Jun17] or by additionally providing new nodes with full snapshots of a previous state [Par17].

Privacy. Most open blockchains¹⁰, including Ethereum, guarantee integrity and availability, but provide little to no privacy. All transactions are broadcast in plaintext and can be intercepted (or later obtained from the blockchain) and analyzed. Deanonimization of blockchain transactions is an active business area with start-ups (e.g., [Cha17b]) offering blockchain analysis tools, which is in line with government demands of KYC/AML compliance for financial services.

A common but only partially efficient privacy preserving practice in Bitcoin, which takes advantage of the UTXO structure of its state, is to use a new address for every transaction. This technique is not applicable in Ethereum, because it uses addresses for authentication and explicitly maps them to accounts states. For instance, if a user purchases tokens using a particular address, they have to use the same address to redeem them.

An additional privacy challenge comes from the requirement to hide business logic behind smart contract code. Though Ethereum only stores bytecode, users are reluctant to trust contracts without published source code. Moreover, bytecode analysis¹¹ tools are already available [NPS+17, Sui17]. Possible research directions in the privacy domain include privacy preserving smart contracts with zero-knowledge proofs [KMS+15] (support for zero-knowledge proofs in Ethereum was first tested in September 2017 [O’L17]), mixing, computations on encrypted data, and code obfuscation.

¹⁰ Except those using dedicated privacy-preserving cryptographic techniques, e.g., Dash, Monero, Zcash.

¹¹ Decompiling bytecode to source code is hardly possible as the information about function and variable names is lost during compilation; nevertheless it is possible to display bytecode as a sequence of mnemonics or convert it into an intermediate higher-level representation suitable for analysis.

3.2 Smart Contract Programming

Programming Languages. Security is of paramount importance in smart contract programming [ABC17, DAK+15]. Contrary to traditional software, smart contracts can not be patched, which brings new challenges to blockchain programming [PPMT17]. Multiple approaches exist to contract programming [STM16]. Areas of research in this domain include systematizing good and bad programming practices [Con16, CLLZ17], designing general-purpose [Hir17a, But17d, PE16] as well as domain-specific [BKT17, EMEHR17] smart contract programming languages, and developing tools for automated security analysis [LCO+16, Sec17] and formal verification [BDLF+16] of smart contract source code, EVM bytecode, and the EVM itself [Hir17b].

Secure Contract Programming. An important challenge is to describe smart contracts' execution model (possibly drawing parallels from concurrent programming on a multi-threaded processor [SH17]) and to develop a usable and formally verifiable high-level language reflecting this model. Some argue that Solidity inclines programmers towards unsafe development practices [ydt16]. Typical vulnerabilities and issues in Solidity might include:

1. **Re-entrancy.** Contracts can call each other. Malicious external contracts can call the caller back. If the victim contract does its internal bookkeeping after returning from an external call, its integrity can be compromised¹².
2. **Miner's influence.** Miners can to some extent influence execution (front-running, censorship, or altering environmental variables, e.g., timestamp).
3. **Out-of-gas exceptions.** Computation in Ethereum is many orders of magnitude more expensive than with centrally managed cloud computing services. Developers who do not take it into account may implement functions that require too much gas to fit in the block gas limit and thus always fail.

Trusted Data Sources. Many smart contract applications (financial derivatives, insurance, prediction markets) depend on real-world data. Ethereum is isolated from the broader Internet to guarantee consistent execution across nodes. A popular approach to providing data to contracts in a trust-minimizing way is an oracle – a specialized data provider, possibly with a dedicated cryptographic protocol to guarantee integrity [Ora17]. A recent development is TownCrier – an oracle built with trusted hardware [ZCC+16].

3.3 Higher Level Issues

Governance. In June 2016, a massive Ethereum-based crowdfunding project – The DAO – ended in a disaster: an unknown hacker exploited a bug in the smart contract and obtained around \$50 million out of \$150 million collected [Sir16].

¹² This bug led to the DAO hack of 2016.

Despite the fact that the Ethereum protocol correctly executed the smart contract code, the Ethereum developers implemented a hard fork that allowed stakeholders to withdraw their deposits. This event raised concerns about Ethereum’s governance, as the fork violated the premise of decentralized applications running “exactly as programmed” and led to the creation of Ethereum Classic [Eth17c]. Governance mechanisms should provide certainty over how updates (potentially breaking compatibility) are introduced.

Though the gas price in ether is determined by the market, the relative gas costs of EVM bytecodes are constant. In September 2016, an attacker exploited a weakness in gas pricing and organized a DoS attack on the network, taking advantage of the fact that certain operations were under-priced [But16b]. The problem was ultimately fixed with a hard fork. Research is needed to propose more flexible mechanisms for determining relative prices of EVM operations.

Incentives. Open blockchains rely on the participants’ rationality [CXS+17] and must maintain incentive compatibility, so that rational behavior leads to the overall benefit for the network [LTKS15]. This introduces a new field of study dubbed *cryptoeconomics* – the study of incentives in cryptographic systems. The trustless nature of smart contracts might be used for benign (managing mining pools [LVTS17]) as well as for malicious (providing automatic rewards for attacking mining pools [VTL17]) purposes. Rigorous research should guarantee the proper functioning of the blockchain networks and applications based on a definition of rational behavior.

Usability. Considering the influx of new people into the blockchain space, usable yet secure lightweight blockchain software is needed. From the human-computer interaction (HCI) perspective, a challenging task would be to help users grasp the smart contracts fundamentals without going into technicalities. Research shows that cryptographically sound systems may fail to gain traction due to usability issues [RAZS15]. HCI research is needed to make blockchains and smart contracts usable by general public.

Ethical and Legal Issues. Information security researchers usually adhere to the “responsible disclosure” policy: they report a bug privately to the vendor and give developers time to fix it before publishing the information in the open. Though some oppose this practice [Sch07], it is assumed to decrease the probability of an attack on the live system (unless the attackers discover the same bug independently before a patch is applied). Ethereum introduces a new dimension to the responsible disclosure debate, as smart contracts can not be patched. It is unclear whether it is ethical to fully disclose a vulnerability discovered in a smart contract, if developers can not fix it anyway¹³.

¹³ A technical response to this issue could be updateable contracts: users communicate with a proxy contract, which redirects their transactions to the latest version of the main contract. Such scheme assumes that the developers are honest and competent so that the latest update does not run away with everyone’s money.

A whole separate range of topics, which is outside the scope of this paper, is how (and if at all) smart contracts fit into existing legal frameworks. For instance, BitLicense [ofs15] – a controversial [Act15] piece of regulation that came into force in New York in 2015 – prompted many cryptocurrency businesses to withdraw their services from the residents of this US state [Rob15]. In July 2017, the US Securities and Exchange Commission stated that issuers of digital assets may be subject to requirements of the US law [SC17].

4 Conclusion

Ethereum is a fascinating research area at the intersection of multiple fields: cryptography and distributed systems, programming languages and formal verification, economics and game theory, human-computer interaction, finance and law. The promise of smart contracts is not limited to making existing processes more efficient by putting parts of their logic onto a very inefficient, yet very secure decentralized network. This new way of handling value without a trusted third party opens up whole new classes of previously impossible use cases. Thorough research is needed to realize this vision.

References

- [ABC17] Atzei, N., Bartoletti, M., Cimoli, T.: A survey of attacks on Ethereum smart contracts (SoK). In: Maffei, M., Ryan, M. (eds.) POST 2017. LNCS, vol. 10204, pp. 164–186. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54455-6_8
- [Act15] EFF Action. Stop the BitLicense (2015). <https://act.eff.org/action/stop-the-bitlicense>
- [And14] Andreev, O.: Proof that proof-of-work is the only solution to the Byzantine generals’ problem (2014). <http://nakamotoinstitute.org/mempool/proof-that-proof-of-work-is-the-only-solution-to-the-byzantine-generals-problem/>
- [ato16] atomh33ls. Ethereum block architecture (2016). <https://ethereum.stackexchange.com/a/6413/5113>
- [Aug17] Augur (2017). <https://augur.net/>
- [BDLF+16] Bhargavan, K., Delignat-Lavaud, A., Fournet, C., Gollamudi, A., Gonthier, G., Kobeissi, N., Kulatova, N., Rastogi, A., Sibut-Pinote, T., Swamy, N., Zanella-Béguelin, S.: Formal verification of smart contracts: short paper. In: Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, PLAS 2016, pp. 91–96. ACM, New York (2016)
- [BGM16] Bentov, I., Gabizon, A., Mizrahi, A.: Cryptocurrencies without proof of work. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 142–157. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_10
- [Bit16] bitcoin/src/chainparams.cpp (2016). <https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp#L146>

- [Bit17a] Block size limit controversy (2017). https://en.bitcoin.it/wiki/Block_size_limit_controversy
- [Bit17b] Cryptocurrency statistics (2017). <https://bitinfocharts.com/>
- [Bit17c] Bitnodes.21.co. Global Bitcoin nodes distribution (2017). <https://bitnodes.21.co/>
- [BKT17] Biryukov, A., Khovratovich, D., Tikhomirov, S.: Findel: secure derivative contracts for Ethereum (2017). <https://hdl.handle.net/10993/30975>
- [BMC+15] Bonneau, J., Miler, A., Clark, J., Narayanan, A., Kroll, J.A., Felten, E.W.: Research perspectives and challenges for Bitcoin and cryptocurrencies. Cryptology ePrint Archive, Report 2015/261 (2015). <http://eprint.iacr.org/2015/261>
- [BP17] Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. CoRR, abs/1703.06322 (2017)
- [Buc14] Buchman, E.: Understanding the Ethereum trie (2014). <https://easythereentropy.wordpress.com/2014/06/04/understanding-the-ethereum-trie/>
- [But13] Buterin, V.: Dagger: a memory-hard to compute, memory-easy to verify Script alternative (2013). <http://www.hashcash.org/papers/dagger.html>
- [But16a] Buterin, V.: Casper the friendly finality gadget (2016). https://github.com/ethereum/research/blob/master/casper4/papers/casper_paper.md
- [But16b] Buterin, V.: Long-term gas cost changes for IO-heavy operations to mitigate transaction spam attacks (2016). <https://github.com/ethereum/eips/issues/150>
- [But17a] Buterin, V.: Casper the friendly finality gadget (2017). http://vitalik.ca/files/casper_note.html
- [But17b] Buterin, V.: Design rationale (2017). <https://github.com/ethereum/wiki/wiki/Design-Rationale>
- [But17c] Buterin, V.: Minimal slashing conditions (2017). <https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c>
- [But17d] Buterin, V.: New experimental programming language (2017). <https://github.com/ethereum/viper>
- [Cas17] Castor, A.: One of Ethereum's earliest smart contract languages is headed for retirement (2017). <https://www.coindesk.com/one-of-ethereums-earliest-smart-contract-languages-is-headed-for-retirement/>
- [Cha17a] Chain (2017). <https://chain.com/>
- [Cha17b] Protecting the integrity of digital assets (2017). <https://www.chainalysis.com/>
- [Chu36] Church, A.: A note on the Entscheidungs problem. *J. Symb. Logic* **1**(1), 40–41 (1936)
- [CL02] Castro, M., Liskov, B.: Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* **20**(4), 398–461 (2002)
- [CLLZ17] Chen, T., Li, X., Luo, X., Zhang, X.: Under-optimized smart contracts devour your money. In: SANER, pp. 442–446. IEEE Computer Society (2017)
- [Coi17a] Cryptocurrency market capitalizations (2017). <https://coinmarketcap.com/>
- [Coi17b] Coindesk. ICO tracker (2017). <https://www.coindesk.com/ico-tracker/>
- [Con16] Ethereum contract security techniques and tips (2016). <https://github.com/ConsenSys/smart-contract-best-practices>

- [Cor17] Corda (2017). <https://www.corda.net/>
- [CXS+17] Chen, L., Xu, L., Shah, N., Gao, Z., Lu, Y., Shi, W.: Decentralized execution of smart contracts: agent model perspective and its implications (2017). <http://fc17.ifca.ai/wtsc/Decentralized%20Execution%20of%20Smart%20Contracts%20-%20Agent%20Model%20Perspective%20and%20Its%20Implications.pdf>
- [DAG17] Dag file size calculator (2017). https://investoon.com/tools/dag_size
- [DAK+15] Delmolino, K. Arnett, M., Kosba, A., Miller, A., Shi, E.: Step by step towards creating a safe smart contract: lessons and insights from a cryptocurrency lab. Cryptology ePrint Archive, Report 2015/460 (2015). <http://eprint.iacr.org/2015/460>
- [DFZ16] Duong, T., Fan, L., Zhou, H.-S.: 2-hop blockchain: combining proof-of-work and proof-of-stake securely. Cryptology ePrint Archive, Report 2016/716 (2016). <http://eprint.iacr.org/2016/716>
- [Dig17] Digiconomist. Bitcoin energy consumption index (2017). <http://digiconomist.net/bitcoin-energy-consumption>
- [doc17] Ethereum documentation. Mining (2017). <http://ethdocs.org/en/latest/mining.html>
- [Dom17] Domchi. What are the ethereum disk space needs? (2017). <https://ethereum.stackexchange.com/q/143/5113>
- [DPS16] Daian, P., Pass, R., Shi, E.: Snow white: provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919 (2016). <http://eprint.iacr.org/2016/919>
- [Dry14] Dryja, T.: Hashimoto: I/O bound proof of work (2014). <https://pdfs.semanticscholar.org/3b23/7cc60c1b9650e260318d33bec471b8202d5e.pdf>
- [Ell17] Ellison, D.: An introduction to LLL for Ethereum smart contract development (2017). <https://media.consensys.net/an-introduction-to-lll-for-ethereum-smart-contract-development-e26e38ea6c23>
- [EMEHR17] Egelund-Müller, B., Elsmann, M., Henglein, F., Ross, O.: Automated execution of financial contracts on blockchains (2017). <https://ssrn.com/abstract=2898670>
- [ENS17] ENS (2017). <https://ens.domains/>
- [ES13] Eyal, I., Gün Sirer, E.: Majority is not enough: Bitcoin mining is vulnerable. CoRR, abs/1311.0243 (2013)
- [ETC16] The Ethereum Classic declaration of independence (2016). https://ethereumclassic.github.io/assets/ETC_Declaration_of_Independence.pdf
- [eth16] eth. How is the address of an ethereum contract computed? (2016). <https://ethereum.stackexchange.com/q/760/5113>
- [Eth17a] Ethash (2017). <https://github.com/ethereum/wiki/wiki/Ethash>
- [Eth17b] Ethash design rationale (2017). <https://github.com/ethereum/wiki/wiki/Ethash-Design-Rationale>
- [Eth17c] Ethereum Classic (2017). <https://ethereumclassic.github.io/>
- [Eth17d] Etherchain.org. Mining statistics (last 24h) (2017). <https://etherchain.org/statistics/miners>
- [Eth17e] Ethernodes.org (2017). <https://www.ethernodes.org/network/1>
- [Fil17] Filecoin (2017). <https://filecoin.io/>
- [Gno17] Gnosis (2017). <https://gnosis.pm/>
- [Gol17] Golem (2017). <https://golem.network/>
- [GvRS16] Gencer, A.E., van Renesse, R., Sirer, E.G.: Service-oriented sharding with Aspen. arXiv preprint [arXiv:1611.06816](https://arxiv.org/abs/1611.06816) (2016)

- [Her17] Hertig, A.: Ethereum’s big switch: the new roadmap to proof-of-stake (2017). <https://www.coindesk.com/ethereums-big-switch-the-new-roadmap-to-proof-of-stake/>
- [Hir17a] Hirai, Y.: Bamboo: a morphing smart contract language (2017). <https://github.com/pirapira/bamboo>
- [Hir17b] Hirai, Y.: Formal verification of Ethereum contracts (2017). <https://github.com/pirapira/ethereum-formal-verification-overview>
- [Hyp17] Hyperledger (2017). <https://www.hyperledger.org/>
- [jnk15] jnk. What is gas limit in Ethereum? (2015). <https://bitcoin.stackexchange.com/a/39197>
- [Joh17] Johnson, N.: What is the exact “longest chain” rule implemented in the ethereum “homestead” protocol? (2017). <https://ethereum.stackexchange.com/a/13750/5113>
- [Jun17] Junge, H.: What is Geth’s “light” sync, and why is it so fast? (2017). <https://ethereum.stackexchange.com/a/11300>
- [KMS+15] Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. Cryptology ePrint Archive, Report 2015/675 (2015). <http://eprint.iacr.org/2015/675>
- [KRDO16] Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889 (2016). <http://eprint.iacr.org/2016/889>
- [LCO+16] Luu, L., Chu, D.-H., Olickel, H., Saxena, P., Hobor, A.: Making smart contracts smarter. Cryptology ePrint Archive, Report 2016/633 (2016). <http://eprint.iacr.org/2016/633>
- [Ler14] Lerner, S.D.: Ethereum “Dagger” PoW function is flawed (2014). <https://bitslog.wordpress.com/2014/01/17/ethereum-dagger-pow-is-flawed/>
- [Lig16] Lightning network (2016). <https://lightning.network/>
- [LNZ+16] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A secure sharding protocol for open blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 17–30. ACM (2016)
- [LTKS15] Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. Cryptology ePrint Archive, Report 2015/702 (2015). <http://eprint.iacr.org/2015/702>
- [LVTS17] Luu, L., Velner, Y., Teutsch, J., Saxena, P.: SmartPool: practical decentralized pooled mining. Cryptology ePrint Archive, Report 2017/019 (2017). <http://eprint.iacr.org/2017/019>
- [May16] Mayer, H.: ECDSA security in Bitcoin and Ethereum: a research survey (2016)
- [Maz14] Mazières, D.: The Stellar consensus protocol: a federated model for internet-level consensus (2014). <https://www.stellar.org/papers/stellar-consensus-protocol.pdf>
- [McA17] McAdams, D.: An ontology for smart contracts (2017). <https://iohk.io/research/papers/#QCNr6SCZ>
- [McC15] McCone, R.: Ethereum Lightning network and beyond (2015). <http://www.arcturnus.com/ethereum-lightning-network-and-beyond/>
- [MHWK17] Milutinovic, M., He, W., Wu, H., Kanwal, M.: Proof of luck: an efficient blockchain consensus protocol. Cryptology ePrint Archive, Report 2017/249 (2017). <http://eprint.iacr.org/2017/249>

- [Mic16] Micali, S.: ALGORAND: the efficient and democratic ledger. CoRR, abs/1607.01341 (2016)
- [Nak08] Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008). <https://bitcoin.org/bitcoin.pdf>
- [Nol17] Landon Curt Noll. FNV hash (2017). <http://www.isthe.com/chongo/tech/comp/fnv/index.html>
- [NPS+17] Norvill, R., Pontiveros, B.B.F., State, R., Awan, I., Cullen, A.: Automated labeling of unknown contracts in Ethereum (2017). <https://bradscholars.brad.ac.uk/handle/10454/12220>
- [ofs15] Department of financial services. Bitlicense regulatory framework (2015). http://www.dfs.ny.gov/legal/regulations/bitlicense_reg_framework.htm
- [O’L17] O’Leary, R.R.: Ethereum’s Byzantium testnet just verified a private transaction (2017). <https://www.coindesk.com/ethereums-byzantium-testnet-just-verified-private-transaction/>
- [Ora17] Oraclize (2017). <http://www.oraclize.it/>
- [Par17] Warp sync snapshot format (2017). <https://github.com/paritytech/parity/wiki/Warp-Sync-Snapshot-Format>
- [PE16] Pettersson, J., Edström, R.: Safer smart contracts through type-driven development (2016). <https://publications.lib.chalmers.se/records/fulltext/234939/234939.pdf>
- [PHC15] Password hashing competition (2015). <https://password-hashing.net/>
- [PoS16] Proof of stake FAQ (2016). <https://github.com/ethereum/wiki/wiki/Proof-of-Stake-FAQ>
- [PPMT17] Porru, S., Pinna, A., Marchesi, M., Tonelli, R.: Blockchain-oriented software engineering: challenges and new directions. CoRR, abs/1702.05146 (2017)
- [Qtu17] Qtum (2017). <https://qtum.org/en/>
- [Rai17] Raiden network: high speed asset transfers for Ethereum (2017). <http://raiden.network/>
- [RAZS15] Ruoti, S., Andersen, J., Zappala, D., Seamons, K.E.: Why Johnny still, still can’t encrypt: evaluating the usability of a modern PGP client. CoRR, abs/1510.08555 (2015)
- [Rob15] Roberts, D.: Behind the “exodus” of Bitcoin startups from New York (2015). <http://fortune.com/2015/08/14/bitcoin-startups-leave-new-york-bitlicense/>
- [Roo17] Rootstock (2017). <http://www.rsk.co/>
- [SC17] U.S. Securities and Exchange Commission. SEC issues investigative report concluding DAO tokens, a digital asset, were securities (2017). <https://www.sec.gov/news/press-release/2017-131>
- [Sch07] Schneier, B.: Debating full disclosure (2007). https://www.schneier.com/blog/archives/2007/01/debating_full_d.html
- [Sec17] Securify. Formal verification of Ethereum smart contracts (2017). <http://securify.ch/>
- [Ser17] Serpent (2017). <https://github.com/ethereum/wiki/wiki/Serpent>
- [SH17] Sergey, I., Hobor, A.: A concurrent perspective on smart contracts. CoRR, abs/1702.05511 (2017)
- [Sha16] Sharding FAQ (2016). <https://github.com/ethereum/wiki/wiki/Sharding-FAQ>
- [SHA17] SHA-3 competition (2007–2012) (2017). <http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>

- [Sia17] Sia (2017). <https://sia.tech/>
- [Sil16] Siludin. Let's talk about how poor this network is at handling any type of major transaction traffic (2016). <https://redd.it/6ifl5f>
- [Sir16] Gün Sırer, E.: Thoughts on The DAO hack (2016). <http://hackingdistributed.com/2016/06/17/thoughts-on-the-dao-hack/>
- [Sol17] Solidity official documentation (2017). <https://solidity.readthedocs.io/>
- [Son17] Sonm (2017). <https://sonm.io/>
- [STM16] Seijas, P.L., Thompson, S., McAdams, D.: Scripting smart contracts for distributed ledger technology. Cryptology ePrint Archive, Report 2016/1156 (2016). <http://eprint.iacr.org/2016/1156>
- [Sto17] Storj (2017). <https://storj.io/>
- [Sui17] Suiche, M.: Porosity. Decompiling Ethereum smart-contracts (2017). <https://blog.comae.io/porosity-18790ee42827>
- [Sun17] Sunnarborg, A.: ICO investments pass VC funding in blockchain market first (2017). <https://www.coindesk.com/ico-investments-pass-vc-funding-in-blockchain-market-first/>
- [SYB14] Schwartz, D., Youngs, N., Britto, A.: The Ripple protocol consensus algorithm. Ripple Labs Inc White Paper (2014). https://ripple.com/files/ripple_consensus_whitepaper.pdf
- [SZ13] Sompolinsky, Y., Zohar, A.: Accelerating Bitcoin's transaction processing. Fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881 (2013). <http://eprint.iacr.org/2013/881>
- [Sza17] Szabo, N.: Money, blockchains, and social scalability (2017). <https://unenumerated.blogspot.lu/2017/02/money-blockchains-and-social-scalability.html>
- [Szt15] Sztorc, P.: Nothing is cheaper than proof of work (2015). <http://www.truthcoin.info/blog/pow-cheapest/>
- [TS15] Tschorsch, F., Scheuermann, B.: Bitcoin and beyond: a technical survey on decentralized digital currencies. Cryptology ePrint Archive, Report 2015/464 (2015). <http://eprint.iacr.org/2015/464>
- [VB+14] Vogelsteller, F., Buterin, V., et al.: Ethereum whitepaper (2014). <https://github.com/ethereum/wiki/wiki/White-Paper>
- [VBR+17] Vogelsteller, F., Buterin, V., Reitwiessner, C., Kotewicz, M., et al.: Merkle Patricia trie specification (2017). <https://github.com/ethereum/wiki/wiki/Patricia-Tree>
- [Vog17] Vogelsteller, F.: ERC: token standard (2017). <https://github.com/ethereum/eips/issues/20>
- [VTL17] Velner, Y., Teutsch, J., Luu, L.: Smart contracts make Bitcoin mining pools vulnerable. Cryptology ePrint Archive, Report 2017/230 (2017). <http://eprint.iacr.org/2017/230>
- [Woo14] Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 151 (2014). <http://yellowpaper.io/>
- [Wui17] Wuille, P.: What does the term "longest chain" mean? (2017). <https://bitcoin.stackexchange.com/a/5542/31712>
- [ydt16] ydtm. The bug which the DAO hacker exploited was not merely in the DAO itself (2016). <https://redd.it/4opjov>
- [ZCC+16] Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town Crier: an authenticated data feed for smart contracts. Cryptology ePrint Archive, Report 2016/168 (2016). <http://eprint.iacr.org/2016/168>

Vulnerability Analysis and Deception Systems



Bounding the Cache-Side-Channel Leakage of Lattice-Based Signature Schemes Using Program Semantics

Nina Bindel, Johannes Buchmann, Juliane Krämer, Heiko Mantel, Johannes Schickel, and Alexandra Weber^(✉)

Computer Science Department, TU Darmstadt, Darmstadt, Germany
{nbindel,buchmann,jkraemer}@cdc.informatik.tu-darmstadt.de,
{mantel,schickel,weber}@mais.informatik.tu-darmstadt.de

Abstract. In contrast to classical signature schemes, such as RSA or ECDSA signatures, the lattice-based signature scheme ring-TESLA is expected to be resistant even against quantum adversaries. Due to a recent key recovery from a lattice-based implementation, it becomes clear that cache side channels are a serious threat for lattice-based implementations. In this article, we analyze an existing implementation of ring-TESLA against cache side channels. To reduce the effort for manual code inspection, we selectively employ automated program analysis. The leakage bounds we compute with program analysis are sound overapproximations of cache-side-channel leakage. We detect four cache-side-channel vulnerabilities in the implementation of ring-TESLA. Since two vulnerabilities occur in implementations of techniques common to lattice-based schemes, they are also interesting beyond ring-TESLA. Finally, we show how the detected vulnerabilities can be mitigated effectively.

1 Introduction

The threat posed by quantum computers to current public-key cryptography is known since Shor presented a quantum algorithm to solve the factorization and the discrete logarithm problem in polynomial time [1]. How serious this threat is taken became clear, e.g., when NIST announced to start a standardization process for quantum-resistant schemes beginning in fall 2017 [2].

A promising quantum-resistant, also called post-quantum, candidate to substitute current public-key cryptography is lattice-based cryptography that enjoys, among other things, strong security guarantees. However, security guarantees can be undermined by side-channel vulnerabilities at the implementation level. So far, this has happened mostly for implementations of classical cryptography, e.g., [3–6]. However, first approaches to analyze lattice-based cryptography with respect to side-channel attacks are already made, e.g., [7–10]. Recently, Groot Bruinderink et al. presented the first attack against a lattice-based signature scheme and broke the scheme BLISS [11] using cache side channels of the Gaussian sampling during the signature generation [12]. Although none of

the existing lattice-based signature schemes (or their implementations) claim to be secure against side channels, the attack in [12] raises the question how lattice-based signature schemes can be implemented without cache-side-channel leakage. Furthermore, in the light of NIST’s standardization process, it is important to analyze lattice-based implementations against cache side channels. The scheme ring-TESLA [13], which is one of the most efficient lattice-based signature schemes, seems to be a good candidate to be implemented without cache side channels: During signature generation, ring-TESLA does not use Gaussian sampling, the sampling method that was exploited in the attack on BLISS in [12].

In this work, we use program analysis to compute upper bounds on the cache-side-channel leakage of lattice-based implementations at the example of the signature generation in ring-TESLA¹. More concretely, we follow an approach based on information theory and reachability analysis, which is implemented in CacheAudit [16]. Variants of CacheAudit were used to analyze multiple cryptographic implementations. CacheAudit 0.2 [16] was used to analyze PolarSSL AES and the eSTREAM Profile 1 portfolio (HC-128, Rabbit, Salsa20, and Sosemanuk). CacheAudit 0.2b was used in a systematic study of AES implementations [17]. Another extension of CacheAudit 0.2 was used on modular exponentiation from the libraries libgcrypt and OpenSSL [18]. In this article, we extend CacheAudit 0.2b to CacheAudit 0.2c and apply it to ring-TESLA. This is the first analysis of a post-quantum scheme using CacheAudit.

With CacheAudit 0.2c, we determine upper bounds on the leakage of a ring-TESLA implementation for four attacker models. The bounds are sound, i.e., conservative with respect to the attacker models. We obtain upper bounds between 2.6 bit and 51.6 bit of potential cache-side-channel leakage. By inspecting the code manually, we then identify vulnerable subroutines. We implement countermeasures in the vulnerable subroutines to mitigate the cache-side-channel leakage. Finally, we argue for the effectiveness of the mitigations. For two subroutines, the argument is completely automated by an analysis with CacheAudit 0.2c that reports 0 bit leakage. According to our code inspection, a potential for leakage in the signature computation remains, which is intrinsic to a method called rejection sampling. Rejection sampling is used by design in the most efficient lattice-based signature schemes, such as [11, 13, 15, 19]. We argue that the attacker cannot exploit this potential leakage to get information about the secret key. Therefore, we consider our resulting implementation of ring-TESLA to be resistant to the four types of cache-side-channel attacks we consider.

In summary, the contributions of this article are the following.

- We detect four cache-side-channel vulnerabilities in an existing implementation of ring-TESLA by code inspection, selectively supported by automatic program analysis.

¹ We analyze an implementation of ring-TESLA despite an error that was detected in its security reduction, since we expect that reductions given for its predecessor TESLA [14, 15] will be applicable to ring-TESLA as well. Hence, we consider it to be a good candidate for practical applications that require post-quantum signatures.

- To mitigate the detected vulnerabilities, we augment the ring-TESLA implementation by side-channel countermeasures. We argue for the effectiveness of the countermeasures, again supported by selective program analysis.
- To automate parts of our analysis of the unmitigated and mitigated ring-TESLA implementation, we extend the analysis tool CacheAudit 0.2b to CacheAudit 0.2c. More concretely, we implement support for ten additional x86 instructions. The support can be used to analyze occurrences of these instructions in x86 binaries and is not limited to ring-TESLA.

The detection and mitigation of vulnerabilities not only hardens the ring-TESLA implementation against side-channel attacks. Multiple lattice-based primitives, such as key exchange protocols, encryption, and signature schemes, use techniques similar to the ones we analyze in ring-TESLA. In particular, rejection sampling and sparse multiplication, where we find two of the potential vulnerabilities, occur in ring-TESLA, as well as in other lattice-based primitives. Hence, our results also pave the way to make other lattice-based implementations more trustworthy using program analysis.

2 Preliminaries

2.1 Notation

For an integer $n \in \mathbb{N}$, we define $q \in \mathbb{N}$ to be a prime with $q \equiv 1 \pmod{2n}$. We denote the finite field $\mathbb{Z}/q\mathbb{Z}$ with representatives in $[-q/2, q/2] \cap \mathbb{Z}$ by \mathbb{Z}_q . Furthermore, we define $\mathcal{R}_q = \mathbb{Z}_q[x]/\langle x^n + 1 \rangle$ and $\mathcal{R}_{q,[B]} = \{\sum_{i=0}^{n-1} a_i x^i \mid a_i \in [-B, B] \cap \mathbb{Z}\}$ for $B \in [0, q/2] \cap \mathbb{Z}$ and $\mathbb{B}_{n,\omega} = \{\sum_{i=0}^{n-1} a_i x^i \mid a_i \in \{-1, 0, 1\}, \sum_{i=0}^{n-1} |a_i| = \omega\}$ for $\omega \in [0, n] \cap \mathbb{Z}$. All logarithms are in base 2. Let $\sigma \in \mathbb{R}_{>0}$. Let v be a polynomial, then $v \leftarrow_{\sigma} \mathcal{R}$ means sampling each coefficient of v with discrete Gaussian distribution with standard deviation σ and mean 0 over \mathbb{Z} . For a finite set S , we write $s \leftarrow_{\S} S$ to indicate that an element s is sampled uniformly at random from S .

2.2 Description of Ring-TESLA

The signature scheme ring-TESLA is parametrized by $n, \omega, d, B, q, U, L, \kappa, \sigma$, by the hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^{\kappa}$, and by the encoding function $F : \{0, 1\}^{\kappa} \rightarrow \mathbb{B}_{n,\omega}$, see Fig. 1. For detailed information about the system parameters and the encoding function F , we refer to the original work [13]. For $c \in \mathbb{Z}$, we denote by $[c]_L$ the unique representative of c in $(-2^{d-1}, 2^{d-1}] \cap \mathbb{Z}$ such that $c = [c]_L$ modulo 2^d and define $[\cdot]_M : \mathbb{Z} \rightarrow \mathbb{Z}, c \mapsto (c - [c]_L)/2^d$. The operators $[\cdot]_L$ and $[\cdot]_M$ correspond to the *least* or *most* significant bits, respectively. We extend the definitions to polynomials by applying $[\cdot]_L$ and $[\cdot]_M$ to each coefficient.

The secret key sk is a tuple of three polynomials $s, e_1, e_2 \leftarrow_{\sigma} \mathcal{R}$, where the entries of e_1 and e_2 have to be small enough for the scheme to be correct; the public key pk consists of the polynomials $a_1, a_2 \leftarrow_{\S} \mathcal{R}_q, b_1 = a_1 s + e_1 \pmod{q}$, and $b_2 = a_2 s + e_2 \pmod{q}$. We depict the algorithm to generate a signature

 Sign($\mu; a_1, a_2, s, e_1, e_2$) :

1 $y \leftarrow \mathcal{R}_{q,[B]}$ 2 $v_1 \leftarrow a_1 y \pmod{q}$ 3 $v_2 \leftarrow a_2 y \pmod{q}$ 4 $c' \leftarrow H([v_1]_M, [v_2]_M, \mu)$ 5 $c \leftarrow F(c')$ 6 $z \leftarrow y + sc$	7 $w_1 \leftarrow v_1 - e_1 c \pmod{q}$ 8 $w_2 \leftarrow v_2 - e_2 c \pmod{q}$ 9 If $[w_1]_L, [w_2]_L \notin \mathcal{R}_{q,[2^d-L]} \vee z \notin \mathcal{R}_{q,[B-U]}$: 10 Restart 11 Return (z, c')
---	---

Fig. 1. Specification of the scheme ring-TESLA [13].

(c', z) for message μ in Fig. 1. For verification of the signature (c', z) , it is checked that $z \in \mathcal{R}_{q,[B-U]}$ and that c' equals $H([a_1 z - b_1 c]_M, [a_2 z - b_2 c]_M, \mu)$, with $c = F(c')$. The parameters proposed for ring-TESLA [13] are currently not supported by a security reduction, since in November 2016 an error was detected in the existing reduction. However, we expect that existing security reductions for ring-TESLA's predecessor TESLA [14, 15] are applicable to ring-TESLA as well². Our modifications described in Sects. 3.1 and 4 do not depend on the *values* of the parameters. Hence, our results can be applied to other parameter sets of ring-TESLA as long as all values can be represented by the data type int.

2.3 Cache Side Channels

A cache is a small piece of memory that stores selected entries from main memory for quick access by the Central Processing Unit (CPU). If the CPU accesses a memory entry, the access can lead to a cache hit (if the entry is stored in the cache) or to a cache miss (if the entry is not stored in the cache). Inside the cache, the memory entries are stored in sections called cache lines. The sequence of cache lines in a cache is partitioned into cache sets. In a k -way set-associative cache, each cache set consists of k cache lines. A cache has a strategy for replacing entries if the cache is full. A popular strategy is to replace the least recently used entry (LRU strategy). Variants of LRU are used, e.g., in Intel processors [20].

A cache-side-channel vulnerability exists if the interaction between a program and the cache depends on secret information, e.g., on a cryptographic key. In this case, an attacker, observing aspects of this interaction, might learn secret information. Attacks on cryptographic implementations have exploited secret-dependence in the trace of cache hits and misses [4], the time taken for cache hits and misses [6], and the final cache state of an execution [5].

² Security properties of schemes over standard lattices (like TESLA) often hold for corresponding schemes over ideal lattices (like ring-TESLA), e.g., the security reduction from [11] holds for the standard-lattice variant and for the ideal-lattice variant.

2.4 Leakage Bounds on Cache Side Channels

In this article, we follow an approach based on information theory and reachability analysis to compute upper bounds on the cache-side-channel leakage of ring-TESLA (compiled to an x86 binary). Let Obs^a be the set of possible observations an attacker a can make about a single run of an x86 binary. Then $\log_2 |Obs^a|$ is an upper bound on the leakage of the binary with respect to min-entropy [21] and Shannon entropy [22] by [23, Theorem 1] and [24, Theorem 5.3].

We consider the following attacker models $a \in \{acc, aced, trace, time\}$ [16]:

aced generalizes techniques like EVICT+TIME and PRIME+PROBE [5]. More concretely, it captures attackers who can determine the number of memory blocks in each cache set in the final cache state after a program execution.

acc captures attackers who can determine the position of each memory block in the final cache state, inspired by techniques like FLUSH+RELOAD [25].

trace captures trace-based attackers who can determine the trace of cache hits and misses that occur during one program execution. For instance, the trace-based attack in [4] uses such traces of hits and misses.

time models time-based attackers who can observe the running time of one program execution. Actual running times, as used in attacks like [6], are modeled by the amount of cache hits and cache misses that occur.

The possible observations under an attacker model can be computed by reachability analysis [16]. Let \mathcal{D} be the set of the possible states during an execution and let $\text{upd}_{\mathcal{D}} : \mathcal{D} \rightarrow \mathcal{D}$ model the concrete semantics of x86 instructions, i.e., how the execution of instructions updates the state. The possible attacker observations depend on the states that an execution can reach according to $\text{upd}_{\mathcal{D}}$.

Instead of implementing a reachability analysis from scratch, we extend the existing tool CacheAudit 0.2b [17] – a version of CacheAudit [16]. CacheAudit performs a reachability analysis using abstract interpretation [26].

For an abstract reachability analysis, an abstract domain $\overline{\mathcal{D}}$ is defined, which abstracts from details of the concrete execution that are not relevant for the analysis. An abstraction function and a concretization function are defined to convert states between \mathcal{D} and $\overline{\mathcal{D}}$. To represent executions in the abstract domain $\overline{\mathcal{D}}$, an abstract semantics $\text{upd}_{\overline{\mathcal{D}}} : \overline{\mathcal{D}} \rightarrow \overline{\mathcal{D}}$ is defined. To allow a transfer of analysis results from the abstract domain to the concrete domain, the abstract semantics $\text{upd}_{\overline{\mathcal{D}}}$ should be sound with respect to the concrete semantics $\text{upd}_{\mathcal{D}}$, i.e., it should overapproximate the set of reachable states in an execution.

CacheAudit uses multiple abstract domains [16]. The position of a memory block in the cache is abstracted by a set of possible positions. The values of registers and memory entries are abstracted by sets of possible values.

3 Enabling the Automatic Analysis of Ring-TESLA

3.1 Integer Implementation of Ring-TESLA

We obtained the implementation of ring-TESLA [13] from the authors. The original implementation makes use of floating point operations. CacheAudit 0.2b, however, cannot analyze floating point operations and can therefore not be used

Listing 3.1. Signature generation in `crypto_sign`

```

1  [...]
2  while(1) {
3      sample_y(vec_y);
4      poly_mul_fixed(vec_v1, vec_y, poly_a1);
5      poly_mul_fixed(vec_v2, vec_y, poly_a2);
6      random_oracle(c, vec_v1, vec_v2, m, mlen);
7      generate_c(pos_list, c);
8
9      computeEc(E1c, sk+sizeof(int)*PARAM_N, pos_list);
10     poly_sub(vec_v1,vec_v1, E1c);
11     if (test_w(vec_v1) != 0){ continue; }
12
13     computeEc(E2c, sk+sizeof(int)*PARAM_N*2, pos_list);
14     poly_sub(vec_v2,vec_v2, E2c);
15     if (test_w(vec_v2) != 0){ continue; }
16
17     computeEc(Sc, sk, pos_list);
18     poly_add(vec_y, vec_y, Sc);
19     if (test_rejection(vec_y) != 0){ continue; }
20
21     for(i=0; i<mlen; i++){ sm[i]=m[i]; }
22     *smLen = CRYPTO_BYTES + mlen;
23     compress_sig(sm+mlen, c, vec_y);
24     return 0; }

```

to analyze the original implementation directly. An extension of CacheAudit 0.2b to support floating point instructions is out of scope for this article. Changing from floating point to integer operations does not affect the security of the signature scheme, since all operations during the signature generation (cf. Fig. 1) are over \mathbb{Z}_q . Moreover, it is a step towards a ring-TESLA implementation for devices without floating point unit, e.g., embedded devices. We replaced all floating point instructions by integer instructions. The resulting implementation can be analyzed with CacheAudit 0.2c (our extension of CacheAudit 0.2b).

Listing 3.1 shows the parts of the signature generation function `crypto_sign` that are most important for our analysis, leaving out variable declarations.

3.2 Extension of CacheAudit 0.2b

The implementation of the scheme ring-TESLA is the first implementation of post-quantum cryptography (and of lattice-based cryptography) that is analyzed with CacheAudit. The implementation of ring-TESLA contains x86 instructions that are not supported by CacheAudit 0.2b. We extended CacheAudit 0.2b to CacheAudit 0.2c by adding support for these instructions.

To add support for additional x86 instructions to CacheAudit 0.2b, the underlying abstract semantics $\text{upd}_{\overline{\mathcal{D}}}$ must be extended. We implemented abstract semantics for the instructions in the ring-TESLA binary that are unsupported in CacheAudit 0.2b. Table 1 lists the opcodes (unique identifiers) and mnemonics (human-readable descriptions) of the instructions that we added.³

³ The instructions 0xF7/3 and 0x99 were integrated independently but concurrently into a different version of CacheAudit by Doychev [27].

Table 1. Additional instructions for ring-TESLA in CacheAudit 0.2c

Type	Opcodes (and mnemonics) of additional instructions
Arithmetic	13 (Adc), 1B (Sbb), 6B (Imul), F7/3 (Neg), F7/4 (Mul), F7/5 (Imul)
Bit string	0FBD (Bsr), 99 (Cdq)
Move	0F9C (Setl), 0F9F (Setg)

We illustrate the process of extending $\text{upd}_{\mathcal{D}}$ at the example of the instruction **Bsr** (Bit scan reverse), which takes the operands **dst** and **src**. The concrete semantics $\text{upd}_{\mathcal{D}}$ of **Bsr dst src** is to compute the index of the most significant bit that is set, i.e., non-zero, in **src** [28]. If such a bit exists in **src**, its index is written to **dst** and the zero flag is set to 0. Otherwise, the zero flag is set to 1.

To support **Bsr**, we extended the parser, the internal instruction representation, and the abstract semantics in CacheAudit 0.2b. We extended the parser to create a **Bsr** instruction in the internal representation when it encounters the opcode 0FBD. We implemented the abstract semantics of **Bsr** by a function **bsr** in the module **valAD**. The function consists of roughly 100 lines of OCaml code.

The function **bsr** operates on sets of potential values for **dst** and **src** and returns a map from possible resulting status flag combinations to the resulting values of registers and memory entries, for which the flag combinations can occur. For each possible value of **src**, we proceed according to the formalization of **Bsr** by Degenbaev [28]. We check whether the value consists only of zeros. In this case, we add a binding (mapping a flag combination to register and memory values) to the resulting map, in which the zero flag is 1 and the value of **dst** is unchanged. Otherwise, we first compute the number of leading zeros by divide and conquer, where we check recursively whether the first half of each non-zero prefix contains bits that are set to 1. The index of the most significant set bit is 64 minus the number of leading zeros. In this case, we add a binding to the resulting map, in which the zero flag is 0 and **dst** contains the computed index.

Our implementation for the other instructions follows the same pattern of parsing and abstract semantics, reusing existing support for similar instructions (e.g., with the same mnemonic) in CacheAudit 0.2b when possible.

4 Detection of Potential Leakage

We use CacheAudit 0.2c to analyze the signature generation in ring-TESLA for potential leakage of the secret key. We assume that the random number generator is secure and analyze the remaining computation with a few adaptations that allow a meaningful analysis with CacheAudit 0.2c. In the following, we provide details on the configurations of CacheAudit and ring-TESLA, details of our adaptations, and the results of our analysis.

Configuration of CacheAudit. We configure CacheAudit to use a 32 kByte, 8-way set-associative data cache with a cache line size of 64 Byte. This cache

Listing 4.1. Code of the subroutine `generate_c`

```

1 void generate_c(uint32_t *pos_list, unsigned char *c_bin){
2   int32_t c[PARAM_N]; int cnt =0; int pos; [...]
3   crypto_stream(r, R_LENGTH, nonce, c_bin);
4
5   for(i=0; i<PARAM_N; i++){ c[i] = 0;}
6   i=0;
7   while(i<PARAM_W){
8     pos = 0;
9     pos = (r[cnt]<<8) | (r[cnt+1]);
10    pos &= PARAM_N-1;
11    cnt += 2;
12    if (c[pos] == 0) { pos_list[i] = pos; c[pos]=1; i++; cnt++; } }

```

configuration is, e.g., used in the first level cache of the Intel Skylake architecture [29]. As the replacement strategy, we fix LRU.⁴

Configuration of ring-TESLA. We set the parameters of the ring-TESLA scheme to $\text{PARAM_N} = n = 512$, $\text{PARAM_SIGMA} = \sigma = 48$, $\text{PARAM_Q} = q = 33550337$, $\text{PARAM_B} = B = 4194303$, $\text{PARAM_W} = \omega = 19$, $\text{PARAM_D} = d = 23$, and $\text{PARAM_U} = U = 2848$. We analyze the function `crypto_sign` from the file `sign.c` in a 32-bit x86 binary of the ring-TESLA implementation. To this end, we use a wrapper function that calls `crypto_sign` with an uninitialized secret key, uninitialized message, uninitialized signature buffer, the message size 59 (as in the ring-TESLA test suite), and a pointer to `smLen` to store the length of the signed message including the signature. By leaving the secret key and message uninitialized, we treat them as secret input in our analysis.⁵

We compiled the ring-TESLA sources and our wrapper with `gcc` version 4.8.4, using `-static` for static linking, `-m32` to target an Intel i386 CPU architecture, and `-fno-stack-protector` to avoid insertion of code for overflow protection.

Adaptation of ring-TESLA. CacheAudit 0.2c does not support memory accesses that could refer to any possible address, e.g., in the analysis of loop counters that are advanced only under certain conditions and used to index array accesses. This occurs in the ring-TESLA routines `generate_c` and `sample_y`.

Listing 4.1 shows the implementation of the function `generate_c`. It uses rejection sampling to generate random values for the parameter array `pos_list`. The loop counter is only increased if the generated value is not rejected. To allow a meaningful analysis, we remove the check that rejects if the same value would occur twice in `pos_list` (highlighted in gray). That is, we overapproximate the possible values of `pos_list`. This manual overapproximation of the semantics

⁴ We also investigated FIFO (first in first out) replacement. The leakage bounds (on the unmitigated implementation) are less than 10 bit lower than under LRU.

⁵ Treating the message as secret is overly conservative in a signature scenario. We investigated the effect of fixing the message to all ‘0’s and obtained the same leakage bounds as for an uninitialized message in the unmitigated implementation.

Listing 4.2. Implementation of `sample_y`

```

1 //original
2 do {[...] if(val<0x7fffff) mat_y[i++] = val-PARAM_B; [...]} while(i<PARAM_N);
3
4 //adapted
5 for (i = 0; i < PARAM_N; ++i) { mat_y[i] = *(int *) (0x4) - PARAM_B; }

```

Table 2. Upper bounds on the leakage of the signature generation

Attacker model	<i>acc</i>	<i>accd</i>	<i>trace</i>	<i>time</i>
Leakage in bit	12.9	2.6	51.6	9.5

preserves the validity of analysis results because it cannot decrease the number of possible attacker observations.

Listing 4.2 shows our adaptation of a loop in `sample_y`. Again, the loop counter advances only if the random number generated in the current iteration satisfies certain criteria. To allow a meaningful analysis with CacheAudit 0.2c, we remove the check of the random number and assign an uninitialized value that overapproximates the possible range to each entry in `mat_y`. With this adaptation, our analysis uses a safe overapproximation of the values that `sample_y` can return, but assumes that `sample_y` itself, i.e., the random number generator, does not have any cache-side-channel leakage.

The function `crypto_sign` contains a potentially infinite while loop, on which CacheAudit 0.2b does not terminate within reasonable time. To make the analysis of this loop feasible, we fix the number of iterations while keeping the source code in the loop body unchanged. More concretely, we fixed the number of iterations to two to account for the effect of more than one iteration.

Note that we use the modifications described in this section only for the initial automatic analysis of ring-TESLA. In the detailed manual inspection in Sect. 5, we use the unmodified integer implementation.

Analysis Results. We obtain the leakage bounds listed in Table 2.⁶ The bounds lie between 2.6 bit and 51.6 bit for the different attacker models. One run of the adapted ring-TESLA leaks at most 2.6 bit to attackers under *accd*, at most 9.5 bit to attackers under *time*, at most 12.9 bit to attackers under *acc*, and at most 51.6 bit to attackers under *trace*. In the remainder of this article we investigate whether these non-zero leakage bounds are substantiated by concrete threats and how the leakage bounds can be reduced by mitigating concrete threats.

⁶ Throughout the article, we round bounds up to one decimal place and truncate the bounds to $3 * \text{PARAM_N} * 32\text{bit} = 49152\text{bit}$, i.e., the maximum size of the key.

5 Manual Analysis of the Potential Leakage

We manually analyze the signature generation `crypto_sign` to check if the potential leakage detected by program analysis corresponds to an actual concern. We identify substantiated threats of leakage to cache side channels (CSCs) in the routines `generate_c`, `test_w`, `test_rejection`, and `computeEc`.⁷ The following variables have to be kept secret during the execution of `crypto_sign`: the secret key `sk`, the randomness `vec_y`, the polynomials `vec_v1` and `vec_v2` to compute the hash value, and the polynomials `E1c`, `E2c`, and `Sc`. Furthermore, the hash value `c` and the representation of the corresponding encoded polynomial `pos_list` have to be kept secret until line 19 in Listing 3.1. In line 19, it is decided whether the potential signature (computed in line 18) and `c` are returned and, hence, whether `c` and `pos_list` become public information (via the encoding function F , the values can be computed from each other), or whether all computed values are discarded. An attacker should not learn the values of the discarded polynomials, e.g., `c` or `pos_list`. If the attacker learns values of `pos_list` or `c`, there exists a potential attack as described below.

Analysis of the subroutine `compute_Ec`. The implementation of the subroutine `compute_Ec` is given in Listing 5.1. The values that have to be kept secret during this computation are `sk`, `e`, `pos_list`, and `pos`. Most loops and branchings do not depend on any of the secret values. However, there might be a possible leakage of `pos` (and hence of the values in the secret `pos_list`) because of the cache hits/misses depending on `e`. In both loop bodies values are read from `e` (namely, either `e[j+PARAM_N-pos]` or `e[j-pos]`) such that in both loops together all entries of `e` are read. However, leakage arises from the chronological order of cache hits and misses. We illustrate the leakage using an example: The array `e` consists of `PARAM_N` many entries of type `int`, i.e., each entry of `e` is represented in 32 bit. Since one cache line is 64Byte (cf. Section 4), 16 entries (depending on the alignment in the memory) of `e` fit into one cache line. Let `pos=14`. Then, under the trace-driven attacker model, an attacker sees one cache miss (on element `e[PARAM_N-14]`) and 13 cache hits (on elements `e[PARAM_N-13]`, ..., `e[PARAM_N-1]`) during the loop in line 9.⁸ However, two more entries of `e` are also already loaded in the cache, namely `e[PARAM_N-16]` and `e[PARAM_N-15]`. Thus, in the second loop in line 10, the attacker sees cache hits on those two elements. He might, hence, be able to determine the value of `pos` from the distribution of the hits for the considered cache line over the loops.

Analysis of the subroutine `test_rejection`. The implementation of the subroutine `test_rejection` is given in Listing 5.2. The variable `poly_z` in Listing 5.2 has to be kept secret. The subroutine `test_rejection` consists of a for-loop that loops independently of the secret over `i=0, ..., PARAM_N`. Within the for-loop, there is a secret-dependent if-condition. This leads to a CSC vulnerability.

⁷ The analysis of the other subroutines of `crypto_sign` can be found in the corresponding technical report under <http://eprint.iacr.org/2017/951>.

⁸ To simplify our explanation we assume that the corresponding cache line starts with `e[PARAM_N-16]` and ends with `e[PARAM_N-1]`.

Listing 5.1. Code of the subroutine `computeEc`

```

1  static void computeEc(poly Ec, const unsigned char *sk, const
2  uint32_t pos_list[PARAM_W]) {
3      int i,j, pos, * e;
4      e = (int*)sk;
5      for(i=0;i<PARAM_N;i++){ Ec[i] = 0;}
6
7      for(i=0;i<PARAM_W;i++){
8          pos = pos_list[i];
9          for(j=0;j<pos;j++){ Ec[j] += e[j+PARAM_N - pos];}
10         for(j=pos;j<PARAM_N;j++){ Ec[j] -= e[j-pos];} } }

```

Listing 5.2. Code of the subroutine `test_rejection`

```

1  static int test_rejection(poly poly_z) {
2      int i;
3      for(i=0; i<PARAM_N; i++){
4          if(poly_z[i]<-(PARAM_B-PARAM_U)||poly_z[i]>(PARAM_B-PARAM_U)){return 1;}}
5      return 0; }

```

Assume a strong trace-driven attacker model, i.e., the attacker has a sequence of occurred cache hits and misses. Assume furthermore that `poly_z` is already loaded in the cache before the if-condition is evaluated.⁹ When the if-condition in line 4, Listing 5.2, is never true, then the value 0 is returned and the attacker gets a sequence of `PARAM_N` (or $2 \cdot \text{PARAM}_N$ — depending on the compilation) hits. This essentially means that all coefficients of `poly_z` are in the interval $[-B + U, B - U]$ and, hence, the corresponding signature is compressed and returned (cf. Listing 3.1). Next, we consider the other case, i.e., the absolute value of at least one of the coefficients of `poly_z` is larger than $B - U$. That means that the if-condition in Listing 5.2 holds true for some $i \in \{0, \dots, \text{PARAM}_N\}$. Hence, 1 is returned in the i -th iteration and the attacker gets a sequence of only `PARAM_N-i` hits. Hence, the attacker knows the exact index of the coefficient that violated the if-condition. Assume the attacker also knows the values in the array `pos_list` (which corresponds to the polynomial $c = F(H([v_1]_M, [v_2]_M, \mu))$ in Fig. 1) from another cache-side-channel vulnerability. Then the attacker might know which coefficients of the secret s contributed to the i -th, large coefficient of `poly_z`. If an attacker learns the exact position i and the corresponding `pos_list` for many different values to the same secret key s then the attacker might receive enough information about the size of the entries in s to successfully break the scheme via a learning-the-parallelepiped-attack [30,31].

Analysis of the subroutine `test_w`. The implementation of the routine `test_w` is given in Listing 5.3. The values `poly_w`, `val`, and `left` in Listing 5.3 have to be kept secret. The CSC vulnerability is similar to the channel described previously for `test_rejection`. In the subroutine `test_w`, the CSC vulnerability comes

⁹ Our arguments hold also true if we assume that `poly_z` is not loaded in the cache. In the ring-TESLA implementation, it is already loaded in line 18 in Listing 3.1.

Listing 5.3. Code of the subroutine `test_w`

```

1  static int test_w(poly poly_w)
2  { int i; int64_t left, right, val;
3    for(i=0; i<PARAM_N; i++){
4      val = (int64_t) poly_w[i];
5      val = val % PARAM_Q;
6      if (val < 0){ val = val + PARAM_Q;}
7      left = val;
8      left = left % (1<<(PARAM_D));
9      left -= (1<<PARAM_D)/2;
10     left++;
11     right = (1<<(PARAM_D-1))-PARAM_REJECTION;
12     if (abs(left) > right){ return -1; } }
13     return 0; }

```

from the early abortion depending on `left` in line 12 of Listing 5.3. When the if-condition in line 12 holds for some `i` and the corresponding `abs(left)`, `-1` is immediately returned and a trace-driven attacker might learn the exact index `i`.

Analysis of the subroutine `generate_c`. The implementation of the subroutine `generate_c` is given in Listing 4.1. The values `pos_list`, `c`, and `pos` in Listing 4.1 have to be kept secret. There are no branchings or loops depending on the secret value, except for one if-condition on `c[pos]` in line 12 of Listing 4.1. If a cache with no-write-allocate policy is used, the values `c[i]` are not cached in line 5. Hence, an attacker might be able to find out which elements `c[i]` are cached in line 12 and to learn information about the values of `pos`. Together with the vulnerability in `test_rejection`, an attacker might be able to successfully break the scheme via a learning-the-parallelepiped-attack [30, 31].

Combined analysis of the overall signature generation. The most important parts of the implementation of `crypto_sign` are depicted in Listing 3.1. In the signature generation, most operations, branchings, or loops are independent of the *value* of the secret. Exceptions are the branchings in lines 11, 15, and 19 in Listing 3.1: They depend on secret values and, hence, the length of the observed trace of cache hits and misses depends on the branches that are taken. What does this mean from a cryptographic viewpoint? Assuming the subroutine `test_w` does not leak any bit, then the attacker does not learn more information about the secret if he knows whether or not the condition in line 11 holds. The attacker would just learn that `vec_v1` does not fulfill the conditions needed for a valid signature. However, the attacker does not learn why exactly the condition was not fulfilled (the attacker does not learn the index on which the if-condition failed). Furthermore, since the value `vec_v1` depends on `vec_y` and the value `vec_y` is discarded if the if-condition in line 11 does not hold, the attacker does not get any additional information about the secret he did not know before. The same explanation also holds for the branchings in line 15 and line 19.

In summary, this means that there exists a potential leakage that we probably cannot get rid of, but it does not affect the security of the signature scheme as long as we do not have leakages in `test_w` and `test_rejection` or `generate_c`.

6 Mitigation of the Vulnerabilities

6.1 Adaptation of Vulnerable Routines

In Sect. 5, we identified substantiated threats of CSC leakage in the routines `test_w`, `test_rejection`, `computeEc`, and `generate_c`. Since the leakage in `generate_c` is only a concern in combination with the leakage in `test_w` and `test_rejection`, it suffices to analyze and mitigate the leakage in `test_w`, `test_rejection`, and `computeEc`. We analyze `test_w`, `test_rejection`, and `computeEc` individually with CacheAudit 0.2c to obtain leakage bounds on the unmitigated implementations. The leakage bounds are listed in Table 3. There are, indeed, non-zero bounds for all three unmitigated routines.

Table 3. Leakage bounds [bit]

	Unmitigated routines				Mitigated routines			
	<i>acc</i>	<i>accd</i>	<i>trace</i>	<i>time</i>	<i>acc</i>	<i>accd</i>	<i>trace</i>	<i>time</i>
<code>test_w</code>	31	31	49152	19.3	0	0	0	0
<code>test_rejection</code>	31	31	10.1	10.1	0	0	0	0
<code>computeEc</code>	0	0	20	5.9	0	0	19	4.4
<code>crypto_sign</code>	12.9	2.6	51.6	9.5	8.1	1.6	48.6	9.0

```

1  int test_rejection(poly poly_z) {
2  int i; int res; res = 0;
3  for(i=0; i<PARAM_N; i++){
4  res |= (poly_z[i] < -(PARAM_B-PARAM_U));
5  res |= (poly_z[i] > (PARAM_B-PARAM_U)); }
6  return res; }
7
8  int test_w(poly poly_w) { [...]
9  for(i=0; i<PARAM_N; i++) {
10 val = poly_w[i]; val = val % PARAM_Q;
11 val += (((unsigned int)val & 0x80000000) >> 31)*PARAM_Q;
12 left = val; left = left % (1<<(PARAM_D));
13 left -= (1<<(PARAM_D)/2); left++;
14 right = (1<<(PARAM_D-1))-PARAM_REJECTION;
15 res |= (abs(left) - right > 0); }
16 return -res; }
17
18 void computeEc([...]) { [...]
19 for(i=0; i<PARAM_N; i++) Ec[i] = 0;
20 for(i=0; i<PARAM_N; i++) tmp = e[i];
21 for(i=0; i<PARAM_W; i++) {
22 pos = pos_list[i];
23 for(j=0; j<pos; j++) { Ec[j] += e[j+PARAM_N - pos]; }
24 for(j=pos; j<PARAM_N; j++) { Ec[j] -= e[j-pos]; } } }

```

We adapt the routines, as shown in the above listing, to mitigate the leakage. In `test_rejection`, we collect the result, i.e., whether 0 or 1 is returned, in an auxiliary variable `res` and return it after `PARAM_N` iterations, instead of returning early in case of a failed test. In `test_w`, we also collect the result, i.e., whether 0 or 1 is returned, in an auxiliary variable `res`, instead of returning early in case of failure. Furthermore, we replace the branching on `val` by an assignment that masks the value by the branching condition. The idea to mask assignments by branching conditions comes from conditional assignment [32] - a program transformation to mitigate timing side channels, which performs rather well in practical evaluation [33]. In `computeEc`, we add preloading of the variable `e` to ensure that the sequence of cache hits and misses does not depend on the secret-dependent order of accesses (under the assumption that no process interferes with the cache during the ring-TESLA execution).

By code inspection, the modifications should remove the CSC leakage in the three routines. In the following, we investigate this with CacheAudit.

6.2 Analysis of the Effectiveness of the Mitigations

We analyze the mitigated routines `test_w`, `test_rejection`, and `computeEc` using CacheAudit 0.2c and obtain the leakage bounds listed in Table 3. For `test_w` and `test_rejection`, we obtain the leakage bound 0 bit for all attacker models. Thus, we effectively removed the potential leakage. For `computeEc`, we obtain 0 bit leakage bounds for `acc` and `accd` and non-zero leakage bounds, namely 19 bit and 4.4 bit, for `trace` and `time`, respectively. The bounds computed with CacheAudit are provable upper bounds, but not necessarily tight. The preloading of `e` should remove the leakage from `computeEc`, because it makes the caching of `e` independent of secrets. Since CacheAudit was able to recognize preloading as effective in other cases [16, 17], it is interesting why CacheAudit 0.2c does not yield a 0 bit leakage bound in this case. The investigation and fine-tuning of the analysis precision is an interesting direction for future work.

Table 3 lists also the leakage bounds we obtain on `crypto_sign` with and without our countermeasures. All four leakage bounds are reduced by our countermeasures. The highest reduction is achieved for the `acc` leakage bound. The `acc` leakage bound is reduced by 4.8 bit to 8.1 bit.

Based on our manual inspection of the individual routines in ring-TESLA, there are two possible sources for the remaining potential leakage reported by CacheAudit 0.2c. One source is `generate_c`, where, as discussed above, the remaining leakage is harmless. The second source is the rejection sampling in `crypto_sign`. This matches the fact that the leakage bounds are non-zero. Note that, since we compute upper bounds on the leakage based on overapproximation, the actual leakage of the implementation could be even lower than the reported leakage bounds. We expect in particular the leakage bounds for `trace` and `time` to be quite conservative because CacheAudit 0.2c was not able to recognize the preloading countermeasure in the implementation of `computeEc` and the results of `computeEc` are propagated further through the implementation. Nev-

ertheless, the bounds show that the CSC leakage of ring-TESLA to *acc*, *accd*, *trace*, and *time* is rather low. For *acc*, *accd*, and *time*, it even lies below 10 bit.

Note that, the leakage bounds refer to information about the secret key, i.e., about the three polynomials that all together are saved in roughly 38,400 bit. By construction of the learning with errors problem (LWE)—the underlying hardness assumption of ring-TESLA—the potential leakage of at most 49 bit of the secret key does not immediately translate to the bit-hardness of LWE (resp., the bit-security of ring-TESLA).

7 Conclusion

In this article, we analyzed an implementation of the lattice-based signature scheme ring-TESLA for cache-side-channel vulnerabilities. We identified four routines in the implementation that are vulnerable through cache side channels. Two of these routines, a rejection sampling and a signature validity check, use a secret-dependent number of iterations. One routine is a sparse polynomial multiplication that traverses one polynomial in a secret-dependent order. We modified these functions to ensure a constant number of iterations and secret-independent caching of the polynomial. By modifying these functions, we also eliminated the possibility to exploit the fourth vulnerability. For the modified ring-TESLA implementation, we obtained low upper bounds on the leakage to four attacker models, using program analysis.

Our results show that implementations of rejection sampling and sparse multiplication should be inspected for side channels with particular care. While these techniques are not very common in classical cryptography like RSA, they play a significant role in post-quantum cryptography. Rejection sampling occurs, for instance, in multiple lattice-based signature schemes [11, 15, 19] and in key exchange protocols [34]. Sparse multiplication also occurs in many lattice-based schemes [11, 19, 35]. Overall, the implementation and analysis of post-quantum cryptography poses additional challenges compared to classical cryptography.

Acknowledgements. We thank the anonymous reviewers for their helpful suggestions and Sedat Akleylek for contributing to our modifications of the original ring-TESLA implementation. This work has been partially funded by the DFG as part of projects P1 and E3 within the CRC 1119 CROSSING.

References

1. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**(5), 1484–1509 (1997)
2. National Institute of Standards and Technology (NIST): Post-quantum project (2016). https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf. Accessed 23 May 2017
3. Yarom, Y., Genkin, D., Heninger, N.: CacheBleed: a timing attack on OpenSSL constant time RSA. In: Gierlichs, B., Poschmann, A.Y. (eds.) CHES 2016. LNCS, vol. 9813, pp. 346–367. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53140-2_17

4. Aciğmez, O., Koç, Ç.K.: Trace-driven cache attacks on AES. *Cryptology ePrint Archive*, Report 2006/138
5. Osvik, D.A., Shamir, A., Tromer, E.: Cache attacks and countermeasures: the case of AES. In: Pointcheval, D. (ed.) *CT-RSA 2006*. LNCS, vol. 3860, pp. 1–20. Springer, Heidelberg (2006). https://doi.org/10.1007/11605805_1
6. Bernstein, D.J.: Cache-timing attacks on AES. Technical report, University of Illinois at Chicago (2005)
7. Oder, T., Schneider, T., Pöppelmann, T., Güneysu, T.: Practical CCA2-secure and masked ring-LWE implementation. *Cryptology ePrint Archive*, Report 2016/1109 (2016)
8. Pessl, P.: Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In: Dunkelman, O., Sanadhya, S.K. (eds.) *INDOCRYPT 2016*. LNCS, vol. 10095, pp. 153–170. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49890-4_9
9. Saarinen, M.J.O.: Arithmetic coding and blinding countermeasures for lattice signatures. *Cryptology ePrint Archive*, Report 2016/276 (2016)
10. Reparaz, O., Roy, S.S., Vercauteren, F., Verbauwhede, I.: A masked ring-LWE implementation. *Cryptology ePrint Archive*, Report 2015/724 (2015)
11. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. In: Canetti, R., Garay, J.A. (eds.) *CRYPTO 2013*. LNCS, vol. 8042, pp. 40–56. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_3
12. Groot Bruinderink, L., Hülsing, A., Lange, T., Yarom, Y.: Flush, gauss, and reload - a cache attack on the BLISS lattice-based signature scheme. In: *CHES*, pp. 323–345 (2016)
13. Akleyek, S., Bindel, N., Buchmann, J., Krämer, J., Marson, G.A.: An efficient lattice-based signature scheme with provably secure instantiation. In: Pointcheval, D., Nitaj, A., Rachidi, T. (eds.) *AFRICACRYPT 2016*. LNCS, vol. 9646, pp. 44–60. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31517-1_3
14. Alkim, E., Bindel, N., Buchmann, J., Dagdelen, Ö., Eaton, E., Gutoski, G., Krämer, J., Pawlega, F.: Revisiting TESLA in the quantum random oracle model. In: *PQCrypto*, pp. 143–162 (2017)
15. Bai, S., Galbraith, S.D.: An improved compression technique for signatures based on learning with errors. In: Benaloh, J. (ed.) *CT-RSA 2014*. LNCS, vol. 8366, pp. 28–47. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04852-9_2
16. Doychev, G., Köpf, B., Mauborgne, L., Reineke, J.: CacheAudit: a tool for the static analysis of cache side channels. *ACM TISSEC* **18**(1), 4:1–4:32 (2015)
17. Mantel, H., Weber, A., Köpf, B.: A systematic study of cache side channels across AES implementations. In: *ESSoS*, pp. 213–230 (2017)
18. Doychev, G., Köpf, B.: Rigorous analysis of software countermeasures against cache attacks. In: *PLDI*, pp. 406–421 (2017)
19. Barreto, P.S.L.M., Longa, P., Naehrig, M., Ricardini, J.E., Zanon, G.: Sharper ring-LWE signatures. *Cryptology ePrint Archive*, Report 2016/1026 (2016)
20. Abel, A., Reineke, J.: Reverse engineering of cache replacement policies in Intel microprocessors and their evaluation. In: *ISPASS*, pp. 141–142 (2014)
21. Smith, G.: On the foundations of quantitative information flow. In: de Alfaro, L. (ed.) *FoSSaCS 2009*. LNCS, vol. 5504, pp. 288–302. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00596-1_21
22. Shannon, C.E.: A mathematical theory of communication. *Bell Syst. Tech. J.* **27**(3), 379–423, 623–656 (1948)

23. Köpf, B., Smith, G.: Vulnerability bounds and leakage resilience of blinded cryptography under timing attacks. In: CSF, pp. 44–56 (2010)
24. Alvim, M.S., Chatzikokolakis, K., Palamidessi, C., Smith, G.: Measuring information leakage using generalized gain functions. In: CSF, pp. 265–279 (2012)
25. Yarom, Y., Falkner, K.: FLUSH+RELOAD: a high resolution, low noise, L3 cache side-channel attack. In: USENIX Security, pp. 719–732 (2014)
26. Cousot, P., Cousot, R.: Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: POPL, pp. 238–252 (1977)
27. Doychev, G.: Commit f063813faa548da9bfb11dea9ff6fe39c0f11626: adding support for CDQ and NEG instructions. <https://github.com/cacheaudit/cacheaudit/commit/f063813faa548da9bfb11dea9ff6fe39c0f11626> (2016). Accessed 23 May 2017
28. Degenbaev, U.: Formal specification of the x86 instruction set architecture. Ph.D. thesis, Universität des Saarlandes (2012)
29. Intel Corporation: Intel[®] 64 and IA-32 Architectures Optimization Reference Manual. Order Number: 248966–032 (2016)
30. Ducas, L., Nguyen, P.Q.: Learning a zonotope and more: cryptanalysis of NTRU_{sign} countermeasures. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 433–450. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_27
31. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: cryptanalysis of GGH and NTRU signatures. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 271–288. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_17
32. Molnar, D., Piotrowski, M., Schultz, D., Wagner, D.: The program counter security model: automatic detection and removal of control-flow side channel attacks. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 156–168. Springer, Heidelberg (2006). https://doi.org/10.1007/11734727_14
33. Mantel, H., Starostin, A.: Transforming out timing leaks, more or less. In: Pernul, G., Ryan, P.Y.A., Weippl, E. (eds.) ESORICS 2015. LNCS, vol. 9326, pp. 447–467. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24174-6_23
34. Zhang, J., Zhang, Z., Ding, J., Snook, M., Dagdelen, Ö.: Authenticated key exchange from ideal lattices. In: EUROCRYPT, pp. 719–751 (2015)
35. Buchmann, J., Göpfert, F., Güneysu, T., Oder, T., Pöppelmann, T.: High-performance and lightweight lattice-based public-key encryption. In: IoTPTS, pp. 2–9 (2016)



Extinguishing Ransomware - A Hybrid Approach to Android Ransomware Detection

Alberto Ferrante¹, Miroslaw Malek¹, Fabio Martinelli²,
Francesco Mercaldo²(✉), and Jelena Milosevic¹

¹ Faculty of Informatics, Advanced Learning and Research Institute,
Università della Svizzera italiana, Lugano, Switzerland

² Institute for Informatics and Telematics,
National Research Council of Italy (CNR), Pisa, Italy
francesco.mercaldo@iit.cnr.it

Abstract. Mobile ransomware is on the rise and effective defense from it is of utmost importance to guarantee security of mobile users' data. Current solutions provided by antimalware vendors are signature-based and thus ineffective in removing ransomware and restoring the infected devices and files. Also, current state-of-the-art literature offers very few solutions to effectively detecting and blocking mobile ransomware. Starting from these considerations, we propose a hybrid method able to effectively counter ransomware. The proposed method first examines applications to be used on a device prior to their installation (static approach) and then observes their behavior at runtime and identifies if the system is under attack (dynamic approach). To detect ransomware, the static detection method uses the frequency of opcodes while the dynamic detection method considers CPU usage, memory usage, network usage and system call statistics. We evaluate the performance of our hybrid detection method on a dataset that contains both ransomware and legitimate applications. Additionally, we evaluate the performance of the static and dynamic stand-alone methods for comparison. Our results show that although both static and dynamic detection methods perform well in detecting ransomware, their combination in a form of a hybrid method performs best, being able to detect ransomware with 100% precision and having a false positive rate of less than 4%.

Keywords: Ransomware · Malware · Hybrid detection
Machine learning · Android · Security

1 Introduction

Ransomware is such a relevant security problem that law enforcement agencies from all around Europe teamed up with antimalware and other IT security companies to form the *No More Ransom!* project¹. According to [8], 2016 might be

¹ <https://www.nomoreransom.org>.

remembered as “the year of ransomware,” confirming the predictions of exponential growth of these kinds of attacks from previous years [7]. Recent events, such as the WannaCry ransomware attack in May 2017, in which over 200,000 computers in more than 150 countries were rendered unusable with ransom demands², demonstrated that these predictions might be beaten in 2017. Ransomware not only targets PCs and servers but also mobile devices. In particular, in the realm of mobile devices, as opposed to other threats that silently try to get into possession of data by copying them and leaking through the network, ransomware denies access to data by encryption or simply by locking the device and scaring users (i.e., users are made believe that data are encrypted even if they are not). In some cases even paying the requested ransom does not guarantee that the access to data is restored. Having in mind that data are one of users’ most valuable assets, a mechanism that can detect ransomware at installation time of applications or during their execution is highly desirable. However, although mobile ransomware threats are on the rise, we have found only a small number of related works in literature addressing the problem of mobile ransomware detection.

We propose to use a hybrid detection method that is composed of a static method, to be used when applications are installed and/or updated, and a dynamic method, to be used at runtime. Static methods detect ransomware by considering features that can be obtained without running the applications (e.g., frequency of op-codes). Dynamic methods, instead, are based on features that can only be observed at runtime and that represent the behavior of applications (memory, CPU, network and statistics on system calls). Static approaches are less computationally intense than dynamic methods as they do not need applications to be run for identifying malware [10], but they are typically ineffective with obfuscated code as well as with run-time infections. On the other hand, dynamic methods are effective in identifying new threats, outperforming static methods, but they need applications to be run to identify malicious behaviour, potentially infecting the device [9]. In addition dynamic methods are able to discriminate malware even when its code is obfuscated [15]. The main idea behind using a hybrid approach is to have the advantages of both static and dynamic methods while reducing or masking their disadvantages.

Having this in mind, the main contributions of this paper are as follows:

1. Evaluation of the effectiveness of a static approach, based on the frequency of op-codes, in detection of mobile ransomware (Sect. 4.1).
2. Evaluation of the effectiveness of a dynamic approach, based on the monitoring of memory, CPU, network and statistics on system calls, in detection of mobile ransomware (Sect. 4.2).
3. Evaluation of the effectiveness of a hybrid, combined static and dynamic, approach in detection of mobile ransomware (Sect. 4.3).

² http://wapo.st/2pKyXum?tid=ss.tw&utm_term=.6887a06778fa.

2 Hybrid Detection Method

Hybrid detection employs both static and dynamic detection. Figure 1 depicts the high-level workflow of the proposed approach: we first use a static detection method when applications are installed, updated, or during periodic checks (e.g., every week). Applications identified as malware are denied running on the device. All the other applications are instead allowed to run, but subject to dynamic detection while they are executed. In this way we are able to complement the coverage of static detection with the one of dynamic detection.

Next, we describe the two methods, static and dynamic, that compose our hybrid method. Additionally, we discuss the steps required to develop and use them – pre-processing, learning, and classification – as well as the system features, both static and dynamic, that are considered.

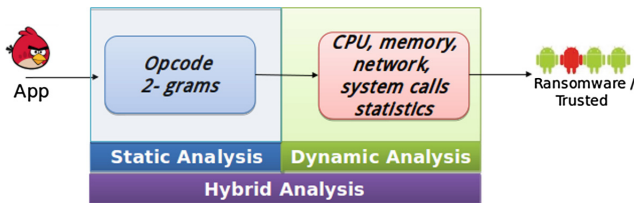


Fig. 1. High-level workflow of the proposed approach. The static analysis consists of a classification based on features obtained from the executable of the application under analysis, while the dynamic one is based on a feature set obtained when the application is running.

2.1 Static Analysis

Structural code analysis has been identified by the research community as effective and highly accurate in static detection methods. For instance, in [3, 4], the opcode occurrences are considered as main features for detecting malware, with precision equal to 0.9. Additionally, the method proposed in [2] demonstrated that the sequences of opcodes are very effective in detecting Android malware, providing an accuracy of 96.88%. In this approach, the authors considered a binary classification problem in which an input application a has to be classified as malware or trusted using the occurrences of two opcodes as features (i.e., 2-grams). n -grams with $n = 2$ demonstrated to provide the best possible performance in identifying Android malware in comparison to other values of n [3, 4, 12].

Other approaches, such as the ones based on features derived from the permissions required by the applications and/or system call occurrences provide good detection performance, but not as good as the previously described methods.

Thus, we adopt an approach similar to [2] for detecting ransomware. In our approach each application is pre-processed in order to obtain the numeric values of frequencies of opcode sequences that are suitable to be processed by the classifier. After pre-processing, the classifier undergoes the learning phase in which

it is trained by using a labelled dataset. After the learning phase, the classifier can be used for the actual classification of the applications as ransomware or trusted.

Let a be the Android application under analysis. In the pre-processing phase, we first use the `apktool`³ tool in order to extract the Smali classes of the application under analysis, thus being able to obtain a unique file containing the full set of opcodes (without the relative argument and parameters) of all the a application classes.

We then compute the frequency of 2-grams opcodes as follows: let \mathcal{O} be the set of possible opcodes, and let $\mathcal{O} = \bigcup_{i=1}^{i=n} \mathcal{O}^i$ the set of n -grams (i.e., sequences of opcodes whose length is up to n , we consider $n = 2$). We denote with $f(a, o)$ the frequency of the n -gram $o \in \mathcal{O}$ in the application a : $f(a, o)$ is hence the number of occurrences of o divided by the total length of the opcode sequences in a . Finally, we set the *feature vector* $\mathbf{f}(a) \in [0, 1]^{|\mathcal{O}|}$ corresponding to a to $\mathbf{f}(a) = (f(a, o_1), f(a, o_2), \dots)$ with $o_i \in \mathcal{O}$. Additionally, we split the application code into chunks corresponding to class methods.

In the next phase, called learning, we train a binary classifier C from two sets A_M, A_T of ransom and trusted applications (the *learning sets*), respectively. The learning phase is divided into a feature selection phase and the actual classifier training phase. The aim of the feature selection phase is twofold: on the one hand, we want to reduce the dimension of the input since with $n = 2$ the size $|\mathcal{O}|$ of each feature vector \mathbf{f} can be up to $\approx 10^{12}$. On the other hand, we want to retain only the more informative n -grams, with respect to the output label.

At first, the average frequencies $\bar{f}_M(o)$ and $\bar{f}_T(o)$ are computed for each 2-gram $o \in \mathcal{O}$ on the ransomware and trusted samples. Then, the relative difference $d(o)$ between the two average values is computed. This relative difference is high if the 2-gram o is frequent among ransomware applications and infrequent among trusted applications (and vice versa). Then, we build the set $\mathcal{O}' \subset \mathcal{O}$ of n -grams composed of the h n -grams with the highest values of $d(o)$, where h is a parameter of our method. We do not include in \mathcal{O}' the n -grams for which $d(o) = 1$ (i.e., we purposely do not consider those 2-grams which occur in only one subset of applications in the learning set): this way, we strive to avoid building a classifier which works well on seen applications but fails to generalize. Finally, we retain in \mathcal{O}' only the remaining 2-grams with the greatest value. Accordingly, we set the *reduced feature vector* $\mathbf{f}'(a)$ corresponding to a using only the frequencies of the 2-grams in \mathcal{O}' , i.e., $\mathbf{f}'(a) = (f(a, o_1), f(a, o_2), \dots)$ with $o_i \in \mathcal{O}'$. We consider as *reduced feature vector* the 50 2-grams with the greatest value, i.e. the most frequent 2-grams. In fact, we have determined experimentally that performance decays when than 50 2-grams are considered. The second step of the learning phase consists of training the actual classifier C using the reduced feature vectors obtained from the applications in the learning sets and the corresponding labels. In this work, we experimented with the J48, NaiveBayes, and Logistic Regression classification algorithms.

³ <https://ibotpeaches.github.io/Apktool>.

In the last phase, named classification, we determine if an application a is labelled as ransomware or trusted, according to the learned classifier C . To this end, we pre-process a as previously explained in order to obtain the reduced feature vector $\mathbf{f}'(a)$. Then, we input $\mathbf{f}'(a)$ to C and classify a into {ransomware, trusted}. In a real system, this step is run every time a new application is installed or updated. Additionally, it can be run periodically to check all the applications installed on the mobile device.

2.2 Dynamic Analysis

In order to perform dynamic detection of ransomware, an effective method based on the observation of system behavior has to be used. Since this method needs to be used at runtime on mobile devices, it also needs to be lightweight on computational and energy resources. For this purpose, as the most suitable candidate we identified MalAware, the approach proposed in [13], that uses only six memory and one CPU related features in order to perform on-device detection at runtime, and that is based on a two-step detection system of low complexity that first classifies execution records, and then complete applications by relying on the past classifications of execution records. Thus, we use a similar two-step detection approach, but in addition to memory and CPU usage we also consider features representing network usage and statistics on system calls. Similarly to static detection, the development of the dynamic detection method undergoes the two phases of pre-processing and learning, with the classification phase used at runtime to actually detect malware.

The pre-processing phase is necessary to extract features from the execution logs of the considered applications. As explained in Sect. 3, execution logs are obtained by running the considered applications in an instrumented environment.

In the learning phase, we first train a classifier to recognize execution records associated with malicious behavior, similarly to the work presented in [14]. The training of classifiers is performed by using the same training set used for the development of static detection as well as the same classifiers of low complexity, suitable for on device, runtime detection: Naive Bayes, Decision Trees (J48), and Logistic Regression. The results of classification are then used as input for a sliding-window based mechanism that considers the history of past execution records to classify the applications. Namely, considering a sliding window of length n , the percentage of records classified as ransomware in the last n instants of time is used to determine whether an application is ransomware or not. To make the mechanism more robust, multiple results, obtained in disjointed sliding windows, are considered: when w windows are marked as ransomware, the application is classified as ransomware. In the training phase we also choose the most suitable parameters for the sliding window mechanism, namely window size, threshold, and number of checks. The choice is done by exploring different combinations of parameters and by studying the obtained results as discussed in Sect. 4.

In the classification phase, the application classifier obtained in the training phase is used at runtime to detect ransomware. The full detection system is composed of three main parts: a feature monitoring block, the record-level classifier, and the applications-level classifier. The first block monitors and extracts features periodically and sends them to the record-level classifier which, in turn, sends its classification results to the application-level classifier. This last classifier is the one that can mark an application as ransomware and raise an alarm.

3 Experimental Setup

In this section, we first describe our dataset, followed by the setup of the experimental environment and the description of collected features.

3.1 Dataset

We have based our experiments on a dataset containing 3,058 mobile applications: 2,386 Android trusted applications downloaded from the Google Play Store⁴ and 672 applications containing ransomware taken from the freely available HelDroid dataset⁵. These ransomware samples appeared from December 2014 to June 2015. Following, we list the malware families to which these samples belong [1]:

- Ransomware applications in the *Locker* family block the screen of infected devices and request a ransom for unlocking it; no file is actually encrypted.
- The *Koler* payload is downloaded by exploiting site redirection; the screen is then occupied by the ransom browser page that cannot be dismissed if not for very short periods of time.
- Ransomware in the *Svpeug* family is based on an overlay attack: legitimate applications launched by the user are overlaid with fake windows imitating the legitimate applications and thus fooling the victim. Additionally, users receive a message, pretending to be sent by FBI and claiming that the device has been locked due to access to child pornography websites. To unlock the phone, a ransom needs to be paid.
- Samples in the *ScarePackage* ransomware family masquerade as well-known applications, such as Adobe Flash or antimalware applications, and, when launched, they pretend to scan your phone. After completing the fake scan, the device is locked and after a reboot a fake FBI message is shown. A ransom is requested to bring back the device to normal.
- Ransomware applications in the *SimpleLocker* family scan the SD card for images, documents and videos and encrypt them by using the AES encryption algorithm; a message notifying the user and asking for a ransom is shown on the display. This is the only family in our dataset that actually encrypts data on the device and it was the first one discovered for Android.

⁴ <https://play.google.com/store>.

⁵ <http://ransom.mobi>.

In order to download trusted applications we crawled the Google Play Store by using the *Python Android Market Library* open-source crawler⁶. The crawler is configured to download trusted applications equally distributed in all the different categories of the market.

All the applications were checked by means of VirusTotal⁷, a service that runs 57 different antimalware on the submitted applications. The analysis confirmed that our trusted samples did not contain any known malicious payload, while the malicious samples contained ransomware specific payloads.

For developing and validating the proposed approach, we have separated the collected applications into a training and a test dataset. The training dataset contains two thirds of the available trusted applications and two thirds of the available application for each of the ransomware family. The testing dataset contains the remaining applications.

3.2 Experimental Environment for Dynamic Analysis

Dynamic analysis relies on runtime observation of memory usage, CPU usage, network behavior, as well as statistics on system calls. Therefore, execution traces containing this information need to be collected by executing the applications in a controlled environment. These traces have been recorded by running the applications, one at a time, on the Android emulator. Monitoring scripts, with a monitoring interval of two seconds have been used. The procedure of executing the applications was automated by means of a Linux shell script, which has been run on a Linux PC and made use of Android Debug Bridge (adb)⁸, a command line tool that allows the PC to communicate with an emulator instance or with an Android device. To collect system calls we used *strace*⁹, a tool for tracing system calls. Network log files were collected by capturing the network traffic of the emulator. Network statistics have been obtained by logging all network traffic of the emulator and by successively running the *tcpstat* tool, set to sample the features at 2s intervals. Log files for CPU, memory, and network are later unified by using timestamps recorded at execution time.

For applying stimuli to applications, the Monkey application exerciser¹⁰ has been used in the script. It is a command-line tool that sends a pseudo-random stream of user events into the system, which acts as a stress test on the application software. One of the main problems of dynamic detection methods is in the execution of samples during the development phase. In fact, there is currently no method to verify automatically that malicious payloads are activated correctly. Due to the high number of samples that need to be used to obtain good quality classifiers, it is not even possible to perform this verification manually. This introduces some additional uncertainty in dynamic detection methods. In this

⁶ <https://github.com/liato/android-market-api-py>.

⁷ <https://www.virustotal.com/>.

⁸ <http://developer.android.com/tools/help/adb.html>.

⁹ <http://linux.die.net/man/1/strace>.

¹⁰ <http://developer.android.com/tools/help/monkey.html>.

work, we have tried to minimize the number of non-activated malicious payloads by providing a high number of stimuli (20,000, with a limit on execution time of 15 min) to each application (both malware and benign). It is our belief that the duration that we have chosen is a good tradeoff between time when most of the ransomware samples expose their malicious intentions and duration of the overall experimentation. The Android emulator of choice is the one included in the Android Software Development Kit¹¹ release 20140702, running Android 4.0, that was one of the most popular versions of Android in period from when ransomware samples originate. The reason why an Android emulator has been chosen instead of real devices is that this solution provides the ability to run a large number of applications, making the obtained dataset more significant. The Android operating system has been re-initialized each time before running each application, to avoid possible interferences (e.g., changed settings, running processes, and modifications of the operating system files) from previously run samples.

In total, 87 features could be extracted from the execution traces, all referred to single applications resource usage. All the features considered are listed in Table 1. Out of the considered features, 59 are related to different aspects of memory usage. Five are related to CPU: three to CPU usage, and two to virtual memory exceptions (major and minor faults), 15 are related to network traffic and five represent statistics on system calls.

4 Experimental Results

In this section, we report the results obtained when considering static detection alone, dynamic detection alone, and the hybrid method. In order to evaluate the detection performance of the proposed method, we report four metrics: precision, recall, F-Measure, and Receiver Operating Characteristics (ROC) curve. F-Measure represents a weighted average of precision and recall, while the ROC Area is defined as the probability that a randomly chosen positive instance is incorrectly classified as a negative instance; ROC Area is a suggested metric to represent detection performance when the considered datasets are unbalanced, as it is in our case, and in malware detection in general [5].

4.1 Static Detection

We first discuss the results obtained by using the static detection method described in Sect. 2.1. As mentioned in Sect. 2.1 we have experimented with three different classifiers, namely J48, Naive Bayes, and Logistic Regression as well as with n equal to 2. In feature selection, we use, as mentioned earlier, $h = 50$. After the learning phase, performed by using the training set described in Sect. 3, we applied the classifier C to each application of the test set and we measured precision, recall, F-Measure and ROC Area.

¹¹ <https://developer.android.com/sdk/index.htm>.

Table 1. List of all the considered features; totals are related only to single applications; unless differently specified, all numbers are related to the considered monitoring period.

Category		Feature names
CPU	CPU Usage	Total CPU Usage, User CPU Usage, Kernel CPU Usage
	Virtual Memory	Page Minor Faults, Page Major Faults
Memory	Native memory	Native Pss, Native Shared Dirty, Native Private Dirty, Native Heap Size, Native Heap Alloc, Native Heap Free
	Dalvik memory	Dalvik Pss, Dalvik Shared Dirty, Dalvik Private Dirty, Dalvik Heap Size, Dalvik Heap Alloc, Dalvik Heap Free, Cursor Pss
	Cursor memory	Cursor Shared Dirty, Cursor Private Dirty
	Android shared memory	Ashmem Pss, Ashmem Shared Dirty, Ashmem Private Dirty
	Memory-mapped native code	.so mmap Pss, .so mmap Shared Dirty, .so mmap Private Dirty
	Memory mapped Dalvik code	.dex mmap Pss, .dex mmap Shared Dirty, .dex mmap Private Dirty
	Memory-mapped fonts	.ttf mmap Pss, .ttf mmap Shared Dirty, .ttf mmap Private Dirty
	Other memory-mapped files and devices	.jar mmap Pss, .jar mmap Shared Dirty, .jar mmap Private Dirty, .apk mmap Pss, .apk mmap Shared Dirty, .apk mmap Private Dirty, Other mmap Pss, Other mmap Shared Dirty, Other mmap Private Dirty
	Non-classified memory allocations	Unknown Pss, Unknown Shared Dirty, Unknown Private Dirty, Other dev Pss, Other dev Shared Dirty, Other dev Private Dirty
	Memory Totals	TOTAL Pss, TOTAL Shared Dirty, TOTAL Private Dirty, TOTAL Heap Size, TOTAL Heap Alloc, TOTAL Heap Free
	Objects	Views, ViewRootImpl, AppContexts, Activities, Assets, AssetManagers, Local Binders, Proxy Binders, Death Recipients, OpenSSL Sockets
SQL	heap, MEMORY_USED, PAGECACHE_OVERFLOW, MALLOC_SIZE	
Network	Link layer networking	Number of ARP packets, AVG. PKT Size bytes, bps, Number of ICMP packets, Size in byte standard deviation
	Internet layer networking	Number of IPv4 packets, Network load over last minute, Maximum packet size in bytes, Minimum packet size in bytes, Number of bytes, Number of packets, Number of packets per second, Number of IPv6 packets
	Transport layer networking	Number of TCP packets, Number of UDP packets
Statistics on system calls		Number of Syscalls, No. of different syscalls, Average no. of calls per syscall, No. of calls occurring once, No. of calls occurring multiple times

The obtained results are shown in Table 2. With the three different classification algorithms considered in the study, we have obtained a precision ranging from 0.968 to 0.998 and a recall ranging between 0.988 and 0.997. The obtained F-measure is between 0.980 and 0.998 and the ROC area is ranging from 0.998 to 1.000. The classifier with best performance is J48.

Table 2. Classification results for ransomware and benign applications when features extracted by static analysis are considered using the J48, NB (Naive Bayes) and LP (Logistic Regression) classifiers.

Algorithm	Precision		Recall		F-Measure		ROC Area	
	Ransom	Trusted	Ransom	Trusted	Ransom	Trusted	Ransom	Trusted
J48	0.998	1.000	0.997	1.000	0.998	1.000	0.998	0.998
NB	0.968	1.000	0.992	0.998	0.980	0.999	0.999	0.999
LR	0.994	0.999	0.988	1.000	0.991	0.999	1.000	1.000

Table 3. Classification results for ransomware and benign applications when features extracted by dynamic analysis are considered using the J48, NB (Naive Bayes) and LP (Logistic Regression) classifiers.

Algorithm	Precision		Recall		F-Measure		ROC Area	
	Ransom	Trusted	Ransom	Trusted	Ransom	Trusted	Ransom	Trusted
J48	0.988	0.994	0.976	0.997	0.982	0.995	0.998	0.998
NB	0.382	0.975	0.940	0.605	0.543	0.747	0.922	0.943
LR	0.933	0.973	0.894	0.983	0.913	0.978	0.986	0.986

The static method classifies correctly all the benign applications (i.e., no benign applications are marked as ransomware), but nine ransomware applications are misclassified as benign. In other words, the static method has no false positive, but it has nine false negatives. False negatives are represented by three malware samples from the *Simple Locker* family and six samples from *Koler* family; all the samples from the other families are classified correctly as malware.

4.2 Dynamic Detection

As described in Sect. 2.2, the applied dynamic detection method identifies ransomware applications as such by first detecting potentially malicious execution records, and then, based on the classification of records, by classifying complete applications. In Table 3 we enclose the classification results obtained by using the classification algorithms for the detection of malicious records. While Naive Bayes provides lower detection accuracy, both Logistic Regression and J48 perform well as shown by all considered metrics. Both of these methods are of low complexity, and, thus, suitable for on-device detection. Due to its ability to assign a probability of being malicious to a record, instead of just providing a binary decision, we opted for the Logistic Regression classifier.

The output of record-level classification is, in all effects, a labelling of the execution records in the execution traces as malicious or not. This piece of information is used by the sliding windows mechanism to classify the applications as ransomware or not. In order to find the most suitable parameters of the sliding window mechanism for our scenario, we have explored different combinations

Table 4. Dynamic detection sliding window parameters used in the training phase.

Parameter	Values
Sliding window length	1, 3, 5, 7, 9, 10, 11, 12, 13, 15
Threshold (%)	60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80
Checks (no.)	1, 2, 3, 4, 5

Table 5. Dynamic detection best results obtained in the training phase with respect to the observed metrics. Detection time is measured from the first record identified as malicious.

Configuration	Window length	Threshold (%)	Checks	Detection rate (%)	False positive rate (%)	F-Measure	Detection time (s)
Highest F-Measure	9	78	1	90.82	3.46	0.85	20.46
Highest detection rate/ lowest detection time	1	60	1	96.33	19.84	0.58	0
Lowest false positives	10	70	2	80.27	2.92	0.80	60.84

of parameters, depicted in Table 4, by running a batch of experiments on the training data set. These sets of parameters were determined by running some preliminary experiments. Based on these results, we have selected the configurations that provide highest F-Measure, highest detection accuracy with false positives below 20%, lowest false positives with accuracy higher than 80%, and lowest detection time. These configurations are considered for different possible application scenarios (those in which detection accuracy would have the highest priority, those where the lowest false positives would have the highest priority, and those in which balance of both of them would be preferred). The best configurations according to the aforementioned metrics are shown, along with the corresponding detection performance, in Table 5. In these results, detection time is measured from the first execution record marked as malicious. This is done to represent the fact that the malicious behavior does not always start at the beginning of the execution of the applications containing ransomware, even though, according to our results, for most applications the first malicious record is identified within the first few seconds of execution. The best trade-off between these requirements is represented by highest F-Measure that in our case is 0.85; this configuration provides high accuracy, with a low number of false positives and a short detection time.

After selecting these optimal parameters on the training set, we tested them on the test set; the results obtained are shown in Table 6. As expected, detection performance decreases with respect to the training set, but it remains high. For example, when the parameters for highest F-measure are selected, the detection rate decreases from 90% to 85%. The case of the parameters chosen for lowest false positive rate provides best performance on the test set, with an increase of the detection rate and a slight increase of the false positive rate.

Considering the fact that execution traces in the test set correspond to ransomware samples that were not used during training (even though other samples

Table 6. Results obtained for dynamic detection in the testing phase with the best parameters obtained in the training phase. Detection time is measured from the first record identified as malicious.

Configuration	Window length	Threshold (%)	Checks	Detection rate (%)	False positive rate (%)	F-Measure	Detection time (s)
Highest F-Measure	9	78	1	85.61	5.31	0.88	24.24
Highest detection rate/ lowest detection time	1	60	1	93.03	25.06	0.79	0
Lowest false positives	10	70	2	84.68	3.81	0.89	44.72

belonging to the same families were) and, therefore, they are unknown to the classifiers, we find the obtained detection performance very promising. In fact, in real life, a mix of both known and unknown malware samples would be analyzed by the ransomware detection mechanism and this should, in our opinion, further increase the obtained detection performance.

When best F-measure parameters are considered, 62 ransomware samples go undetected. By relying on Virustotal, we have classified these samples in categories. We have taken as a reference F-secure that provides descriptive ransomware definitions. Most of the ransomware samples that are not identified by our dynamic malware detection method belong to the Simplelocker (61%) and to the Koler (19%) families. The remaining samples were not identified as ransomware by F-secure, even though other antiviruses identified them as such. Similarly to static detection, and although providing promising results, dynamic detection alone cannot detect all the observed ransomware samples.

4.3 Hybrid Detection

To evaluate the effectiveness of the hybrid approach, we have considered the list of ransomware applications in the test set that are not detected by our static method, and we have checked whether dynamic detection could correctly detect them as ransomware. All the nine applications that are not detected by our static method are correctly identified as ransomware by our dynamic method, with all the three sets of optimal parameters. Therefore, we have verified our initial assumption that by using a hybrid method we could increase the coverage of ransomware detection. In fact, starting from a detection rate of 99.8% for static detection and of 85.61% (parameters optimized for F-measure) for dynamic detection, we obtain a 100% detection rate for hybrid detection. While the static method has no false positives, the dynamic method has some, as shown in Table 6. Considering the extremely good detection performance, we can choose to optimize the dynamic method for detection speed or for the lowest number of false positives, choosing the corresponding parameters of Table 5. In summary, the results show that using the hybrid method is the way to go in order to provide effective protection against ransomware not only for its increased coverage, but also to unite the accuracy of static detection with the possibility of detecting ransomware at runtime of dynamic detection.

Finally, while our method provides 100% detection of ransomware, we need to point out that static and dynamic detection are different in what they offer. In fact, static detection is able to detect ransomware before the application containing it is executed, thereby preventing the malicious payload from executing and, thus, preventing any harm to the system. Dynamic detection, instead, detects malware while applications are being executed and, thus, when some harm to the system might have been already caused. As previously discussed, our detection method is relatively fast in detecting malicious behavior, but still it leaves open the possibility of harming the system. On the other hand, dynamic detection is able to capture ransomware at runtime, thus offering protection against dynamically installed code, which cannot be detected by static methods since they normally run with much lower periodicity, and against malicious actions hidden in obfuscated code.

5 Related Work

The main difference between ransomware and other widespread mobile malware types is in their behaviour: as demonstrated in [18], Android malware generally focuses on remaining hidden while gathering and sending to the attackers user sensitive and private information. Ransomware, instead, leverages the knowledge of its presence by users to obtain the payment of a ransom. Due to this, current methods proposed by the research community for the identification of malware are not necessarily effective in detecting ransomware, as the recent rise in ransomware attacks demonstrates. As also stated in [1], this ineffectiveness of traditional methods, when used alone, exposes more than a billion users to this threat. Effective solutions for detection of ransomware on mobile devices are needed, but up to date, there are only a few works addressing this issue in the literature. Here, we first discuss them in detail and then, we compare current literature with the approach that we propose.

The first method that introduced ransomware detection for Android is HelDroid [1]. This tool includes a text classifier based on NLP features, a lightweight smali emulation technique to detect locking strategies, and the application for detecting file-encrypting flows. The main weakness of HelDroid is that it strongly depends on a text classifier: as a matter of fact, the authors trained it on generic threatening phrases, similar to those that typically appear in ransomware or scareware. This strategy can be easily thwarted by means of techniques such as string encryption and data ciphering [15]. Furthermore, the proposed method strongly depends on language dictionaries; this is the reason why, as stated by the authors, when the analyzed ransomware is targeting non-English speakers, the dictionary must be switched to a different language. In addition, this method can be evaded by altering the occurrences of such words. From the performance point of view, they identify rightly 375 Android ransomware on a dataset composed of 443 samples: 11 ransomware were not detected due to unsupported language (e.g., Spanish, Russian) with 9 out of 12,842 false positives.

In [17], the authors propose a performance tool in order to help to understand what can be done to cope with Android ransomware detection. This tool

provides the ability to dump the log of system messages, including stack traces. However, this method remained at the level of a proposal, with no implementation. Therefore, there are no results that can prove its effectiveness.

Song et al. [16] designed an approach with the aim of identifying mobile ransomware by using process monitoring. They consider features representing the I/O rate as well as the CPU and memory usage. They evaluate the proposed method with only one ransomware sample developed by the authors. This sample has the ability to encrypt the file by using AES.

An approach based on formal methods that is able to detect Android ransomware and to identify the malicious sections in the application code is described in [11]. The authors evaluate a dataset composed of 2,477 samples with real-world ransomware and trusted applications. Starting from the payload behaviour definition, the authors formulate logic rules that are later applied to detect ransomware. The main weakness of the proposed method is represented by the human analyst effort required to build the logic rules. As a matter of fact the proposed method foresees the payload identification but the process rule building has to be done by hand, and as such, is a time consuming task.

A hybrid static-dynamic approach is proposed in [6]. In that work, applications are first scanned by using a static approach and marked as *Benign*, *Suspicious*, or *Malware*; only the applications marked as suspicious are then examined by using a dynamic approach. The static approach is based on text and image classification as well as on API calls and application permissions. The dynamic approach is based on sequences of API calls that are compared, with a periodicity of five minutes, against malicious sequences identified for each malware family. While global performance of the method is not clearly reported, it can be derived that the authors can obtain a recall of 98% with a number of false positives below 1.5%. While applying dynamic detection only on applications marked as *Suspicious* provides the ability to reduce the system overhead, it leaves the possibility of having malware not recognized as such (e.g., due to obfuscation) by static detection running on the system. Dynamic detection takes a minimum of five minutes to identify a ransomware sample.

Table 7 shows a comparison between the state-of-the-art methods for mobile ransomware detection. The only methods, as depicted in Table 7, that are able to discriminate automatically between ransomware and trusted applications are the ones proposed in [1] and in [6]. The former has the limitation that if the ransomware applications are translated into different languages than the original ones, their detector is not able to identify the threat unless re-trained. The latter, is a hybrid approach that has very good detection performance, but it applies dynamic detection only to a subset of applications that are not identified as ransomware and it takes up to five minutes to detect malware by using the dynamic method. On the other hand, the approach proposed by Mercaldo et al. in [11] is able also to localize the malicious behaviour and to obtain a precision equal to 1 with a recall equal to 0.99 but it requires the formulation of logical rules that require human intervention.

Table 7. Current literature comparison in mobile ransomware detection.

Authors	Approach	Weaknesses
Andronio et al. [1]	Text-based classification	Natural language dependent
Yang et al. [17]	High-level design	No implementation
Song et al. [16]	Process monitoring	Evaluation on one self-developed sample
Mercaldo et al. [11]	Formal methods	Requires human intervention for rules building
Gharib and Ghorbani [6]	Hybrid detection	Dynamic detection is applied only on <i>Suspicious</i> applications

The approach that we propose is different than all the others proposed in the current literature, since it is first, independent from the language application (for instance, we consider for the static analysis features derived from the structural characteristics of the code) and second, the considered extracted features (for both dynamic and static analysis) are fully automated and do not require the security analyst intervention. Additionally, our approach considers as target for dynamic detection all the running applications. In this way, also ransomware that is able to evade static detection, (e.g., by means of code obfuscation) can be identified. While our method requires continuous monitoring of the applications, it uses features and methods that are not as demanding as sequences of API calls and it can detect ransomware, on average in 20–60s, depending on the configuration chosen. The main weakness of the method proposed by Song et al. [16] is that the authors use only one ransomware sample, developed by them, to evaluate their solution. This makes any comparison with other methods, in terms of detection performance, impossible. Considering that approach proposed in [17] is only a high-level design with no implementation and validation about its effectiveness, it is not possible to perform a comparison between our proposal and the considered approach.

6 Conclusions

Mobile ransomware attacks are on the rise and pose threat not only to companies and governmental institutions but to the entire society. Both static and dynamic methods commonly used for ransomware detection offer good performance alone, but, according to our results, none of them can detect all different ransomware samples. Also other state-of-the-art methods do not provide effective solution.

In this paper, we propose a hybrid approach to ransomware detection that has a 100% detection rate coupled with a false positive rate below 4%, even when analyzing previously unseen applications. This performance was achieved using dynamic method to complement the static one, thus increasing coverage and allowing us to combine the advantages of both methods. Finally, given high

achieved detection accuracy on one hand and the detection methods of low complexity used for on-device dynamic detection on the other, we are convinced that such hybrid method can be used to detect ransomware not only in mobile phones but also in other IoT devices.

Acknowledgements. This work has been partially supported by H2020 EU-funded projects NeCS and C3ISP and EIT-Digital Project HII.

References

1. Andronio, N., Zanero, S., Maggi, F.: HELDROID: dissecting and detecting mobile ransomware. In: Bos, H., Monrose, F., Blanc, G. (eds.) RAID 2015. LNCS, vol. 9404, pp. 382–404. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26362-5_18
2. Canfora, G., De Lorenzo, A., Medvet, E., Mercaldo, F., Visaggio, C.A.: Effectiveness of opcode ngrams for detection of multi family android malware. In: 2015 10th International Conference on Availability, Reliability and Security (ARES), pp. 333–340. IEEE (2015)
3. Canfora, G., Mercaldo, F., Visaggio, C.A.: Evaluating op-code frequency histograms in malware and third-party mobile applications. In: Obaidat, M.S., Lorenz, P. (eds.) ICETE 2015. CCIS, vol. 585, pp. 201–222. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30222-5_10
4. Canfora, G., Mercaldo, F., Visaggio, C.A.: Mobile malware detection using op-code frequency histograms. In: Proceedings of International Conference on Security and Cryptography (SECRYPT) (2015)
5. Carbonell, J.G., Michalski, R.S., Mitchell, T.M.: An overview of machine learning. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M. (eds.) Machine learning. SYMBOLIC. Springer, Heidelberg (1983). https://doi.org/10.1007/978-3-662-12405-5_1
6. Gharib, A., Ghorbani, A.: DNA-Droid: a real-time android ransomware detection framework. In: Yan, Z., Molva, R., Mazurczyk, W., Kantola, R. (eds.) NSS 2017. LNCS, vol. 10394, pp. 184–198. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64701-2_14
7. Infosec Institute: Evolution in the World of Cyber Crime. Technical report Infosec Institute, June 2016. <http://resources.infosecinstitute.com/evolution-in-the-world-of-cyber-crime/#gref>
8. McAfee Labs: McAfee Labs Threats report - December 2016. Technical report. McAfee Labs, August 2016. <https://www.mcafee.com/au/resources/reports/rp-quarterly-threats-dec-2016.pdf>
9. Martinelli, F., Mercaldo, F., Saracino, A.: Bridemaid: An hybrid tool for accurate detection of android malware. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 899–901. ACM (2017)
10. Martinelli, F., Mercaldo, F., Saracino, A., Visaggio, C.A.: I find your behavior disturbing: static and dynamic app behavioral analysis for detection of android malware. In: 2016 14th Annual Conference on Privacy, Security and Trust (PST), pp. 129–136. IEEE (2016)
11. Mercaldo, F., Nardone, V., Santone, A., Visaggio, C.A.: Ransomware steals your phone. Formal methods rescue it. In: Albert, E., Lanese, I. (eds.) FORTE 2016. LNCS, vol. 9688, pp. 212–221. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39570-8_14

12. Mercaldo, F., Visaggio, C.A., Canfora, G., Cimitile, A.: Mobile malware detection in the real world. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 744–746. ACM (2016)
13. Milosevic, J., Ferrante, A., Malek, M.: Malaware: Effective and efficient run-time mobile malware detector. In: The 14th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2016). IEEE Computer Society Press, Auckland, New Zealand (2016)
14. Milosevic, J., Malek, M., Ferrante, A.: A friend or a foe? Detecting malware using memory and CPU features. In: 13th International Conference on Security and Cryptography SECRYPT 2016 (2016)
15. Rastogi, V., Chen, Y., Jiang, X.: Droidchameleon: Evaluating android anti-malware against transformation attacks. In: Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, pp. 329–334. ACM (2013)
16. Song, S., Kim, B., Lee, S.: The effective ransomware prevention technique using process monitoring on android platform. In: Mobile Information Systems 2016 (2016)
17. Yang, T., Yang, Y., Qian, K., Lo, D.C.T., Qian, Y., Tao, L.: Automated detection and analysis for android ransomware. In: IEEE 17th International Conference on High Performance Computing and Communications, IEEE 7th International Symposium on Cyberspace Safety and Security, IEEE 12th International Conference on Embedded Software and Systems, pp. 1338–1343. IEEE (2015)
18. Zhou, Y., Jiang, X.: Dissecting android malware: Characterization and evolution. In: 2012 IEEE Symposium on Security and Privacy (SP), pp. 95–109. IEEE (2012)



Deception in Information Security: Legal Considerations in the Context of German and European Law

Daniel Fraunholz¹(✉), Christoph Lipps¹, Marc Zimmermann¹,
Simon Duque Antón¹, Johannes Karl Martin Mueller²,
and Hans Dieter Schotten¹

¹ Intelligent Networks Research Group,
German Research Center for Artificial Intelligence, Saarbrücken, Germany
{daniel.fraunholz,christoph.lipps,marc.zimmermann,duque.anton,
hansdieter.schotten}@dfki.de

² The Project Group: Constitutionally Compatible Technology Design,
University of Kassel, Kassel, Germany
johannes.mueller@uni-kassel.de

Abstract. Deception systems have produced promising results in protecting networks from recent attack campaigns. Their development and operation, however, is regulated by technical and legal circumstances. There are several aspects to be considered when operating a deception system, such as privacy, entrapment and liability. In addition to these general aspects, domain specific law that, for example, applies to research or government, needs to be accounted for. In this work German and European law was investigated with respect to deception systems focusing on the aspects listed above and others. The findings are applied to the design, operation of a Honeypot, as well as the generation and publication of information. We found that it is not forbidden to use deception systems in general but several facets have to be considered in the technical implementation.

Keywords: Information security · Privacy · Deception · Honeypots
European law · German law

1 Introduction

Deception technology and especially Honeypots are an advanced IT-security mechanism to oppose cyber crime. This technology relies on purposely providing false or delayed information, hiding information and misleading ongoing attack campaigns into a course controlled by the operator of the deception systems. Usually the parried attacks are monitored and analyzed to gain further insight of the adversaries intentions and approaches. Questions about privacy, copyright and other legal interests are thrown up by this process. Furthermore, devices connected to the Internet can be reached globally. Still domestic law of every participating party applies which leads to legal uncertainty, especially since the origin

of an attacker is not known beforehand. In this work the legal concerns when employing such technology are investigated. Related work is listed in Sect. 2. We first introduce technical aspects and major peculiarities in Sect. 3. In Sect. 4 we introduce relevant parts of German, European and international law. These laws are refined by domain specific law in some cases. In some cases, such as governmental or research applications, domain specific law has to be applied in addition to the general law. Examples of relevant domain specific laws are given in Sect. 5. In Sect. 6, the application of the findings are mapped towards real-world Honeypots and the publication of data. It is motivated by previous works of the authors that faced the legal considerations described in this paper. The findings are concluded in Sect. 7.

2 Related Research

Previous work is mostly focused on American law. Legal concerns in all of the reviewed publications are: Liability, entrapment and privacy [9, 11, 12]. To the best of our knowledge, there is only one work where the concept of Honeypots is investigated in the light of European law [13]. The law of a specific European country has not been analysed with respect to deception systems yet.

3 Technological Aspects

In this section, the technical intricacies of deception systems are introduced that impact the legal evaluation. Honeypots are the most common kind of deception system. They can be distinguished by their type, the deployment strategy, the level of interaction and the counter attack strategy.

3.1 Honeypot Types

Common types are server-side, client-side and token Honeypots. Server-side Honeypots are considered passive in their context. They wait for incoming connection and respond as a genuine server would. In contrast to that, client-side Honeypots actively connect to servers and pretend to be a genuine client system. Honeytokens, such as Honeyfiles and Honeylinks, are data or information embedded in a context like a *HTML*-document or database. Like server-side Honeypots, tokens are passive and wait for an attacker to illicitly access or misuse them otherwise. In this work, we focus on server-side Honeypots, as there is a broad amount of deception technologies that can hardly be considered in full in one scientific work.

3.2 Deployment and Intention

The deployment is only relevant to server and token Honeypots. Research and production deployment modes are distinguished. Research mode is employed to

enable a broad amount of attacks. A common research mode deployment is connecting a Honeypot directly to the Internet and making it addressable with a public IP address. These systems are employed to investigate large scale campaigns such as botnets and common exploitation techniques to access restricted resources from an external context. On the other hand production mode deployment is a strategy where the Honeypot is within a non public context. Interaction on these Honeypots always indicates breaches in the perimeter thus revealing compromises in early stages.

3.3 Level of Interaction

Honeypots may differ in the depth of emulation of the resource they mimic. Telnet servers, for example, typically prompt for an authentication when connected. A system only registering the connection or emulating the login prompt would be considered as a low-interaction system. More advanced systems grant access and provide the full functionality of the given operating system. These systems are considered as high-interaction systems. In literature, medium-interaction systems are not consistently defined but commonly placed between the functional scope of low- and high-interaction systems.

3.4 Counter Measures and Aggressive Honeypots

Recently, more aggressive counter measures such as hacking back the attackers are discussed in the context of self defence [8]. From a technical perspective counter attacks can be classified in the same taxonomies as the initial offensive. They extend from denial of service or resource exhaustion techniques to more specific attacks such as dictionary or brute force attacks against the maintenance protocol (most likely *Secure Shell*) of the attacking server. Mirroring of the attacking technique is promising, particularly against propagating malware or botnets since the system was most likely compromised originally by the applied technique.

4 Legal Concerns with Deception Systems

In this section the impact of German and European law on the above introduced technical intricacies is discussed. We identified five legal key aspects that need to be taken into consideration when operating a deception system. A simplified relation of German national and European law is shown in Fig. 1.

The Basic Constitutional Law of the Federal Republic of Germany (*GG*) is derived from the German constitution. All domain specific laws have to be compatible to the basic law as well as the constitution. More than that there are two kinds of European legislation: Directives and regulations. Regulations are directly applied in each member country, while directives need to be adapted into national law first. The application of law in Germany therefore consists of the Basic Law, the domain specific laws that derive from the German constitution as well as European directives and European regulations.

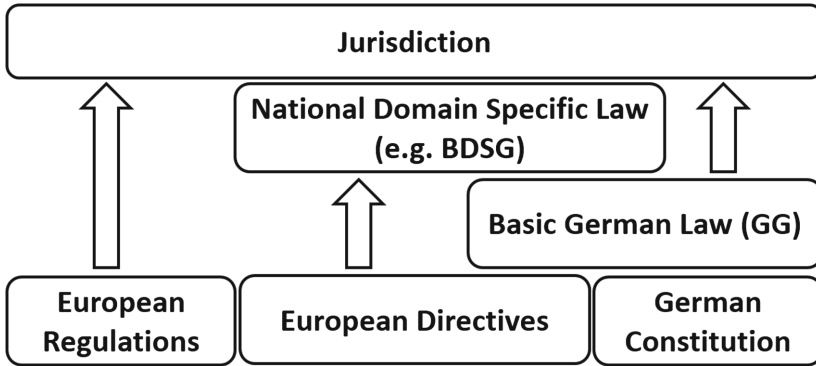


Fig. 1. Application of law

4.1 Privacy

In this section, European regulations and German laws are discussed in the context of privacy. A major task of deception systems is the collection of threat intelligence. The basis of threat intelligence is information collected from the interaction with deception systems. Deception operators need to consider the amount of data they intend to collect with respect to the regulations and laws. §8 of the European Convention of Human Rights provides a right to respect for private and family life in home and correspondence. This convention was published in 1950 and is the foundation of all European and national regulations and laws. It states that persons working with technologies that are capable of collecting or processing personal data where a specific person may be identified are responsible to prevent corresponding violations of privacy.

Fundamental European Rights. In 1981, the first appearance of European law containing regulations for the processing of personal data has been published as the European treaty series No. 108 (Convention for the Protection of Individuals with regard to Automatic Processing of Personal Data). This convention specifies a right of privacy for individuals, with regard to autonomic processing of corresponding personal data. In §5 it is clarified that personal data has to be obtained and processed “fairly and lawfully”, stored only for specified and legitimate purposes and no longer than required. §5 is only applied to data, that enables identification of a data subject. As a consequence, it is legitimate to store collected data if it is impossible to identify the corresponding person. In §5 an exception for the processing of data in deception systems for scientific research or statistical purposes as well as for law enforcement is given, with respect to the binding to their purpose. The exception requires, that there is no obvious risk of a violation of privacy. However, in §8 every person is granted the right to assert the existence as well as main purpose of collected personal data and the identity of the data controller. Additionally, every data subject is able to demand the deletion of all relevant data, if the criteria for collection are not fulfilled.

European Directives. In 1995, the European Union published the directive *95/46/EC* that focuses on the protection of individuals with regard to the processing of personal data and on the free movement of such data. This directive establishes the fundamental rights in regulations all member states need to regulate in national law. In Art. 7, criteria for the legitimate processing of personal data are named. According to this paragraph, data can only be processed if processing is necessary for

- the data subject unambiguously given consent,
- the performance of a contract,
- compliance with a legal obligation,
- the performance of a task carried out in public interests or vital interests of the data subject or
- legitimate interests as long as they do not interfere with fundamental rights of the data subject.

A person whose data has been collected has the rights to get all relevant information about the data and the data controller and has to be notified if someone obtains personal data that has not been directly obtained from that person. According to §17, the controller has the duty to take care of an up-to-date security to protect the data and the processing of the data from accidental or unlawful usage or destruction. EU Directive *2002/58/EC* is an addition to *95/46/EC* and focuses on the processing of personal data and the protection of privacy in the sector of public electronic communications. §5 of this directive contains regulations about the confidentiality of the communication. Communication networks and services shall prevent all kinds of tapping, interception or surveillance of communications without consent. The directive distinguishes data in traffic and location data. Traffic data is used for the transmission of messages and has to be deleted or anonymized when it is no longer required for the purpose of transmission. Traffic data can legally be processed for billing of subscribers and location data only within the necessary duration for transmission or if they are anonymized.

General Data Protection Regulation. EU directives *95/47/EC* as well as *2002/58/EC* are the two important directives in the context of collecting data with deception systems. However, due to the fast technical progress the directives are obsolescent and hardly cope with modern communication. The European Union published two new regulations, that are coming into force on May 25th in 2018. EU regulation *2016/679/EC* General Data Protection Regulation (GDPR) annuls directive *95/46/EC* and *2017/0003/EC* (e-Privacy) will annul the prior directive *2002/58/EC*. Both regulate the same domain as their predecessor but cover more details with respect to new processing techniques. In *2016/679/EC*, it is described that a data controller needs to inform a person at the time of collection but not in case of getting data otherwise and for the purpose of scientific research. Personal data controller have more obligations such as

- securing the data,
- keeping a register about all data and processing steps and
- informing the supervisory authority if processing may violate the privacy of certain persons.

Researchers can process data if they guarantee to respect all rights of privacy and only use a “minimal amount of data”. If possible the data needs to be anonymous. Otherwise, it needs to be pseudonymous.

German Law. The foundation for domain specific law is composed by the *GG*. It establishes several fundamental rights with respect to privacy such as:

- Human Dignity (Art. 1 *GG*),
- Personal Freedoms (Art. 2 *GG*),
- Privacy of correspondence, post and telecommunications (Art. 10 *GG*) and

Article 10 regulates the inviolable privacy of correspondence, mail and telecommunication. However, Sect. 2 state the inviolability as restricted pursuant to other law. Please note, that fundamental rights do not directly apply between natural persons. The Federal Data Protection Act (*BDSG*) is the most important law for privacy in Germany and is currently adapted to the *GDPR*. It focuses on the rights of a person about processing personal data and privacy, thus realizes EU directive *95/46/EC*. Both state that personal data is only collected for a “limited duration of time” and only if “required to provide services” or if the user gave his permission to do so. The *BDSG*, however, allows several exceptions. §4a dictates that the permission has to be given in written form except for valid scientific purposes. Additionally, §4d commits data processing organizations to report to supervisory authorities about automated data processing techniques. German law differentiates between collecting, processing and using data. Collecting data for scientific purposes is not explicitly allowed without the permission of the user, but the processing of received data is allowed. According to §40, personal data, that was collected for scientific purposes can only be used for those purposes and has to be anonymized as soon as possible. The German Telemedia Act (*TMG*) describes the rights and obligations for all electrical information and communication services. The §15 *TMG* state that personal data can only be collected if it is necessary to provide the service or for billing. As in *95/46/EC* the provider has to inform the user about collection and purpose of data and the user has to give his permission. §13 also declares that the data has to be deleted right after the end of the service. The *TMG* is very restrictive, deception systems with extensive data collection may be illegal in the context of the *TMG*. However, the term *electrical information and communication service* is defined in §1 of the *TMG* and a deception system may not pose a service as required to apply the *TMG*. §13 sec. 7 states that the provider ensures the security of the service and the data. This clause may allow the collection of data of potential threats to the service.

Court Judgments. Due to the lack of detail about the content of personal data, there have been several trials. Those trials were mainly about the legitimacy of the collection of personal data and special types of data such as IP addresses. In Germany, the Higher Regional Court of Cologne dealt with those questions in the case *12U16/13* in December 2015. In this case, they had to decide if it is legitimate to hold information about dynamic IP addresses for 4 days. The defendant, a small size service provider, collected data to protect the system from disruptions. From the courts point of view, Denial-of-Service attacks, spam mails and malware can result in disruptions. Therefore, it is allowed to collect and process relevant data as long as needed to get important information about potential disturbances. This case affects providers of communication services but not providers of media services. In October 2016, the European Court of Justice dealt with a case about a litigation between the federal public of Germany and Mr. Breyer that was initialized in October 2014. Breyer stated that it is illegal to collect information about user containing time data and dynamic IP addresses for provider of online media services. The court decided that this data is personal if the provider has the legal means to get the identity of the person behind the address. Furthermore, it is not allowed to store this information for a longer duration than necessary to provide the requested service. Due to this judgment, the German Federal Supreme Court of Justice had to form an opinion about the original case. In May 2017 an announcement was made that no final consideration could be made.

4.2 Entrapment and Accessoryship

Entrapment is not regulated by European law. Members of the European Union are self dependent in this domain. In Germany entrapment is covered by §26 and §30 Penal Code (*StGB*). However, the requirement of both is to dictate somebody to commit a crime. A criminal abusing a honeypot had the criminal intention before. The criminal actively searched for vulnerable systems and exploited the identified vulnerability to take advantage of the honeypot. Even if a client side honeypot is considered, the attacker did set up a server with offensive abilities before. She also implemented a trigger to employ the offensive capacities against a connecting system. Honeypots do not dictate criminal activities to intruders. A conviction for entrapment is unlikely. §27 *StGB* covers accessoryship: “Intentionally rendering aid to another in intentional commission of an unlawful act”. This could be the case if a deception system is used to perform attack against third parties. In this case the operator of said deception system could be considered an accessory, since she aids the attacker by implicitly providing resources.

4.3 Liability and Other Claims

Any operator of a service can be liable for damage caused by this service. Liability is specifically affecting honeypots as the operator is liable for damage occurring from the honeypot operation as well. As Honeypots are intended to be exploited,

the risk of an attack propagating because of misconfiguration is high compared to well maintained systems. Liability is not regulated in the European Law. In German Law liability is regulated in §823 Civil Code (*BGB*). Liability needs to be considered in several cases:

- Damage of third parties due to an intruders interaction with the honeypot,
- Damage due to information published that was collected with the honeypot.

To palliate the probability for a conviction, technical measures need to ensure a proper policy enforcement. Policies need to be defined to mitigate pivoting attempts. Additionally, the collected data needs to be handled with a state of the art security level and publications need to fulfill the requirements of privacy. With adequate technical measures, a reduction of the risk of being held liable to the level of a common service can be achieved. Honeypots do not expose the operator to significant juridical consequences any more than other systems. In addition to liability, it may happen that a honeypot collects personal data from third parties. According to Art. 5, *2016/679/EC*, this kind of collection is illegal as it is not related to a valid cause. Violating the *2016/679/EC* can result in claims against the operators.

4.4 Copyright

High-interaction deception systems enable extensive monitoring of intrusion campaigns. These systems may be able to obtain malicious programs from attackers and botnets. Malicious programs can be obtained in the form of compiled binaries or a sequence of commands. Any person has the rights on her intellectual property, no matter how the program is used. In European law, the directive *2004/48/EC* focuses on the enforcement of intellectual rights. §5 of this directive states that the author of literature or art is determined by a name, indicating the works author. Additionally, in directive *2009/24/EC* on the legal protection of computer programs §1 claims that computer programs are protected by copyright as pieces of literature. In German law, the Act on Copyright and Related Rights (*UrhG*) regulates intellectual rights. This act has the same formulations with regard to computer programs as the European law. According to §§15-22, the author has the right to choose about publication, duplication, spreading and exhibition. If a work is free to use it can be taken as basis to create a new product without notifying the author of the original work. In §§97-99 *UrhG*, the author of any kind of written work can insist on stopping the usage of his intellectual property. After this warning, proceedings against the usage can be initiated to compensatory damages. However, in case of an attacker who creates malicious software it is very unlikely that she will claim damages because she usually has created the software to do illegal actions she can be condemned for.

4.5 Self-defence

Self-defence, for example *hackback*, in the context of cyber crime, describes counter measures against intruders or intrusion attempts. These counter

measures can be criminal acts by §202a-d *StGB*, §206 *StGB*, §263a *StGB* or §303a-b *StGB* depending on the technical design of the counter measure. However, the German law allows active counter measures, such as hackback, under specific circumstances. There are two different cases: Self-defence and necessity. Self-defence is “any defensive action to avert an imminent unlawful attack on oneself or another”. It requires that the attacking system is juristic property of the attacker. This cannot be ensured as most attacks stem from infected systems. More than that self-defence also requires the attack to be ongoing. From a technical perspective honeypots are able to trigger instant counter attacks. This requirement can therefore be fulfilled, if the technical implementation is adequate. However, self-defence salvages the risk of a conviction if a third party system is attacked. Necessity means the aversion of an imminent danger, such as a cyber attack, upon a legal interest with any means necessary, given they are proportional and given the legal interest outweighs the effect of the counter measure. In case of necessity, third party actors are also allowed to be attacked. Preventative counter measures are also allowed if a threat is present. Defence motivated by necessity needs to be adequate with respect to the opposed threat. This implies that the kind of attack needs to be considered for the counter measure. For example port scans may not be a legitimate response to distributed denial of service attacks as response. Counter measures are a legitimate option against attackers. However, the context is significant. Adjacent to the discussed situations, the operator is a major factor. State involved counter attacks may pertain international or martial law.

5 Domain Specific Law

In this section we discuss specific domains in which general regulations are supplemented, as, for example, the *StGB* does for criminal law. Table 1 gives an overview of the investigated domains and the corresponding codes of law. The domains law enforcement, research, federal law & public sector and telecommunication providers are the four domains that were identified as relevant for application in the area of deception systems.

Table 1. Domains and corresponding laws

Domain	Corresponding law
Law enforcement	§100g <i>StPO</i> , §7 BKAG
Research	§28 S2 no.3 <i>BDSG</i> , §40 <i>BDSG</i>
Federal law & public sector	§§13,14 <i>BDSG</i>
Telecommunication providers	§96 <i>TKG</i> , §100 <i>TKG</i>

Subsequently an overview of different cases/institutions with specific laws that are able to extend general laws is given.

5.1 Law Enforcement

In order to protect personal data in the purpose of the prevention, investigation, detection or prosecution of crime or the execution of criminal penalties, respectively, there is the directive *2016/680 EC*, wherein the fundamental right and freedom of natural persons and their right to the protection of personal data is fixed. As all European directives it needs to be transferred in national law, which is currently in process in Germany. There still exist some sections in German law that allow to collect personal data for law enforcement. In German law there is a fundamental understanding of the commensurability of a governmental intervention. In this context the

- suitability,
- necessity,
- reasonability and appropriateness

of the intervention must be given. Beyond that, as already mentioned above, any legal foundation is required for every intervention. For instance conditions regarding the interception of telecommunication are defined in §100a Criminal Procedure (*StPO*). It is allowed to intercept and record telecommunication, also without the knowledge of the concerned person, if certain facts give rise to the suspicion that a person focuses a serious crime or, in cases where there is criminal liability for attempt, has attempted to commit such a crime or has prepared such a crime.

Due to §100g *StPO* it is also allowed to collect communication traffic data, in terms of §96 *TKG*, if certain facts give rise to the suspicion that a person has committed a crime, in cases where there is criminal liability for attempt, has attempted to commit such a crime or has prepared such a crime or has committed a crime by the means of telecommunication.

Extended permissions are also given to governmental authorities such as the Federal Criminal Police Office. According to §7 Federal Police Law (*BKAG*) they are allowed to collect, process and use privacy data, as well as, for example, time and scene of a crime (§8 *BKAG*), if this is necessary in compliance to their task as central office of law enforcement. Furthermore they are allowed to impose and store other privacy data.

5.2 Research

The German law is very detailed in the context of data security. According to §28 Sentence 2 no. 3 *BDSG* the collection and storage of data shall be admissible, if it is necessary in the interest of a research institute for conducting scientific research. Sentence 6 no. 4 of the same section, extends these permissions in a way that not only the collection but also the processing and the use of special types of personal data is allowed, if necessary, for the purposes of scientific research. But there are also some restrictions, there is a specific section within the *BDSG* especially for research institutes. §40 defines that collected and stored personal data may processed or used only for scientific research purposes. Due to sentence

2 it is also necessary to anonymize personal data if this is possible. Sentence 3 states that a publication of personal data is only allowed if the data subject has consented or if the data is indispensable for the presentation of research findings on contemporary events. Researchers intending to publish results, coercively need to anonymize these.

In European jurisdiction there is a restriction for the storage of privacy data for the purpose of research. According to §6 Sect. 1e *2000/31 EC*, data shall not be stored longer as necessary for the purpose and it shall not be possible to identify participating entities.

5.3 Federal Law and Public Sector

Public bodies of the Federation are regulated in a specific section within the *BDSG*. In the sections §§12 – 14 *BDSG* it is defined which actions are explicitly authorized. The collection of personal data shall be admissible if the knowledge of them is needed to perform the duties of the bodies collecting them (Sect. 13 sentence 1 *BDSG*). If it is necessary for the performance of the duties of the controller of the filing system and if it serves the purpose for which the data was collected, it is allowed to store, modify or use personal data (Sect. 14 sentence 1 *BDSG*).

The German Federal Office for Information Security is responsible for the security within information technology. In order to defend against risks for critical infrastructure, they are empowered to collect and evaluate data, especially information about security vulnerabilities, malware, happened or attempted attacks aligned on information security and also the exact proceeding of the attackers (§8b S.2 Nr. 1 - *BSIG*).

5.4 Telecommunication Providers

Service providers are committed to safeguard the secrecy of telecommunication according to §88 Telecommunications Act (*TKG*). Furthermore they are obligated to protect the personal data of communication participants (§91 ff. *TKG*). If there is any justifying purpose, however, they are granted certain permissions, like collecting communication traffic data, especially phone numbers, and connection meta data, such as time and duration (§96 *TKG*). In case of disturbance of their infrastructure, they are allowed, according to §100 *TKG*, to collect additional data. To define the case of disturbance there is a legal decision of the Higher Regional Court of Cologne (*I-12 U 16/13 OLG Koeln*) in which cyber attacks are defined as such a disturbance. According to §98 *TKG*, they are also allowed to collect location data if they provide additional services relying on this data, which however needs to be anonymised. A specific restriction for the storage of this data is the duration. Service providers need to retain data only for a period of ten weeks, location data only for four weeks (§113b *TKG*).

6 Honeypot Design and Threat Intelligence

In this section, we apply the findings from the previous sections to the design and threat intelligence of Honeypot systems. Regulations that restrict the collection and usage of data are considered, as well as possibilities to still gather and publish information. Furthermore, lessons learned during the research for this work and the operation of Honeypots are explained.

6.1 Application

The design, operation and threat intelligence can be split into four categories. First, in the deployment, the kind of Honeypot to operate is chosen. After that, the possible operation modes are introduced. The data processing is evaluated after that. Finally, the possibilities for publishing information and insights are discussed.

Deployment. Due to the specific nature of production Honeypots they need to follow significantly less restrictions than research Honeypots. As they are placed within the perimeter an attacker has to have breached the network security, having committed a crime already. The authors conclude that entrapment is not an issues. As the operator is always liable, measures have to be taken to ensure the attacker does not influence third party systems. This can be achieved firewall rules and policy enforcement. A best practice is to block or limit outgoing traffic [4]. Blocking can be problematic as no interaction is possible, for example with the Command and Control (*C&C*)- or download-server, hindering further analytics. Sometimes, the inability to create outbound traffic leads to the deletion or abort of infection routine of the malware, a common anti-forensics mechanism [3]. In contrast to research Honeypots, production Honeypots are usually placed in productive environments. If activity can be detected, an attacker has obviously gained access to this environment, potentially endangering the production facility. Counter measures as means of defending the production infrastructure can therefore be considered as self-defence.

Research honeypots, on the other hand, are usually publicly accessible and not connected to productive systems. Therefore, any client can access them and no damage to one entity's assets is imminent. This makes entrapment possible, as it offers obvious vulnerabilities. Entrapment, according to German and European law, requires more than just offering an opportunity, however. It is necessary to actively try to get a victim to perform an illegal action, which cannot be seen here, neither for client-side, nor for server-side Honeypots, according to the authors. Liability lies with the operator, as she has to make sure that no outbound traffic can harm any third-party system. The best practices are the same as described above.

The authors operate several research Honeypots, whose deployment has been described in previous works [5]. As no outbound traffic is possible on our systems, we take no risk of liability for damage on third-party systems. It is, however, not

inconceivable that an attacker compromises the underlying operating system, for example via a previously unknown vulnerability, to execute attacks against third parties. In this case, liability seems improbable, as the code was created according to best practices for secure programming.

Operation. Honeypots can be operated in different fashions. One of the most important distinguishing feature is their ability to counterattack. Especially since malware usually runs on host systems that are not owned by malicious adversaries, hackbacks can only be executed under certain conditions, as described in Sect. 4. To verify these conditions beforehand is infeasible, making hackbacks risky. On the other hand, many botnets have only been defeated by law agencies because traffic was infiltrated into *C&C* communication [1]. The authors conclude, that law enforcement agencies have a higher tolerance for actively counterattacking malicious adversaries than research institutes. The Honeynet we previously introduced [5] is not capable of hacking back.

Data Processing. Honeypots are created and operated to gather data. Typical kinds of collected data are *IP*-addresses, timestamp, location of the attacker and payload, such as credentials or command sequences. *IP* addresses will supposedly be considered personal data according to German and European law [2, 6]. Timestamp, location and other metadata can be personal data if they are able to identify an individual. If the conjunction of different kinds of data with metadata allows someone to identify an individual, metadata is classified as personal data. The collection of metadata therefore poses a threat to the operator of Honeypots, since it can be personal data to which strict regulations apply. Payloads of attacks can be copyright protected data, especially if an attacker exploits a vulnerability that was unknown beforehand. Such exploits are sold for sums of several thousand dollars up to \$1.5 million [15]. The simple collection of copyright protected data, however, does not pose a problem to the operator of the Honeypot.

According to German and European law, personal data has to be deleted as soon as it is no longer required for technical reasons, for example offering web services. The purposes of research and law enforcement create exceptions. Law enforcement agencies are allowed to store personal data as long as the investigation is ongoing and it is allowed by a judge. Research institutes are allowed to store personal data for the duration of their research. It needs to be noted that all assessments in this section only account for storing data, not distributing or utilising it. The author's Honeypots store all this information in a secure way to comply with German privacy protection regulations.

Publication. Publication and distribution of collected data is desirable for different reasons.

Law enforcement agencies exchange data in the context of cross-border prosecution of crime. Internet architecture makes it easy for an attacker to avoid

the jurisdiction of the country she is attacking in by originating her attack in a different country. This needs for international cooperation in crime fighting.

Research institutes publish scientific results and findings. The publication of personal data, however, is strictly forbidden, unless it is necessary to describe the situation or unless it is a person of public interest. That means personal data has to be anonymised to avoid legal consequences. Another possibility of making use of the data and publishing results is statistical analysis and release of the findings. This allows for full usage of the data without compromising personal data and has, among others, been employed by the authors in previous works [5]. Another common practice, as practiced by *Google Analytics* [7] for example, is deleting one octet of an IP address. This way, some information, such as local area, can be obtained without disclosing identities.

Telecommunication provider. Usually, personal data, such as IP and access times, are stored by the telecommunication providers only for a short duration of several days. This data can be shared with law enforcement agencies for prosecution of crimes, as well as with other providers for misuse prevention.

Copyright protected data, as described in Sect. 6.1, must not be distributed or copied in an unauthorized manner. However, since the creator of this data has to claim his rights, she will inevitably admit for having committed cyber crimes. This makes it unlikely for the operator of a Honeypot to be held accountable for distribution of copyright protected data, even though she is technically transgressing the law.

6.2 Lessons Learned

During the operation of our Honeynet, we found that the legal foundations can change in a way that influences our research. This leads to the insight that constant evaluation of the legal landscape is necessary for operators of deception systems in order to prevent being prosecuted. Sometimes, the laws change for the better from a research institutes perspective, as with the *General Data Protection Regulation*, presented in Sect. 4. It explicitly states exceptions for storage of personal data for research institutes. Beforehand, this was tolerated but not regulated by German law. Despite the release of laws, a factor of uncertainty always lies in the jurisdiction. The relatively new topic of cyber crime and its defence has not often been dealt with in court, making the outcome uncertain as there are not many test cases. Furthermore, a challenge lies in the origin of attacks. According to common law, the jurisdiction of the country of origin is applied, making it infeasible for the operator of a Honeypot to check all possible laws. The authors of this work alone have monitored attacks from 174 countries in 222 days and analysed access from 95 countries monitored during 111 days in previous works [5].

In previous works, we published statistical analysis of the attacks to avoid legal issues. The published data is not suitable to identify an individual. For research projects the authors are currently working on, the generation of attack

signatures is required. This could create a conflict due to the implicit publication of copyright protected content and needs to be checked thoroughly. German institutes, such as the *Deutsche Telekom AG* with their *DTAG Honeynet* [14], only publish the country of origin, the timestamp and the content of any attack. This information without an *IP*-address is not able to identify individuals. American institutes, such as the *Norse Corp.* with their *NorseMap* [10], publish the *IP* addresses as well. This is due to the significantly different laws on personal data in the United States of America. These were out of scope in this work, but highlight the drastic differences in jurisdiction between different countries.

Art. 5, 2016/679/EC demands several conditions to be fulfilled for processing of data to be legal. Many of them, for example the right of deletion of data or the necessity of a valid cause are inherently incompatible with the idea of deception systems. A legally sound implementation is a challenging task, as many restrictions on anonymisation are to be met. This will be discussed in future works.

7 Conclusion

There are several European directives that take all relevant aspects of the legitimacy of deception systems into consideration, as well as regulations, such as the *GDPR* and the *e-Privacy* regulation, that will be applied by national jurisdiction. They contain regulations on privacy, entrapment, liability, copyright and self-defence. Additionally, most of them distinguish between domains such as law enforcement, research, public sector and telecommunication providers. European directives have to be implemented into acts by every member state. Therefore, operators of honeypots have to consider which domain they represent and which law is relevant for them. Especially in case of collecting and processing personal data for research purposes, there are restrictive regulations regarding privacy implemented in European law. For further processing and publishing results, personal data has at least to be pseudomised or anonymised so that it is not possible to identify a certain individual. Several research and Honeypot projects show that it is possible to operate a Honeypot and publish the results in a legally conform way.

Acknowledgment. This work has been supported by the Federal Ministry of Education and Research of the Federal Republic of Germany (Foerderkennzeichen KIS4ITS0001, IUNO). The authors alone are responsible for the content of the paper.

References

1. Andriess, D., Rossow, C., Stone-Gross, B., Plohmann, D., Bos, H.: Highly resilient peer-to-peer botnets are here: An analysis of Gameover Zeus. In: International Conference on Malicious and Unwanted Software, vol. 8, pp. 116–123 (2013)
2. Bundesgerichtshof: Bundesgerichtshof zur zulässigkeit der speicherung von dynamischen ip-adressen (2017)

3. Edwards, S., Profetis, I.: Hajime: Analysis of a decentralized worm for IoT devices
4. Fraunholz, D., Pohl, F.: Towards basic design principles for high- and medium-interaction honeypots. In: European Conference on Cyber Warfare and Security, vol. 16 (2017)
5. Fraunholz, D., Zimmermann, M., Duque Anton, S., Schneider, J., Schotten, H.D.: Distributed and highly-scalable WAN network attack sensing and sophisticated analysing framework based on honeypot technology. In: International Conference on Cloud Computing, Data Science & Engineering, vol. 7 (2017)
6. Gerichtshof der Europäischen Union: Urteil in der rechtssache c-582/14 (2016)
7. Google Inc.: Google analytics (2017). <https://analytics.google.com/>
8. Koch, A.: Die rechtlichen rahmenbedingungen von hackback (2008)
9. Mokube, I., Adams, M.: Honeypots: Concepts, approaches, and challenges. In: Proceedings of the 45th Annual Southeast Regional Conference (2007)
10. Norse Corp.: Norse attack map (2017). <http://map.norsecorp.com/#/>
11. Radcliffe, J.: Cyberlaw 101: A primer on us laws related to honeypot deployments. In: Information Security Reading Room (2007)
12. Scottberg, B., Yurcik, W., Doss, D.: Internet honeypots: Protection or entrapment? In: International Symposium on Technology and Society (2002)
13. Sokol, P., Misek, J., Husak, M.: Honeypots and honeynets: Issues of privacy. EURASIP J. Inform. Secur. (2017)
14. Telekom DTAG: Fruhwarnsystem, sicherheitstacho (2017). <http://www.sicherheitstacho.eu/>
15. ZERODIUM: Zerodium exploit acquisition program (2017). <https://zerodium.com/>

Defence Against Attacks and Anonymity



SATYA: Defending Against Adversarial Attacks Using Statistical Hypothesis Testing

Sunny Raj¹(✉), Laura Pullum², Arvind Ramanathan², and Sumit Kumar Jha¹

¹ Computer Science Department, University of Central Florida, Orlando, FL, USA
{sraj, jha}@cs.ucf.edu

² Computational Science and Engineering Division, Oak Ridge National Laboratory,
Oak Ridge, TN, USA
{pullum11, ramanathana}@ornl.gov

Abstract. The paper presents a new defense against adversarial attacks for deep neural networks. We demonstrate the effectiveness of our approach against the popular adversarial image generation method DeepFool. Our approach uses Wald's Sequential Probability Ratio Test to sufficiently sample a carefully chosen neighborhood around an input image to determine the correct label of the image. On a benchmark of 50,000 randomly chosen adversarial images generated by DeepFool we demonstrate that our method *SATYA* is able to recover the correct labels for 95.76% of the images for CaffeNet and 97.43% of the correct label for GoogLeNet.

1 Introduction

Over the last few years, it has been shown that small perturbations to an input can cause machine learning algorithms to produce incorrect answers [5, 12, 16, 18]. In particular, computer implementations of vision algorithms including approaches based on deep learning have been shown to be vulnerable to such adversarial attacks. These attack approaches cover a broad spectrum from random sampling of images to the framing of an optimization problem often solved using variants of stochastic gradient descent. This knowledge of adversarial synthesis can be leveraged by an attacker to generate unwanted or malicious output from machine learning systems. Tampering with machine learning systems using adversarial attacks that are directly interacting with humans such as autonomous driving can lead to immediate catastrophic results [17]. As the adoption of machine learning systems is increasing rapidly the security and robustness of these systems gain even more importance. Given the ease with which adversarial inputs can be generated for deep learning algorithms, two questions are of natural interest:

1. Can we detect the adversarial nature of the input to a neural net?
2. Can we recover the correct results even when deep neural networks are exposed to adversarial inputs?

Table 1. *SATVA* correctly identifies 95.76% of adversarial images generated by DeepFool against the Caffe deep learning framework for 50,000 random images. The accuracy is 97.43% for adversarial versions of 50,000 randomly selected images for GoogLeNet. The accuracy for original images is within 2% of the DNN classification accuracy.

DNN	DNN accuracy on original image	<i>SATVA</i> accuracy on original image	<i>SATVA</i> accuracy on adversarial image
CaffeNet	73.76%	71.99%	95.76%
GoogLeNet	78.19%	77.97%	97.43%

In this paper, we make progress towards answering both these questions for image classification using deep neural networks. We show that the sampling of a suitably-selected neighborhood of the input image that spans two or more classes can be used to correctly classify the input image with high probability. The Sequential Probability Ratio Test (SPRT) allows our approach to adaptively sample this carefully-crafted neighborhood of the input image and decide the label of a (possibly adversarial) image in a computationally efficient manner [23]. In our experimental studies, *SATVA* is able to correctly classify 95.76% of adversarial images generated by the DeepFool system for the CaffeNet [22] deep learning framework. We are also able to correctly classify 97.43% of the adversarial images generated from the GoogLeNet [8] deep learning framework. For comparison the method shown in [7] detects DeepFool adversarial images with only 85–90% accuracy. This method detects 50% of the original non-adversarial image as adversarial. In comparison, our method detects less than 2% of non-adversarial images as adversarial. To the best of our knowledge, *SATVA*'s accuracy on adversarial images synthesized by DeepFool [12] is the highest reported in the literature so far.

The idea that sampling can act as a defense against adversarial attacks is simple and intuitive, though no concrete result of detecting adversarial examples using sampling around the space of input image has been shown in literature. In this paper, we show that simply sampling around the input image is enough to get good results. We show that *SATVA* gives us an improvement of more than 1% over this simple sampling approach. We also show that *SATVA* is more resilient to variations in the location of adversarial image.

2 Related Work

A variety of machine learning approaches, including deep learning [5] and human-crafted vision algorithms [18] such as histogram-of-gradients, have been shown to be susceptible to adversarial inputs. Small but carefully crafted perturbations in an input can cause a machine learning algorithm to produce an incorrect output. In fact, it has been observed that adversarial examples designed for one deep learning classifier can transfer to another unrelated deep learning classifier and produce incorrect results even in the second classifier [15].

2.1 Adversarial Networks

The design of algorithms for generating adversarial images has received significant attention in the machine learning [9, 14] and in software security [16] communities. A framework for generating adversarial nets using backpropagation for multiplayer perceptrons was proposed in [4], and the approach was illustrated on multiple datasets including MNIST, TFD and CIFAR-10. The approach was extended in [10] to conditional generative adversarial nets and experimental results on both MNIST and MIT Flickr 25,000 dataset have been reported. A pyramidal hierarchy of generative adversarial nets have also been used to create image models that are confused to be natural by human evaluators [1].

An interesting white-box approach to adversarial attack [20] relies on exploring the internal layers of the deep neural network representation of an image and making minimal possible perturbation to the image so that its internal representation matches a completely different natural image – thereby leading to incorrect classification of the image. This approach has the ability to trick a deep neural network to confound any image with any other chosen image through cleverly chosen perturbations, and can generate multiple adversarial examples.

Our experimental studies use the state-of-the-art DeepFool [12] algorithm to generate adversarial examples. The DeepFool algorithm is known to compute perturbations that efficiently create adversarial images for deep neural networks. To the best of our knowledge, SATYA's ability to defend against adversarial perturbations generated by DeepFool with more than 95% probability is new and has not been reported before in the literature.

2.2 Defense Against Adversarial Attacks

Virtual adversarial training [11] uses a KL-divergence based robustness metric of a model against local perturbation around a datapoint to regularize the model. This approach has been reported to work better than ordinary adversarial training on several benchmarks, including MNIST, SVHN and NORB.

Adversarial attacks have theoretically been shown to be more powerful than random noise perturbations [2, 3] specifically in the context of linear classifiers. The observation of adversarial attacks in the context of high-dimensional data has been explained by a formal proof demonstrating that robustness to random noise is \sqrt{d} times more than that to adversarial perturbations. Our work is motivated by this observation. Instead of feeding a single adversarial image as an input to a deep neural network, we sample a carefully-constructed neighborhood of the adversarial image and hence avoid making a decision on a single image.

Robust optimization has been used to increase the local stability of artificial neural networks [21], thereby making it harder to generate adversarial examples for such robust networks. They report upto 79.96% accuracy on adversarial images generated from the MNIST benchmark and about 65.01% accuracy on adversarial images generated from the CIFAR-10 benchmark. Our work is different from their approach as we do not seek to optimize the training of the network

itself but instead seek to query enough samples so as to prevent an adversarial attack on a pre-trained classifier like GoogLeNet or CaffeNet. Of course, our approach also happens to produce better experimental results with accuracies as high as 95.76% on adversarial examples for CaffeNet and 97.43% on adversarial examples for GoogLeNet.

A statistical method to detect adversarial examples has been proposed in [6], this method has been shown to work on MNIST, DREBIN and MicroRNA data with attack vectors chosen using FGSM, JSMA, SVM and DT attacks. Impressive performance of about 100% detection in certain tests have been reported. Though no method to recover the original label of the image has been shown, one thing to note is that the data sets used in this method are significantly less complex than ImageNet data set that we are using for our method. Another method for detecting adversarial perturbations has been presented in [7]. This method has been shown to work on ImageNet datasets against DeepFool perturbations. The detection probability for adversarial images has been shown to be around 85–90% for DeepFool perturbed images, though the false positive rate of identifying normal images as adversarial is around 50%. In comparison *SATYA*, has a false positive rate of less than 2%.

3 The *SATYA* Algorithm: Defending Against Adversarial Attacks

Our approach to detecting and recovering from adversarial inputs is based on the SPRT-driven sampling of a carefully-crafted neighborhood around a (possibly adversarial) input image. In Sect. 3.1, we first present an intuitive method of sampling the neighborhood of input image. In Sect. 3.2, we improve upon the intuitive method to sample in a carefully crafted subspace and discuss its advantage over Sect. 3.1. The use of Wald’s sequential probability ratio test to drive an efficient exploration of this neighborhood is discussed in Sect. 3.3. An overview of the full method is presented in Algorithm 1.

3.1 Sampling as a Defense Against Adversarial Images

A very intuitive approach for developing a defense against adversarial images would be to sample the neighborhood of adversarial images. The underlying idea is simple: *If the adversarial image happens to be adversarial only because it is carefully crafted, its neighbors may still be correctly classified by a deep neural network and hence may help void the adversarial nature of the input.* We show the arrangement of adversarial space in Fig. 1. One important point to keep in mind is the dimensionality of the input image; even though we have shown a two-dimensional space in Fig. 1, the search space of images has very high dimensionality. The number of dimensions d for an input image to CaffeNet is $227 \times 227 \times 3$, where 227 is the input dimension and 3 is the number of channels corresponding to RGB values of the input image.

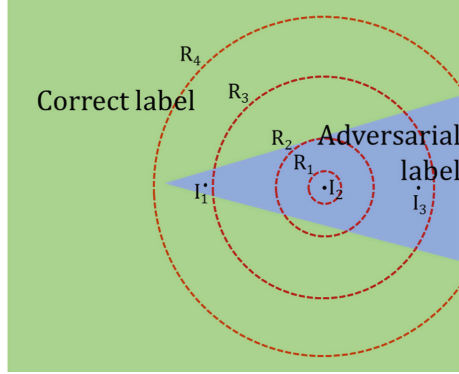


Fig. 1. The correct non-adversarial label space is shown in green, while the adversarial space is shown in blue. The adversarial images I_1 , I_2 and I_3 are located at different locations inside the adversarial space. The dotted lines denote hyperspheres with varying radii drawn with the image I_2 as the center. Sampling on a hypersphere of radius R_1 will give incorrect results while sampling on a hypersphere of radius R_4 will give correct results. (Color figure online)

For a simple sampling approach, we sample on the surface of the hypersphere centered around the input image. We use the sampling method described in [13] to generate uniformly distributed points on the surface of a d -dimensional hypersphere. To generate a random point P_i , we generate d random numbers $r_{i0}, r_{i1} \dots r_{id}$ from d independent standard normal distribution with $\mu = 0$, $\sigma = 1$ and $d = x \times y \times c$, where x and y are the dimensions of the image and c is the number of channels in the image. Let $G_i = [r_{i0} \ r_{i1} \ \dots \ r_{id}]$, then the random point P_i on the surface of d -dimensional hypersphere with radius R is given by Eq. 1. This equation is implemented by the function \mathcal{N} in Algorithm 1.

$$P_i = \frac{R}{\|G_i\|} G_i^T \quad (1)$$

If an adversarial image is deep inside the adversarial space, sampling on low radius hyperspheres will only give adversarial samples as shown by R_1 and R_2 in Fig. 1. An image like I_3 that is further inside the adversarial space will require a larger hypersphere radius to give correct samples when compared to the image I_1 . We test the accuracy of detection at various radii for 1000 sample images for both CaffeNet and GoogLeNet; we show the results in Table 2. The performance of CaffeNet is optimal for a radius of 500 units where it reaches the peak accuracy of 92.6%. The performance of GoogLeNet is optimal at 1000 units where the accuracy is 97.0%. We suspect that this difference in peak accuracy at different radii is due to the generally different performance of DeepFool on these two different networks. In the case of CaffeNet, DeepFool might be creating adversarial images closer to the boundary of the adversarial space whereas for GoogLeNet the adversarial image might be further inside the adversarial space.

Table 2. Sampling the neighborhood of images at varying sampling radii. Third and fourth columns show the percentage of image correctly classified by CaffeNet and GoogleNet. Fifth and sixth column show the average number of different labels on the hypersphere.

Index	Hyperhsphere radius	Correct percentage CaffeNet	Correct percentage GoogLeNet	Average number of labels CaffeNet	Average number of labels GoogLeNet
1	50	5.7%	6.7%	1.94	1.91
2	100	21.9%	8.0%	1.80	1.71
3	200	58.3%	33.2%	1.60	1.48
4	500	92.6%	94.0%	1.43	1.31
5	1000	90.8%	97.0%	1.36	1.27
6	1500	89.2%	96.3%	1.38	1.29
7	2000	86.4%	94.5%	1.45	1.34
8	3000	79.2%	91.0%	1.65	1.49
9	5000	64.1%	83.7%	2.31	2.06

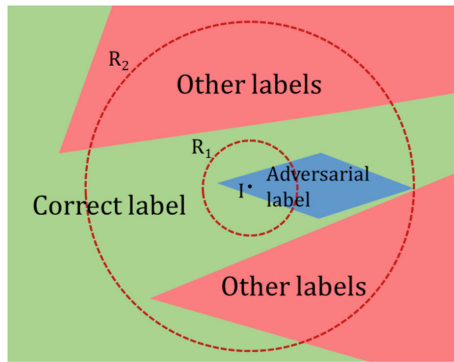


Fig. 2. Sampling around the image I . A low radius R_1 will give correct results, while a very high radius R_2 is likely to give incorrect results.

One trend that we observe in Table 2 is the decline in accuracy as the radius of the hypersphere increases beyond 1500 units. Hyperspheres with higher radii have larger volume and can accommodate samples of multiple labels as shown in Fig. 2. Multiple labels on the hypersphere can decrease the chance of the correct label having the maximum number of samples. This hypothesis is confirmed by the fact that we observe an increase in the average number of labels on the hypersphere as the radius increases from 1500 units. One natural conclusion from these experimental observations is that an algorithmic approach to defend against adversarial attacks should construct a consistent search space unaffected by the position of the adversarial image inside the adversarial space.

3.2 Constructing a Suitable Sample Space

One suitable candidate for a consistent sampling space is the neighborhood of an image that is on the boundary between the correct non-adversarial space and the adversarial space. This image is shown as I_{mid} in Fig. 3. The location of I_{mid} reduces the need for finding the optimal hypersphere radius for sampling. We present the procedure to calculate the image I_{mid} in this section. We iterate that the figures shown here are a simplification of the more complicated high dimensional space.

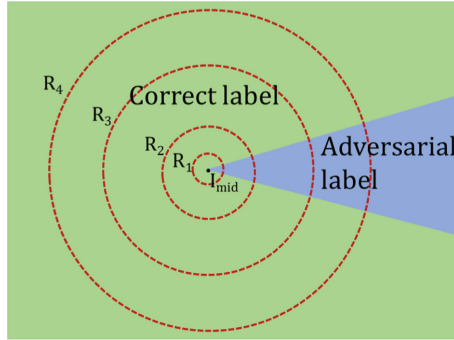


Fig. 3. Sampling around the image I_{mid} at the border of the space of correct non-adversarial label and adversarial label. Sampling on most hypersphere radii will give correct results.

Given an input image I and a deep neural network (DNN) classifier \mathcal{C} , SATYA first calculates the best classification label l_1 for the image I . It also computes the second-best classification label l_2 for the same image. For an adversarial image generate by DeepFool, the second label l_2 is the correct label. Similarly for the non adversarial image, the label l_2 is the incorrect label and is the label of the adversarial space. We generate the image I_{mid} using the gradient generated by the backpropagation function of the DNN. The error function $\mathcal{E}(I, l)$ of the classifier \mathcal{C} gives the backpropagated error at the input layer with the input image I and the correct label l . At each step j of the iteration, for an input image I^j the error $\mathcal{E}(I^j, l_2)$ of the input layer of DNN is generated by assuming l_2 as the correct label of the image. The new image I^{j+1} is generated using the update $I^{j+1} = I^j + \mathcal{E}(I^j, l_2)$. Each iteration generates an image with higher confidence of l_2 . We continue adding $\mathcal{E}(I^j, l_2)$ until at iteration k , l_2 is the highest confidence label of the image. The image I^k has the label l_2 and the image I^{k-1} has the label l_1 . The image I_{mid} is calculated by doing a binary search between the images I^k and I^{k-1} to get an image on the separating boundary of the two labels l_1 and l_2 . The algorithm then samples images on the surface of an n -dimensional hypersphere of a fixed radius around I_{mid} using the method shown in Sect. 3.1.

The results of sampling with various radii for 1000 images is shown in Table 3. For both CaffeNet and GoogLeNet, good accuracy is obtained at the smallest sampled radius of 50 units. We note that the accuracy of \mathcal{SATYA} is better than simple sampling around the adversarial image. We revisit this comparison with higher number of samples in the experimental section. We can also note that there is less variation between the accuracy at different hypersphere radii for \mathcal{SATYA} . In Table 2 the standard deviation for CaffeNet is 29.98 whereas the standard deviation in Table 3 is 10.16. Similarly for GoogLeNet, the standard deviation in Table 2 is 37.05 where as it is 4.34 for Table 3. This simple metric shows us that the results obtained from \mathcal{SATYA} is more resilient to variations in the location of the adversarial image.

Algorithm 1. \mathcal{SATYA} adversarial image classification

Input: Image I , Set of labels L , Deep Neural Network Classifier \mathcal{C} , Input layer error function of classifier \mathcal{E} , Type I/II error e , Maximum number of samples N , Indifference region $[p_0, p_1]$, Maximum number of iterations for searching middle image M , Sampling radius R

Output: Classification label for image I

$l_1 = \arg \max_{l \in L} \mathcal{C}(I, l)$ \triangleright Find best label for image I

$l_2 = \arg \max_{l \in \{L \setminus l_1\}} \mathcal{C}(I, l)$ \triangleright Find second-best label for I

$l_c = l_1, I_1 = I, I_2 = I, m = 0, n = 0, s = 0$

while $l_c \neq l_2$ **do**

$I_1 = I_2$

$I_2 = I_2 + \mathcal{E}(l_2)$

$l_c = \mathcal{C}(I_2)$

end while

\triangleright Perform binary search to compute the boundary between l_1 and l_2

$I_{mid} = \mathcal{B}(I_1, I_2)$

\triangleright Compute SPRT stopping criteria for Type I/II error

$S_{min} = \log\left(\frac{e}{1-e}\right), S_{max} = \log\left(\frac{1-e}{e}\right)$

repeat

$n = n + 1$

\triangleright Increment total number of samples

$J = \text{sample } i.i.d. \text{ from } \mathcal{N}(I_{mid}, R)$

if $\mathcal{C}(J) = l_2$ **then**

$s = s + 1$

\triangleright Increment no. of successful samples

end if

\triangleright Update Sequential Probability Ratio

$S = \log\left(\frac{p_1^s(1-p_1)^{n-s}}{p_0^s(1-p_0)^{n-s}}\right)$

until $S < S_{min}$ **or** $S > S_{max}$ **or** $n \geq N$

if $s > n - s$ **then**

print Class label: l_2

else

print Class label: l_1

end if

In our current implementation, we have only considered the top-2 labels l_1 and l_2 for deciding the correct label of the image. The limitation of top-2 labels works in the case of DeepFool as the adversarial attack algorithm works gradually towards an adversarial label and the algorithm terminates at the first instance of a wrong label thus leaving the correct label as l_2 . For implementing a top- n variant of this algorithm, a competition between the top- n labels can be organized and the winner declared as the current label.

Table 3. Sampling the neighborhood of images at varying sampling radii and percentage of those images classified correctly for 1000 images.

Index	Hypersphere radius	Correct prediction CaffeNet	Correct percentage CaffeNet	Correct prediction GoogLeNet	Correct percentage GoogLeNet
1	50	974	97.4%	970	97.0%
2	100	963	96.3%	973	97.3%
3	200	952	95.2%	974	97.4%
4	500	926	92.6%	977	97.7%
5	1000	896	89.6%	963	96.3%
6	1500	886	88.6%	956	95.6%
7	2000	858	85.8%	941	94.1%
8	3000	783	78.3%	915	91.5%
9	5000	635	63.5%	834	83.4%

3.3 Statistical Hypothesis Testing

The sampling neighborhood around the transition image I_{mid} constructed in the previous subsection is quantitatively different for different input images. For adversarial images generated from images that were correctly recognized by the DNN classifier \mathcal{C} with high-confidence, we find that the sampling neighborhood contains an overwhelming majority of images that are correctly labeled by the DNN classifier \mathcal{C} . On the other hand, the sampling neighborhood only contains a thin majority of images correctly labeled by the DNN classifier \mathcal{C} if the original image was correctly classified by the classifier with a very low confidence.

SATYA uses the Sequential Probability Ratio Test (SPRT) to adaptively sample the neighborhood constructed in the previous subsection [23]. The test rejects one of the following two hypotheses:

Null Hypothesis: \mathcal{C} assigns the label l_2 to images in the neighborhood of I_{mid} with probability more than p_1 .

Alternate Hypothesis: \mathcal{C} assigns the label l_2 to images in the neighborhood of I_{mid} with probability less than p_0 .

The user specifies an indifference region $[p_0, p_1]$, Type I/II error e and the maximum number of samples to be obtained N . SPRT then samples the neighborhood recording the total number of images sampled (n) and the number of images (s) labeled by the classifier as l_2 . Using these inputs, SPRT computes the likelihood ratio:

$$\frac{p_1^s(1-p_1)^{n-s}}{p_0^s(1-p_0)^{n-s}}$$

If the likelihood ratio falls below a threshold derived from the Type I/II error, SPRT rejects the null hypothesis. If the likelihood ratio exceeds a threshold, SPRT rejects the alternate hypothesis.

If the probability of sampling an image with the label l_2 is more than p_1 , the algorithm will produce this label with probability $1 - e$. For example, if $p_1 = 0.51$ and $e = 0.01$, our algorithm produces this label with 99% accuracy if the sampling neighborhood has at least 51% correctly labeled images. Of course, greater accuracy can be achieved by reducing the Type I/II error and by setting the value of p_1 to $0.5 + \epsilon$ for a small $\epsilon > 0$. However, this comes at the expense of a larger number of samples needed to reach a conclusion.

In Figs. 6 and 7, we show how the number of samples required by our statistical hypothesis testing algorithm can vary widely among different input images. In particular, adversarial images that are generated from images for which the DNN classifier \mathcal{C} was making a correct but low-confidence prediction tend to require larger number of samples. Figure 8 shows that the number of samples required to disambiguate an adversarial image generated from an original image classified with confidence between 0.4 and 0.6 is more than 5 times the number of samples required for a high-confidence (0.8–1.0) prediction. Thus, the SPRT-driven adaptive sampling is critical to ensure an efficient performance of *SATYA*.

4 Experimental Results

We evaluated *SATYA* on 50,000 images from the ILSVRC2013 training dataset [19] for both the CaffeNet [22] and the GoogLeNet [8] deep learning frameworks. Adversarial versions of these images were created using the state-of-the-art DeepFool [12] adversarial attack system. In our experiments, we have used the following parameters for the *SATYA* algorithm: Type I/II error $e = 0.000001$, maximum number of samples $N = 2000$, indifference regions $p_0 = 0.47$ and $p_1 = 0.53$, maximum number of iterations for searching the transition image I_{mid} $M = 500$ and hypersphere radius for sampling $R = 200$ units. Our experiments were carried on a 16 GB Intel(R) Core(TM) i7-4770K CPU @ 3.50 GHz workstation with an NVIDIA GeForce GTX 780 GPU. Our experimental evaluation has four goals:

1. How well does *SATYA* perform on the adversarial images generated by the DeepFool algorithm working with CaffeNet and on adversarial images generated for GoogLeNet?

2. What is the impact of *SATYA* on original non-adversarial benchmarks?
3. How is the runtime performance of *SATYA*?
4. Does the runtime of our approach vary significantly depending on the image being investigated?

4.1 Accuracy on Adversarial Images

SATYA correctly identifies more than 95% of all the adversarial images generated for both the CaffeNet deep neural network and the GoogLeNet deep learning framework. The results for the execution on 50,000 ILSVRC images is shown in Table 1. The hypersphere radius for sampling was taken to be 200 units for both CaffeNet and GoogLeNet. The accuracy of detection of the correct label was 95.76% for CaffeNet and 97.43% for googleNet.

To compare *SATYA* with simple sampling approach around input adversarial image we ran the benchmark on the same set of 10,000 random ILSVRC images. The hypersphere radius for best accuracy was 500 units for CaffeNet and 1000 units for GoogLeNet. We show the experimental results in Table 4. We can see that an accuracy gain of 2.12% for CaffeNet and 1.14% for googleNet was achieved using *SATYA*.

Table 4. Accuracy of *SATYA* compared to simple sampling approach for a sample set of 10,000 ILSVRC images. The hypersphere radius for CaffeNet was taken to be 500 units and for googleNet it was taken to be 1000 units.

Benchmark	Simple sampling	<i>SATYA</i>	Accuracy gain
CaffeNet	93.40%	95.52%	2.12%
GoogLeNet	96.55%	97.69%	1.14%

4.2 Accuracy on Original Unperturbed Images

While *SATYA*'s performance on adversarial images is very good, it would not be a useful algorithm if its performance on non-adversarial images turned out to be poor. *SATYA* performs very well even on original non-adversarial images. The accuracy of CaffeNet on the original non-adversarial image is 73.76% while the accuracy of *SATYA* on CaffeNet is 71.99%. The accuracy of GoogLeNet on the original non-adversarial image is 78.19% while the accuracy of *SATYA* on original image is 77.54%. We can see that the underlying detection algorithms perform only slightly better than *SATYA*. The breakup for images correctly and incorrectly classified by CaffeNet is given in Table 5. We can see that *SATYA* correctly classifies 7.53% of image originally incorrectly classified by CaffeNet and 2.45% of image originally incorrectly classified by GoogLeNet.

Table 5. $SATVA$ correctly identifies 95.10% of the 36882 original images correctly recognized by the Caffe deep learning framework (called CaffeNet+ here) and recognizes 7.53% of the 13118 original images not correctly recognized by Caffe (called CaffeNet– here). The accuracy is 98.62% for original versions of 39093 correctly recognized images and 2.45% for original version of 10907 images not correctly recognized by GoogLeNet. Here, the two classes are referred as GoogLeNet+ and GoogLeNet– respectively.

Benchmark	Prediction		Correct
	Wrong	Correct	Percentage
CaffeNet+	1808	35074	95.10%
CaffeNet–	12199	919	7.53%
GoogLeNet+	540	38553	98.62%
GoogLeNet–	10646	261	2.45%

4.3 Runtime Performance

The time required by $SATVA$ to analyze both original and adversarial images for CaffeNet is shown in Fig. 4 and for GoogLeNet is shown in Fig. 5. The runtime performance of $SATVA$ is acceptable for high-fidelity applications like cyber-physical systems. An overwhelming majority of the images were analyzed by $SATVA$ within 4s. We should note that the availability of enough parallel computational resource can be used to speed up $SATVA$ to match the runtime performance of the underlying classifier by using massively parallel calls from $SATVA$ to the underlying classifier such as CaffeNet or GoogLeNet.

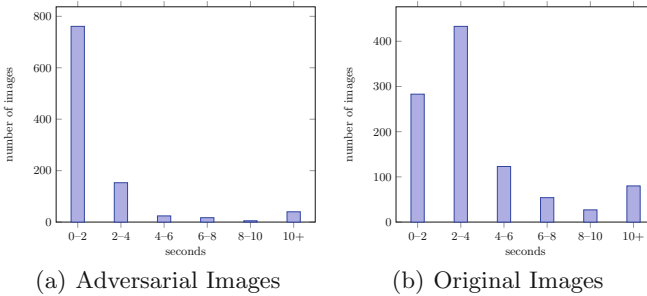


Fig. 4. Time taken by $SATVA$ for predicting the label of adversarial and ordinary images used in CaffeNet.

$SATVA$ correctly recognizes about 91% of the adversarial images and 70% of the original images within 4s on our single GPU machine with only sequential calls to the DNN CaffeNet classifier. The worst case runtime of our approach on adversarial images is 22s for the CaffeNet deep neural network.

The performance of $SATVA$ on images from the GoogLeNet is qualitatively similar to the results on CaffeNet. About 98% of the adversarial images and

75% of the original images can be analyzed within 4s. The worst case runtime of *SATYA* on adversarial images is 27s for GoogLeNet.

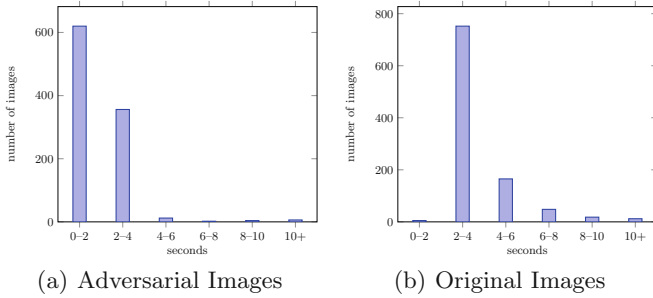


Fig. 5. Time taken by *SATYA* in calculating the label for images used in GoogLeNet. The performance of both adversarial and original images have been analyzed.

4.4 Dependence of Performance on the Confidence of Classification

The performance of *SATYA* depends upon the number of images sampled by the sequential probability ratio test. In Fig. 6, we illustrate the number of samples required by *SATYA* while analyzing original and adversarial images for CaffeNet. About 50% of the adversarial images can be analyzed by studying only 200 samples. Similarly, about 55% of the original images can be classified by analyzing only 200 samples. Only a small fraction of images require more than 1,000 samples.

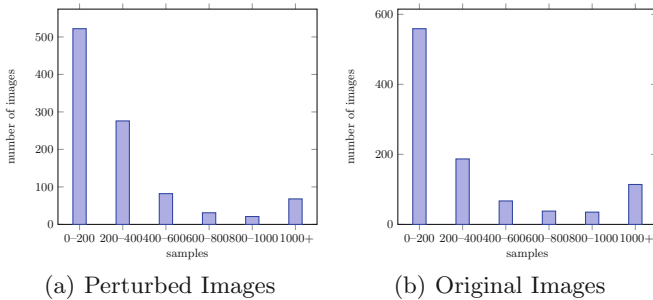


Fig. 6. Number of samples tested to determine the label of an original as well as an adversarial image for CaffeNet.

Figure 7 shows the number of samples required by original and adversarial images for the GoogLeNet deep learning framework. About 80% of the perturbed images and more than 75% of the original images were analyzed by sampling fewer than 200 samples. In the light of this variation in number of samples for

a small fraction of the images, a natural question that arises is the source of this variability. Using Fig. 8 we establish an empirical relationship between the number of samples required to disambiguate an image and the confidence with which the classifier assigns a label to the image. If CaffeNet is able to correctly label an image with a confidence of 0.6 or more, *SATVA* only needs 500 or fewer samples to assign a label to an adversarial input created using this image.

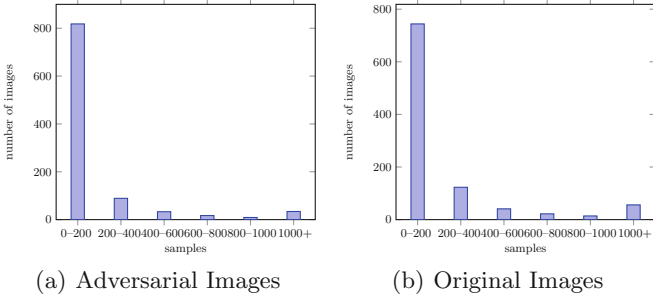


Fig. 7. Number of samples tested to determine the label of an original as well as an adversarial image for GoogLeNet.

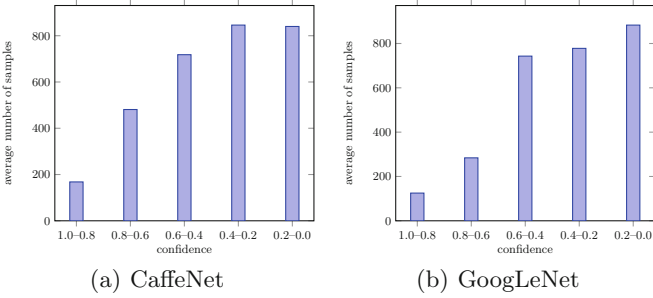


Fig. 8. The average number of samples needed to classify an image increases as the classifier’s confidence in the label of the image decreases.

We observed qualitatively similar results for adversarial images generated for the GoogLeNet deep learning classifier. Adversarial inputs generated using images that were assigned high-confidence labels by GoogLeNet (0.6 or more) were easily labeled by *SATVA* using fewer than 300 samples. The fact that *SATVA* is able to recover the labels of adversarial inputs generated from high-confidence images is extremely desirable. Such a performance behavior implies that high-confidence predictions from a classifier may be difficult to distort in a manner where they cannot be readily recovered by *SATVA* and other defensive approaches.

5 Conclusion and Future Work

SATYA provides a highly effective defense against adversarial attacks. In our experimental evaluation, more than 95% of adversarial images generated by DeepFool against CaffeNet deep neural network and against GoogLeNet deep learning framework are correctly recognized by our approach. *SATYA* also performs comparably to the underlying image detection system for non-adversarial images. When compared to simple sampling approach, *SATYA* gives better accuracy and it more resilient to variations in the position of the adversarial input image.

Several natural avenues for future research are open. A theoretical explanation of the success of our approach perhaps using manifolds will help clarify the interaction of deep neural networks and high-dimensional big data. Practical efforts towards parallelizing *SATYA* would help make the tool deployable in real-time settings.

Acknowledgments. The authors would like to thank the US Air Force for support provided through the AFOSR Young Investigator Award to Sumit Jha. The authors acknowledge support from the National Science Foundation Software & Hardware Foundations #1438989 and Exploiting Parallelism & Scalability #1422257 projects. This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-16-1-0255. This research was partially supported by ORNL’s Laboratory Directed Research and Development (LDRD) proposal 7899. This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy.

References

1. Denton, E.L., Chintala, S., Fergus, R., et al.: Deep generative image models using a Laplacian pyramid of adversarial networks. In: Advances in Neural Information Processing Systems, pp. 1486–1494 (2015)
2. Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers’ robustness to adversarial perturbations. arXiv preprint [arXiv:1502.02590](https://arxiv.org/abs/1502.02590) (2015)
3. Fawzi, A., Fawzi, O., Frossard, P.: Fundamental limits on adversarial robustness. In: Proceedings of ICML, Workshop on Deep Learning (2015)
4. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in Neural Information Processing Systems, pp. 2672–2680 (2014)
5. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint [arXiv:1412.6572](https://arxiv.org/abs/1412.6572) (2014)
6. Grosse, K., Manoharan, P., Papernot, N., Backes, M., McDaniel, P.D.: On the (statistical) detection of adversarial examples. CoRR abs/1702.06280 (2017). <http://arxiv.org/abs/1702.06280>
7. Hendrik Metzen, J., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. ArXiv e-prints, February 2017

8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) *Advances in Neural Information Processing Systems* 25, pp. 1097–1105. Curran Associates Inc., New York (2012). <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
9. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial machine learning at scale. arXiv preprint [arXiv:1611.01236](https://arxiv.org/abs/1611.01236) (2016)
10. Mirza, M., Osindero, S.: Conditional generative adversarial nets. arXiv preprint [arXiv:1411.1784](https://arxiv.org/abs/1411.1784) (2014)
11. Miyato, T., Maeda, S.I., Koyama, M., Nakae, K., Ishii, S.: Distributional smoothing with virtual adversarial training. arXiv preprint [arXiv:1507.00677](https://arxiv.org/abs/1507.00677) (2015)
12. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: DeepFool: a simple and accurate method to fool deep neural networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582 (2016)
13. Muller, M.E.: A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM* **2**(4), 19–20 (1959). <http://doi.acm.org/10.1145/377939.377946>
14. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 427–436 (2015)
15. Papernot, N., McDaniel, P., Goodfellow, I.: Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint [arXiv:1605.07277](https://arxiv.org/abs/1605.07277) (2016)
16. Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z.B., Swami, A.: The limitations of deep learning in adversarial settings. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 372–387. IEEE (2016)
17. Raj, S., Ramanathan, A., Pullum, L.L., Jha, S.K.: Testing autonomous cyber-physical systems using fuzzing features derived from convolutional neural networks. In: *ACM SIGBED International Conference on Embedded Software (EMSOFT)*. ACM, Seoul (2017)
18. Ramanathan, A., Pullum, L.L., Hussain, F., Chakrabarty, D., Jha, S.K.: Integrating symbolic and statistical methods for testing intelligent systems: applications to machine learning and computer vision. In: *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 786–791. IEEE (2016)
19. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis. (IJCV)* **115**(3), 211–252 (2015)
20. Sabour, S., Cao, Y., Faghri, F., Fleet, D.J.: Adversarial manipulation of deep representations. arXiv preprint [arXiv:1511.05122](https://arxiv.org/abs/1511.05122) (2015)
21. Shaham, U., Yamada, Y., Negahban, S.: Understanding adversarial training: increasing local stability of neural nets through robust optimization. arXiv preprint [arXiv:1511.05432](https://arxiv.org/abs/1511.05432) (2015)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. *CoRR abs/1409.4842* (2014). <http://arxiv.org/abs/1409.4842>
23. Wald, A.: *Sequential Analysis*. Wiley, Hoboken (1947)



Attack Graph-Based Countermeasure Selection Using a Stateful Return on Investment Metric

Gustavo Gonzalez-Granadillo¹, Elena Doynikova^{2,3}, Igor Kotenko^{2,3},
and Joaquin Garcia-Alfaro⁴(✉)

¹ Atos Research and Innovation, Cybersecurity Laboratory, Barcelona, Spain

² St. Petersburg Institute for Informatics and Automation (SPIIRAS),
Saint Petersburg, Russia

³ Information Technologies, Mechanics and Optics (ITMO) University,
Saint Petersburg, Russia

⁴ Télécom SudParis, CNRS SAMOVAR, Paris-Saclay University, Paris, France
garcia_a@telecom-sudparis.eu

Abstract. We propose a mitigation model that evaluates individual and combined countermeasures against multi-step cyber-attack scenarios. The goal is to anticipate the actions of an attacker that wants to disrupt a given system (e.g., an information system). The process is driven by an attack graph formalism, enforced with a stateful return on response investment metric that optimally evaluates, ranks and selects appropriate countermeasures to handle ongoing and potential attacks.

1 Introduction

Network attacks are frequently represented as attack graphs, in order to identify the paths taken by an attacker in the exploitation of a given series of vulnerabilities, as well as to analyze all possible countermeasures that could be implemented to mitigate the attack [1, 12]. To compute exhaustive lists of possible attack scenarios, and to select the most effective countermeasures, attack graphs must rely on quantitative metrics that may base their analysis in cost-sensitive parameters.

With the above challenge in mind, we present the integration of a stateful return on response investment metric to the attack graph formalism presented in [2, 7]. The resulting combination allows to evaluate, rank and select optimal countermeasures based on complementary assessment functions (e.g., from both financial and security dimensions). The new metric is evaluated at each state of the system while considering the already deployed countermeasures and effects of adding or suppressing other security actions. Our contributions can be summarized as follows. We provide a network security model that evaluates individual and combined countermeasures against complex attack scenarios, in order to anticipate the actions of an attacker that wants to disrupt the security of a given system. The same process dynamically evaluates multiple countermeasure candidates

while considering restrictions and inter-dependency among them. As a result, the optimal set of countermeasures is proposed and enforced over the system.

Paper Organization – Sect. 2 provides related work. Section 3 presents our construction. Section 4 concludes the paper.

2 Related Work

Kheir et al. [6] propose a process for the selection of security countermeasures by combining a service dependency framework and a cost-sensitive metric. The solution provides a systematic solution to applying policy rules while minimizing configuration changes and reducing resource consumption. Samarji et al. [11] combines a graph theoretic-solution and *situation calculus* to automatically generate mitigation graphs. Lippmann et al. [8] and Poolsappasit [10] use attack graph formalism to implement preventive and reactive countermeasures against vulnerability exploitation, accordingly. Martinelli and Santini [9] suggest the use *argumentation logic* to automate response reasoning under system attacks. The use of *argumentation logic* adapts well to problems where multiple causes for a specific anomalous behavior are possible, and multiple countermeasures can be taken to mitigate the problem. The manipulation of this reasoning process comes with a cost in terms of the chosen metrics.

With regard to the aforementioned contributions, the approach presented in this paper may estimate the risk of simultaneous attacks against the system, and compute the cost of the final decisions by acting on the decision process itself, as well as, evaluate the impact of combined responses over dependent services. It builds over the attack graph formalism presented by Doynikova and Kotenko in [2, 7], complemented with a cost-sensitive metric that extends the work by Gonzalez et al. in [4, 5]. The resulting formalism is used as an automated response selection mechanism, that anticipates forecasted steps of an attacker that aims at disrupting the security of a given system. The cost-sensitive metric builds upon the Return on Response Investment (RORI) index, initially proposed by Kheir et al. [6] as an extension of the Return On Security Investment (ROSI) index [13]. The metric provides a common reference to compare different countermeasures. Precise information about the computation of each specific parameter of the RORI index can be found in [4, 5].

3 Our Construction

We present a countermeasure selection formalism that connects attack actions on the basis of pre and post conditions w.r.t. vulnerability exploitations and Bayesian probabilities. It extends previous contributions presented in [2, 4, 7]. Its distinctive features are as follows: an opportunity of automated attack graph generation using network configuration and publicly available indexes for vulnerabilities; joint consideration of the attack probabilities and attack impact for the system assets; consideration of the attacker profile; connection with security

events; preventive and reactive countermeasure selection. The goal is to represent, anticipate and handle attack actions performed by an attacker targeting a given system. We start with the core definitions. Then, we move to presenting the operation modes (e.g., preventive and reactive selection of countermeasures).

Definition 1 (Attack Graph). *A graph $G = (S, L, \tau, P_c)$ where S contains the nodes of the graph (i.e., the set of attack actions), L represents the set of links between actions (s.t. $L \subseteq S \times S$), τ the relation between attack actions, and P_c the discrete local conditional probability distributions.*

Definition 2 (Attack Action). *A 5-tuple $S = (H, V, S_c, S_t, P_r)$, where H identifies the attacked host, V the exploited vulnerability, S_c the process used by the attacker to get information about the host, and P_r the probability that the attack action is in state S_t ($P_r \in [0, 1]$).*

3.1 Preventive Mode, Prior Mapping of System Attacks

By combining Definitions 1 and 2, we can now represent all the possible attack actions (e.g., vulnerability exploitations and information gathering) and transitions between the actions of a multi-step attack scenario [1, 12]. In addition, stateful information is represented under the action states in S_t . This enables the use of a preventive mode, prior detecting precise attack instances, to already evaluate both local and global levels of risk in the system. The goal is to apply an initial set of preventive countermeasures to reduce the global level of risk in the system. Further countermeasures, selected under a reactive mode, e.g., once precise attacks have been detected and mapped to the attack graph, are presented later in Sect. 3.2. Next, we provide definitions and processes used under the preventive mode.

Definition 3 (Preventive Risk Calculation). *Under the preventive mode, a precise level of risk is associated to each node of the attack graph. It relies on a product combination of two main parameters: AttackImpact \times AttackPotentiality.*

The value of the AttackImpact parameter (cf. Eq. 1) is a linear combination of potential damages in terms of confidentiality, integrity and availability (denoted in Eq. 1 as cImpact, iImpact, aImpact) of the asset in case of exploitation of vulnerabilities considering CVSS indexes [3]; as well as the criticality of such assets in terms of confidentiality, integrity and availability (denoted as cCrit, iCrit, aCrit in Eq. 1).

$$\text{AttackImpact} = (\text{cCrit} \times \text{cImpact}) + (\text{iCrit} \times \text{iImpact}) + (\text{aCrit} \times \text{aImpact}) \quad (1)$$

The AttackPotentiality parameter refers to the vulnerability probability associated to each node of the graph. It is computed by using a total probability formula, considering both a local vulnerability probability p , and a conditional probability P_c that considers all the possible states of its ancestors P_a . If compromising a node requires to compromise all the parent nodes, then P_c is set

to zero when it exists an S_i in P_a whose exploitation state is marked as *False*; otherwise, P_c equals p . If compromising a node requires to compromise at least one parent node, then P_c is set to zero when $\forall S_i \in P_a$ the exploitation state is marked as *False*; otherwise, P_c equals p . The value of p is computed as follows:

$$p = \begin{cases} 2 \times \text{AccessVector} \times \text{AccessComplexity} \times \text{Authentication} & (\text{root nodes}) \\ 2 \times \text{AccessComplexity} \times \text{Authentication} & (\text{other nodes}) \end{cases} \quad (2)$$

where *AccessVector*, *AccessComplexity*, and *Authentication* are extracted from the CVSS indexes [3] associated to the list of vulnerabilities defined for each node, and normalized between 0 and 1, using the 2 factor in Eq. 2. The global estimation of the risk level of an attack sequence is defined as the combination of the minimum probability of the attack nodes and the maximum impact.

Based on the combination of *AttackImpact* and *AttackPotentiality*, we can now conduct a selection of countermeasures for those nodes of the graph with a risk level that exceeds a predefined threshold. The process is conducted by using a countermeasure selection index in terms of *Efficiency*, *Cost* and *Collateral Damages* associated with each countermeasure (or combination of countermeasures). The value of such an index can directly be obtained by using the *RORI* metric (cf. Refs. [4,5]).

The process (summarized in Fig. 1) aims at maximizing the countermeasure selection index for each node of the graph. In turn, this leads to maximizing the reduction of risk as a whole. First, the countermeasures with zero-cost expenses are implemented (Step 1). A determination is made on whether or not there are still uncovered nodes (Step 2), so that countermeasures that impact over all the security properties are sorted according to their impact area (Step 2a) and a countermeasure selection index is calculated accordingly (Step 3). The measure that affects the largest number of the graph nodes and properties is selected, the next countermeasures are selected according to the largest mismatch of the covered nodes. If there are countermeasures that affect the same number of nodes, then multiple lists are generated (Step 2a) and the following steps are performed for all lists (the list with maximum countermeasure selection index is selected). If there are countermeasures that affect the same nodes, then multiple countermeasures are added on the same level of the list. Countermeasures that maximize the selection index are selected from the list of countermeasures that impact all the security properties on each level (i.e., confidentiality, integrity and availability).

If there are still uncovered nodes (Step 4), then countermeasures that impact two or less security properties are sorted similarly to the first list, starting from the measure that impacts the largest number of the not covered nodes (Step 4a). Similar rationale as in the first case is considered to compute the countermeasure selection index (Step 3). That is, if there are still nodes under risk (Step 4a), then countermeasures that impact separate vulnerabilities are selected (Step 5). In the end, the list with the maximum countermeasure selection index is selected and enforced, to conclude the process (Step 6).

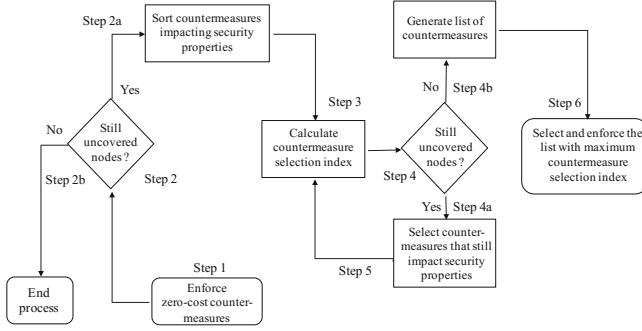


Fig. 1. Workflow of the preventing countermeasure selection process

3.2 Reactive Mode, Posteriori to the Mapping of System Attacks

Under the reactive mode, new countermeasures are selected and activated to stop the propagation of ongoing attacks. On the basis of real instances of detected security violations, a priori and a posteriori steps of an attacker are mapped, and the level of risks of the attack-graph nodes is updated. The process undertakes the phases defined below.

Definition 4 (Attack Mapping). *It follows an event model E_i to process security incidents and responses under the reactive mode, such that E_i is a 3-tuple (T_i, H_i, T_{ei}) , where T_i is the event fixing time; H_i is the event fixing host; and T_{ei} is the event type. Events are mapped on the attack graph considering the event fixing host H_i . Graph nodes that correspond to the compromised host H_i are outlined. Then, considering event type T_{ei} (e.g., security properties violation or illegitimate access) attack graph nodes that have appropriate post-conditions are selected.*

Definition 5 (Risk Update). *Mapping the security event on the attack graph results in recalculation of the risk levels for the attack sequences that go through the compromised node, considering new attack probability values. The probability for the previous nodes is recalculated using Bayes theorem, whereas for the next nodes we use the formula of total probability considering that the state of the compromised node is changed to True. The previous attacker steps are defined on the basis of the maximum probability change for the previous graph nodes. The attacker skill level is defined according to the maximum CVSS access complexity of these steps. The attacker skill level asl is used for the recalculation of the local probability for the next graph nodes as depicted in the following equation*

$$p = \begin{cases} 2 \times \text{AccessVector} \times \frac{\text{AccessComplexity} + asl}{2} \times \text{Authentication} & \text{(root nodes)} \\ 2 \times \frac{\text{AccessComplexity} + asl}{2} \times \text{Authentication} & \text{(other nodes)} \end{cases}$$

where the 2 and $\frac{1}{2}$ factors are used in the above equations in order to get medium values from access complexity and attacker skills, which results into a probability value from 0 to 1.

Based on the aforementioned mapping and risk update processes, a reactive selection of countermeasures can now be conducted, whenever an attack reported by the system increases the accepted level of risk for some nodes. The main difference between the preventive and reactive mode relies on the mapping of real instances of attacks identified in the system. Some countermeasures may be selected during the preventive phase, but only enforced during the reactive phase (e.g., software tokens that can be used to enable multi-factor authentication). This parameter and some others (i.e., affected vulnerability, impact area, impact type, affected security properties) are specified in the countermeasure model. The set of the available countermeasures is added to the database before the countermeasure selection process. The set of the available countermeasures in the reactive mode depends on the countermeasures set selected during the preventive mode. To conduct the reactive countermeasure selection process, the RORI metric proposed in [4,5] is extended towards a new Stateful Return On Response Investment Metric (hereinafter denoted as StRORI), presented in the sequel.

3.3 Stateful Return on Response Investment (StRORI)

We propose an improvement in the computation of the parameters composing the formula in [4,5], so that the new metric considers the state at which the RORI evaluation is performed. We assume a dynamic security monitoring process, where detection tools are permanently inspecting system and network events, in order to identify attack instances. To ease the presentation of the StRORI metric, we assume a discrete monitoring system that based on temporal snapshots. Each snapshot provides a list with the different nodes affected in the attack scenario, as well as all the remainder security parameters. The evaluation process is assumed to be unique for each evaluation run.

Figure 2 depicts a simple case with two transitions (i.e., from T_0 to T_1 , and from T_1 to T_2). In the initial state of the system (T_0) we assume that no countermeasure from the authorized mitigation action list has been deployed. At T_0 we perform the RORI evaluation with two candidates (e.g., C_1 , C_2) and we have three possible lists of countermeasures: (i) add C_1 (i.e., $L_{01} = \{+C_1\}$); (ii) add C_1 and C_2 (i.e., $L_{02} = \{+C_1 + C_2\}$); (iii) No operation, meaning that no mitigation action must be implemented (i.e., $L_{03} = \{\}$). In case the RORI index indicates the best action is to implement L_{01} , we implement C_1 and the state changes to T_1 .

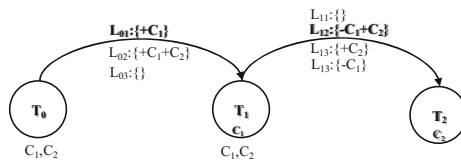


Fig. 2. Transition process in the stateful RORI evaluation

At T_1 , we perform a new snapshot of the system that considers the number of active nodes and updates the system's configuration (e.g., consider previously implemented countermeasures). The RORI index is performed at this state by evaluating all authorized mitigation actions (even those already implemented in the system) to find the best list of countermeasures. Assuming that we evaluate C_1 and C_2 , we will have four possible lists: (i) add C_1 , meaning that no action must be performed since C_1 is already implemented (i.e., $L_{11} = \{\}$); (ii) add C_2 , meaning that C_1 must be uninstalled in order to install C_2 (i.e., $L_{12} = \{-C_1 + C_2\}$); (iii) add C_1 and C_2 , meaning that only C_2 will be added since C_1 is already implemented (i.e., $L_{13} = \{+C_2\}$); and (iv) no operation, meaning that C_1 must be uninstalled since no mitigation action must be implemented (i.e., $L_{13} = \{-C_1\}$). In case the RORI index at T_1 indicates the best action is to implement L_{12} , we must uninstall C_1 and install C_2 and the state changes to T_2 . The process is repeated for a new snapshot of the system. A complete methodology for computing each parameter of the RORI metric, and related processes, is available in [4].

3.4 Validation of the Approach

The countermeasure selection process discussed in Sect. 3.3 allows extending the graph-driven selection process previously presented in [2, 7] by using the new countermeasure coverage areas provided by the StRORI metric. Such areas shall be computed for all the available countermeasures as soon as new attack instances are identified. Each state of the attack graph after a new attack event is processed leads to the transition state depicted in Fig. 2. Countermeasure coverage is used to update those attack graph nodes whose risk level exceeds a predefined threshold.

Figure 3 shows a sample attack graph generated by our proposal. Sample attack graph representation generated by a proof-of-concept prototype. Low risk nodes are depicted in gray, medium risk nodes are depicted in yellow. High and critical risk level nodes that require preventive countermeasures are represented with orange and red colors, accordingly. The first security incident is generated as a result of the detection of a web-server vulnerability exploitation. After the processing of the security incident the next nodes are included to the list for the countermeasure selection as soon as risk levels for these nodes exceed the threshold: nodes that correspond to the Web server 1; nodes that correspond to the Web server 2; and nodes that correspond to the DB Server (Fig. 3). For example we review the next countermeasures: shutdown service/host (EF = 10%, COV = 1, ALE = 3000, ARC = 80, AIV = 30000); enable/disable additional firewall rules (EF = 80%, COV = 0, 7, ALE = 3000, ARC = 200, AIV = 30000); block suspicious connection (EF = 80%, COV = 1, ALE = 3000, ARC = 0, AIV = 30000); block ports/IP addresses (EF = 80%, COV = 1, ALE = 3000, ARC = 80, AIV = 30000). Resulted StRORI index for the countermeasures: StRORI (shutdown service/host) = 0.7; StRORI (enable/disable additional firewall rules) = 4.9; StRORI (block suspicious connection) = 8; StRORI (block ports/IP addresses) = 7.7. The selected

countermeasures considering the maximum StRORI index: block suspicious connection. Further details and views of the attack graphs are available on-line at <http://j.mp/stRORI>.

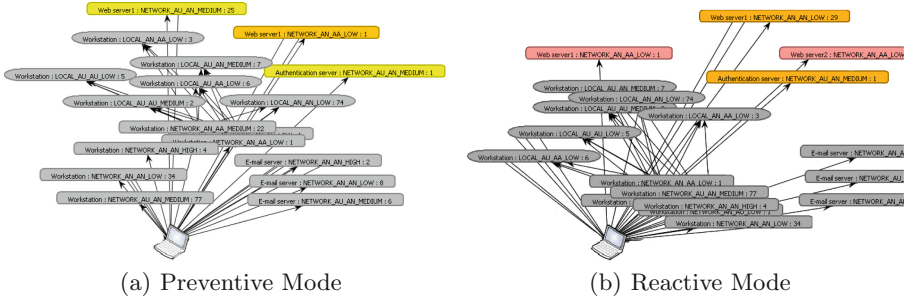


Fig. 3. Sample attack graph representation generated by our proof-of-concept prototype. Low risk nodes are depicted in gray, medium risk nodes are depicted in yellow. High and critical risk level nodes that require preventive countermeasures are represented with orange and red colors, accordingly. Further details and views of the attack graphs are available on-line at <http://j.mp/stRORI> (Color figure online)

3.5 Discussion

The main advantages of our ongoing construction are the following. We use a cost-sensitive metric to evaluate response goodness of single and combined actions against individual and multiple attack scenarios. The approach allows to rank and select the most suitable countermeasure or group of them against a given attack in a particular state of the system. The approach provides a response relative to the size of the infrastructure, which allows to compare the evaluation results of different systems regardless of their size. The model allows to handle the case of selecting no countermeasure, which results into a value of zero, meaning that no gain is expected if no solution is implemented. It also considers restrictions and conflicts among countermeasures (e.g., mutually exclusive, partially or totally restrictive countermeasures).

In addition, the proposed approach considers interdependence among countermeasures (i.e., how the application of a countermeasure affects the effectiveness of others). We, therefore, consider the impact of adding, modifying and/or suppressing a series of countermeasures previously deployed or enabled in different parts of the system.

In terms of limitations, we can observe that a great level of accuracy is required in the estimation of the different parameters of our construction. This is overcome by the use of a risk assessment methodology that considers relative values on all the elements composing the StRORI index.

4 Conclusion

We have proposed a mitigation security model that evaluates individual and combined countermeasures against multi-step attack scenarios. The process is driven by an attack graph formalism, enforced with a stateful return on response investment metric. The resulting construction optimally evaluates, ranks and selects appropriate countermeasures to handle the evolution of system risks. The approach provides preventive mitigation, prior identification of system attacks; and reactive mitigation, once attacks instances have been mapped to the attack graph. Future work will concentrate on a more thorough analysis of the approach presented in this paper towards near-continuous time dimensional domains.

Acknowledgments. E. Doynikova and I. Kotenko acknowledge support from the Russian Science Foundation under grant number 15-11-30029. G. Gonzalez-Granadillo and J. Garcia-Alfaro acknowledge support from the European Commission under grant number 610416 (PANOPTESSEC project).

References

1. Cuppens, F., Autrel, F., Bouzida, Y., Garcia, J., Gombault, S., Sans, T.: Anti-correlation as a criterion to select appropriate counter-measures in an intrusion detection framework. *Ann. Telecommun.* **61**(1), 197–217 (2006)
2. Doynikova, E., Kotenko, I.: Countermeasure selection based on the attack and service dependency graphs for security incident management. In: Lambrinouidakis, C., Gabillon, A. (eds.) *CRISIS 2015*. LNCS, vol. 9572, pp. 107–124. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31811-0_7
3. Forum of Incident Response and Security Teams. Common vulnerability scoring system v3.0 specification document. Technical paper, version: release20170402. Accessed July 2017
4. Gonzalez-Granadillo, G., Belhaouane, M., Debar, H., Jacob, G.: RORI-based countermeasure selection using the OrBAC formalism. *Int. J. Inf. Secur.* **13**(1), 63–79 (2014)
5. Gonzalez-Granadillo, G., Garcia-Alfaro, J., Alvarez, E., El-Barbori, M., Debar, H.: Selecting optimal countermeasures for attacks against critical systems using the attack volume model and the RORI index. *Comput. Electr. Eng.* **47**, 13–34 (2015)
6. Kheir, N., Cuppens-Boulahia, N., Cuppens, F., Debar, H.: A service dependency model for cost-sensitive intrusion response. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) *ESORICS 2010*. LNCS, vol. 6345, pp. 626–642. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15497-3_38
7. Kotenko, I., Chechulin, A.: Computer attack modeling and security evaluation based on attack graphs. In: 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), vol. 2, pp. 614–619. IEEE (2013)
8. Lippmann, R.P., Ingols, K., Scott, K.P.C., Kratkiewicz, K., Artz, M., Cunningham, R.: Validating and restoring defense in depth using attack graphs. In: Military Communications Conference (MILCOM 2006), pp. 1–10. IEEE (2006)

9. Martinelli, F., Santini, F.: Debating cybersecurity or securing a debate? In: Cuppens, F., Garcia-Alfaro, J., Zincir Heywood, N., Fong, P.W.L. (eds.) FPS 2014. LNCS, vol. 8930, pp. 239–246. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-17040-4_15
10. Poolsappasit, N., Dewri, R., Ray, I.: Dynamic security risk management using Bayesian attack graphs. *IEEE Trans. Dependable Secure Comput.* **9**(1), 61–74 (2012)
11. Samarji, L., Cuppens, F., Cuppens-Boulahia, N., Kanoun, W., Dubus, S.: Situation calculus and graph based defensive modeling of simultaneous attacks. *CSS* **8300**, 132–150 (2013)
12. B. Schneier. *Modelling security threats*. Dr. Dobbs J. (1999)
13. Sonnenreich, W., Albanese, J., Stout, B.: Return on security investment (ROSI)-a practical quantitative model. *J. Res. Pract. Inf. Technol.* **38**(1), 45–56 (2006)



Weighted Factors for Evaluating Anonymity

Khalid Shahbar^(✉) and A. Nur Zincir-Heywood

Dalhousie University, Halifax, Canada
{Shahbar,Zincir}@cs.dal.ca

Abstract. Many systems provide anonymity for their users, and most of these systems work on the separation between the users' identity and the final destination. The level of anonymity these services provide is affected by several factors, some of which are related to the design of the anonymity service itself. Others are related to how the system is used or the user's application/purpose in using the anonymity service. In this paper we: (i) propose five factors that aim to measure anonymity level from the user's perspective; (ii) evaluate these factors for three anonymity services, namely Tor, JonDonym, and I2P as case studies; and (iii) present a mechanism to evaluate anonymity services based on the proposed factors and measure their levels of anonymity.

Keywords: Anonymity factors · Metrics · Tor · Jondonym · I2P

1 Introduction

There are many tools, applications, and websites on the Internet claiming to protect the privacy of their users. The levels of privacy protection provided by these services are different based on the way they work. For example, VPN (Virtual Private Network), which can be provided either as a free or a paid service, hides the user's identity while surfing the Internet anonymously. However, the VPN service provider has access to the user's identity and his/her activity on the Internet. Some of these service providers also keep the logs of their users. This is also the case with free proxy websites, which claim that they protect the user's identity.

Tor, JonDonym, and I2P are popular anonymity services. They provide anonymity to their users to hide their identity from Internet web servers and hide the websites they have accessed. These systems prevent not only the web servers from revealing users' identities, but also the operators of the systems themselves from identifying the users. However, there are many details behind this kind of anonymity that might not be clear or obvious to the user.

Therefore, the anonymity level of the users is not the same, even when using an anonymizing tool. The reason behind using an anonymity service varies from one user to another. This could affect the anonymity level and the choice of the

right anonymity service. The design of the anonymity tools varies based on: (i) Which services such a tool offers to users, and (ii) How the user decides or measures anonymity level, given all the different anonymity services. In this paper, we present a method of calculating and comparing user anonymity levels that takes into consideration the different needs of different users to answer the aforementioned questions. Therefore, we aim to assist the user in choosing the most suitable anonymity service for their needs. The proposed method depends on evaluating anonymity systems based on five factors. To measure the anonymity level using this method, the factors are converted to numeric values in order to assign weights and scores. In addition, each factor is compared with the others according to the goal or purpose of anonymity. Therefore, the relative weights (importance) of the factors are determined based on who is using the anonymity service and why. In doing so, our objective is to provide a comprehensive measurement technique that could be used to evaluate the level of anonymity based on the environment in which the anonymity service is used.

The rest of this paper is organized as follows. The related literature is reviewed in Sect. 2. The Tor network, the JonDonym network, and the I2P network are discussed in Sect. 3. Section 4 presents and discusses the five factors regarding the level of privacy in anonymity services studied in this work, and Sect. 5 evaluates these anonymity factors. Finally, conclusions are drawn and potential future work is discussed in Sect. 6.

2 Related Literature

Measuring the anonymity level is a challenge for a number of reasons. One is the difference in the design and the goal of the anonymity systems (networks). On the other hand, there is no single way to measure anonymity levels on different anonymity networks. In addition, anonymity level is not directly quantifiable as compared to other network traffic measurements such as delay, bandwidth, volume, etc. In [13], Ries et al. evaluated five anonymization tools with regard to performance, usability, anonymity, network reliability, and cost. The evaluated tools were Tor, I2P, JonDonym, Perfect Privacy and Free proxies. Performance factors used to evaluate and rank these tools were Round Trip Time (RTT), Inter-Packet Delay Variation (IPDV), and throughput. Additionally, they used installation, configuration, and verification of the anonymization connection as factors to define the usability of these tools.

Dhiah el Diehn et al. examined the usability of four anonymity tools (Tor, JonDo, I2P, and Quicksilver) during the installation phase [1]. They detailed the installation process of these tools, applying four tasks to test the installation phase: success of installation, success of configuration, confirmation of anonymization, and ability to disable anonymization. To test the usability of these tools, they used eight guidelines from [3], which focused on the user's ability to perform the four tasks mentioned above.

Wendolsky et al. compared Tor and AN.ON (JonDonym) from the user's perspective, based on performance and number of users [18]. Latency and bandwidth

were used to measure performance, and the results showed that Tor performs unpredictably based on time of day.

The above studies focused mainly on evaluating anonymity services based on their performance or usability, where anonymity was not the focus of the evaluation. On the other hand, there are studies where measuring anonymity was the main goal. In these cases, the idea of measuring anonymity is about minimizing the ability of an attacker to correlate the sender and the receiver, even if they communicate over a channel observed by the attacker. To anonymize against such a threat model, Chaum [2] presented the concept of the “anonymity set”, in which the set is the total number of participants in the anonymity service that may include the sender. When the size of the set is increased, the anonymity level is considered to increase as well.

Serjantov and Danezis [14] developed the concept of the anonymity set by using the information-theoretic metric based on anonymity probability distribution. Diaz et al. [4] also used an information-theoretic model to evaluate the anonymity level of a system in a particular attack scenario.

Murdoch [12] surveyed studies performed on measuring anonymity for low-latency anonymous networks and high-latency email anonymous networks and discussed the development of the techniques used for measuring anonymity.

Even though the above studies have been important in measuring anonymity levels, the “anonymity” of the anonymity services is affected by other factors, too, such as the users’ behaviors and browsers settings. Therefore, in this research, we present a method to measure the level of anonymity by analyzing the anonymity service from different perspectives and propose metrics (factors) that enable us to measure the anonymity of such services. The anonymity set which is presented by Chaum [2] is a way to measure the level of anonymity on the multilayer-encryption anonymity networks. It presents the number of possible choices to which a message on the anonymity network belongs for a specific user. The higher the value of the anonymity set, the better the anonymity becomes. This way of measuring the anonymity level focusses on the probability of linking the message to the user. However, the level of anonymity could be affected by other factors, too. Therefore, in this research, weighted factors for measuring anonymity services are presented as another way to measure and quantify the anonymity level. The method takes into consideration multiple factors: quantifying, comparing and applying them to evaluate the anonymity level of different use cases.

3 Anonymity Systems Studied

Multilayer-encryption anonymity networks share the goal of providing anonymous services to their users. The anonymity services vary in terms of design, performance, delay, and provided services. The following introduces the most popular multilayer-encryption anonymity networks: Tor, JonDoNym, and I2P.

3.1 Tor Network

The Tor network is based on volunteers to run their machines as Tor relays (also called routers or nodes). Tor provides anonymity to its users by hiding the IP addresses of the users and by hiding the content of the users' traffic, as long as that traffic is still on the Tor network. The IP addresses of the users are hidden by relaying all the users' requests through the Tor network. The users' traffic is hidden by dividing the packets into smaller fixed-size encrypted cells. Tor also provides a service called Hidden Services that hides the IP address of a web server for users who want to keep their identities hidden.

There are three types of nodes on the Tor network: entry node, middle node and exit node. The entry node is the first node that the user communicates with when trying to establish a circuit to carry their traffic. The middle node is an intermediate node that lies between the entry node and the exit node, and the exit node is the node used to relay the user's request to the web server. Since all three types of nodes are run by volunteers, running an exit node is an optional choice available while configuring the node to run in the Tor network. The exit node has the option to be configured for allowing certain types of traffic based on the port number. This enables the volunteered user who runs the exit node to determine the type of traffic to block/pass through the exit node.

Whenever the user sends his/her traffic through Tor, a virtual circuit is used to relay the user's traffic. The virtual circuit consists of a connection of the three types of nodes (entry, middle, and exit nodes). The user starts by establishing a TLS (Transport Layer Security) connection with the first node. After the connection is made with the first node, the user requests that the entry node extend the connection to the middle node. Finally, the connection is extended again to the exit node. The Tor browser is responsible for translating all the user's requests to the virtual circuit. This includes hiding the IP address of the user, dividing the packets into smaller cell(s), encrypting the traffic with three layers of encryption, receiving the data from the web server that comes in encrypted cells and decrypting the received cells.

3.2 JonDoNym

JonDonym is a network of mix cascades, providing anonymity to the users based on multilayer encryption. The cascade consists of two (free) or three (paid) mix servers. The user starts the connection to the JonDonym network by selecting the mix cascade. Currently there are five free cascades and eleven paid cascades the user can choose from.

Only one active connection to one cascade is possible during the user's connection to the JonDonym network. Each HTTP request will create a connection from the browser (JonDoFox) with the client software JonDo. The JonDoFox browser can generate multiple connections with the JonDo. All these connections are multiplexed into one connection to the first mix server, which receives connections from multiple users. All the users' connections are then multiplexed

into one TCP/IP connection to the second mix, or to the last in case of only two mixes in the cascade.

The information about the available cascades, the number of users, the loads, and the mix status are stored in the InfoService [9]. The user gets the information about the cascades from the InfoService, and the last mix sends the users' requests to cache proxies. Multilayered encryption is used during the communication between the user and the last mix, which ensures that even the mixes cannot access the user's data. The path that the user's data takes is fixed based on the chosen cascade. To choose another path (cascade), the user has to start a new connection to the JonDonym. The user can only have one connection to one cascade at any given time.

3.3 I2P

I2P network is a decentralized anonymous network with no central database or server that contains the network database. The network database (netDb) is distributed by using the Kademlia algorithm [10], which is used in many applications where peer-to-peer (P2P) communication is needed in a decentralized network. The information that the user gets from the netDb enables the user to build tunnels. Communications over I2P require inbound and outbound tunnels, which are unidirectional. The netDb contains the leaseSet of the tunnels and routers. LeaseSet shows the routers involved in a tunnel. RouterInfo in the netDb shows how to contact a specific router. The user has the option to modify the number of routers in the outbound tunnel. I2P uses the concept of garlic routing [5], where layered encryption is implemented in addition to binding multiple messages together. The messages within the I2P network are encrypted end-to-end as long as the two communication parties are within the I2P network. However, when the user communicates with an end-system that is outside of the I2P network using an outproxy, then the encryption is not end-to-end.

By default, the user within the network transfers their data and that of other users where the user's machine functions as a resource for the network. The user can change the amount of bandwidth dedicated to the network from the console. The users' contributions in relaying the network data are restricted by relaying the data only within the I2P network. A different configuration is required when a user wants to relay the I2P traffic to an end-system outside of the I2P network (outproxy). The number of outproxies in the I2P network is limited.

One of the major differences between I2P and other anonymity networks such as Tor and JonDonym is that I2P is designed as a private network. The users mainly communicate within the network. The user builds two tunnels: inbound and outbound. The inbound tunnels are used to receive messages, and the outbound tunnels are used to send messages.

4 Proposed Factors

This section presents the five proposed factors to analyze the anonymity level of the aforementioned anonymity systems (Tor, JonDonym, and I2P).

4.1 The Level of Information Available for the Service Provider

When a Tor user connects to the Tor network, a virtual circuit is created. The circuit consists of three nodes; the first node has the actual IP address of the user, and therefore his identity, but it does not have knowledge of his Internet activity. This information could be used to perform attacks that depend on the correlation between the duration, data, and the server. The exit nodes, through which all the requests of the users are relayed, have a considerable amount of information, as they are the links between the Internet and the Tor users. The operator of the exit node has the ability to know and statistically evaluate the user's activities on the Tor network [11]. Another important fact about the exit node that might not be clear for non-technical users is that the encryption of the requests through the exit node are all based on the encryption of the original requests and has nothing to do with the three levels of encryption on the Tor Network. Therefore, the exit node alone can breach the anonymity of the users if they use their login information to access their email or any web server without sending an encrypted request.

Furthermore, JonDonym works in a similar fashion to Tor. The first point on the JonDonym network (First Mix) receives the connection request from the user, which has the information about the connection duration and the user's identity. The last point (Last Mix) does not know the user's identity, but it has the activities or the websites that the users request. The encryption layers used by JonDonym and Tor overlay networks protect the data, even from the operators; an exception is when the data sent by the user to the webserver is not encrypted; then, the last node/mix has the ability to access the data sent by the user. The anonymity mechanism in Tor/JonDonym depends on relaying the user data through multiple points (Node/Mix). Each node/mix only knows part of the connection information, not the whole information required to connect the user to the request.

On the other hand, what if all the nodes/mixes on the path between the user and the server are compromised or attacked? On the Tor network, the three nodes in the circuit path are selected by using the path selection protocol, which specifies the three nodes the user will use to relay the data in conjunction with the policy that the exit node operator defines. In addition, the user has the ability to override the path selection protocol and to choose a specific exit/entry node. This flexibility and randomness in node selection makes it harder for an attacker to target a specific user by trying to compromise the three nodes that the user selects. Moreover, running and compromising three nodes do not mean that these three nodes will be selected by the path selection protocol.

On the JonDonym network, this type of attack is also possible; the difference lies in the operation of the mixes. The number of mixes on the JonDonym is fewer than the number of nodes on the Tor network. But, the operators of the mixes are registered with their identities. They also sign an agreement with JonDonym not to exchange information between operators of the mixes and not to save user data. One of the differences between Tor and JonDonym is that JonDonym mixes do not change, and the path is always the same. In the case

of cooperation between all the mixes, it is possible to breach user anonymity on the JonDonym network.

Last but not the least, the goal of the I2P network is different than Tor and JonDonym. I2P is designed to provide anonymity for the users within the I2P network; However, that does not mean that I2P services are limited by the network boundaries. Browsing webpages outside the I2P network requires configuring the user's machine to use an outproxy. In this case, the information available to the outproxy is similar to Tor's exit router or JonDonym's last mix. The outproxy has access to all traffic passing through; if the traffic is not encrypted, the outproxy can see sensitive information.

The common point between the three anonymity services is that at any point, part of the network has some user information. This information could be the IP address of the user that is available at the first point on the anonymity network that connects the user to the network. Or, it could be the amount of traffic that the last point can see when sending traffic to its final destination. Thus, the difference in the design of the anonymity services regarding how to relay the traffic does affect the difficulty (anonymity) of correlating the user with their traffic.

4.2 Blocking Anonymity and Obfuscation Options

The anonymity systems can hide user activity on the Internet, but could not always hide the fact that such a system is in use. Sometimes, using anonymity systems might raise questions about why such a system is in use. In some countries, the IP addresses of the hosts running such systems are blocked, so obfuscation systems are used to evade these. Furthermore, Pluggable Transports (PT) [17] (obfuscation for Tor) work as an interface between the Tor user and the Tor network. The user connects to a pluggable transport, which sends the connection request to the Tor network on behalf of the user in order to hide the connection between the user and the Tor network. These tools work differently, using different techniques to resist different blocking methods. For example, Obfs3 [16] is one of the PT that obfuscate the Tor TLS to look like random strings, using another layer of encryption to wrap the TLS handshake used by Tor. Scramblesuit [19] is another PT designed to prevent active probing attacks.

JonDonym has two options for bypassing blocking of the service. The first one is using TCP/IP forward method, in which the user will use an encrypted connection to another user who has unblocked access to the JonDonym network. The second method is using Skype to tunnel the blockage of the JonDonym service, which is more reliable than using the TCP/IP forward method.

On the I2P network, there are no obfuscation options similar to Tor pluggable transports, and it is thus possible for an observer to collect the routers' IP addresses with a harvesting attack [6]. However, the I2P network implemented other improvements in the design of the transport layer to obscure the identification of the I2P network traffic. In addition, several obfuscation options are still considered by the developers of the I2P network, including using padding

techniques at the transport layer to achieve random length, studying the signature of the packet size distribution, and studying the technique used to block Tor.

In short, an observer, who wants to de-anonymize the user, needs to determine that the user is connecting to the anonymity service in the first place. Therefore, the existence of obfuscation options is a factor that should be taken into consideration to measure the level of anonymity.

4.3 Application and Anonymity

The common way to use an anonymity service is to use the default browser of the aforementioned services to browse the web. However, these anonymity services can be used with other applications in addition to web browsing. This requires the user to configure the application and the anonymity service to work together. The configuration for these applications is not that simple for non-technical users. When configuring any application to work with an anonymity service, it is important to fully understand how this application works to ensure that the user information is not leaked.

The configuration of the application and how the user sets the application on the anonymity network is an important issue. For example, the web browser contains many details other than what anonymity system the user is using. The anonymity tools aim to make their browsers undistinguishable to raise the anonymity level. Tor browser is a modified version of Firefox based on Mozilla's Extended Support Release (ESR) Firefox branch [15]. It includes HTTPS-Everywhere [7], NoScript, modifying some of the default Firefox settings, and the default extension settings. JonDoFox is the browser of JonDonym and is a modified version of Firefox [8].

Even when using the default browser for anonymity services, the correct configuration for the browser is important to ensure the safety of the user against many Internet websites that track their visitors. Moreover, some of the tools used by web sites could also identify the user or their behavior for the purpose of advertisements, collecting data for different types of studies, or building a database about the visitors of the website. Thus, the question to consider is: how such tools address the trade-off between browsing the websites with full offered services and preserving the anonymity of the users.

4.4 Authority and Logs

No doubt that the policy of the anonymity services about the cooperation with the authority (operator or regulator) and keeping logs affects the level of privacy. For example, JonDonym's agreement with the operators requires keeping no logs and not exchanging information between operators of the mixes. The reason behind this policy is that the identities of the operators are known, and they work according to the regulations in their own countries. Therefore, in JonDonym, there are several points that must be taken into consideration when evaluating the anonymity of such a system:

- The mixes that construct the path are fixed. That means knowing that the user employs one of these mixes, e.g., the last mix, implies knowing the first and the second mix.
- The number of mixes on the JonDonym network is very limited compared to the Tor network. On the JonDonym network, there are only nine cascades; six are operated by companies and three by individuals.
- The operators of these mixes are known and registered. They work according to the regulations of the authorities in their countries.

On the other hand, on the Tor network:

- The nodes that construct the user's path are not fixed. The user connects to three nodes that change periodically. Therefore, knowing that the user connects to a specific exit node does not necessarily imply knowing the first or the middle node.
- The number of Tor nodes is around 8000, which makes it relatively harder to get information about them.
- The operators of these nodes are not known. Tor does not require their users to identify themselves when offering to run a node. This might help to protect the operators' identities, but it does not guarantee that the operators are trusted.
- The nodes on the Tor network are supposed to be online as much as possible. However, there is no guarantee because most of these nodes are run by volunteers.

Furthermore, on the I2P network:

- The I2P user has the option to modify the number of routers used when exchanging messages. In addition, end-to-end encryption is used. The concept of garlic routing is also used when exchanging messages. This way, messages that pass through the routers are not distinctive, which means the purpose or the content of the messages cannot be extracted or inferred easily. For example, information such as whether the messages form an extension to the number of routers in the tunnel or if they contain data would not be extracted from the messages.
- The I2P network is decentralized, so there is no single point that is responsible for or represents the network.
- The user does not need to know all the routers in the network to be able to use that network's resources.
- I2P network's design is different from Tor and JonDonym; it is basically designed to provide a private network within the Internet. The number of outproxies is very limited. This also makes the browsing outside the network low compared to Tor and JonDonym. Therefore, the possibility that the user will frequently use the same exit point is high. This does not mean that it is a threat, but increases the probability of correlating the user with their traffic based on factors such as access time, duration, and the amount of data used.

Based on the above, what information the service provider (operator) has about the user and the operator's willingness to provide this information when asked to do so is also important in measuring the level of anonymity.

4.5 Threat Models

The anonymity services are built based on the separation between the user identity and the data sent or received by the user. One of the threats that such services face is somehow correlating the user data with the final destination data. Hypothetically, this may be possible by monitoring the first point in the anonymity network and the last point that connects the user with the final destination through data analysis. For example, the path on the JonDonym network is known, and if the attacker has the ability to monitor the traffic from the first mix and the last mix (out of the last mix), then connecting the users of this cascade and the amount of sent and received data may be possible. The path on the Tor network is not fixed, but the correlation is also a possible threat. To this end, there are studies on using marking techniques to trace user activities, but they are often limited to a specific user, a specific webserver, or even a specific exit node. The attacker could compromise both an entry node and an exit node, in which case the traffic out of the entry node is marked. The attacker then watches for the mark to appear at the exit node. On the other hand, the design of I2P network makes this kind of correlation a low threat. The path is not fixed or specified; users build inbound and outbound tunnels that do not count on the type of the router. All routers on the networks can be part of any path. The encryption mechanism provides for the confidentiality and the integrity of the messages. However, if the attacker has the resources to monitor all routers, then they may have enough data to discover paths.

As for the JonDonym network, this type of attack can target a mix server. A mix server has a limit on the number of users it can serve. The attacker could use this limit to break the anonymity of the mix server. If the attacker connects to a mix server to fill its capacity (n) to the point $(n - 1)$ when the user connects to the only space left in the mix server, the attacker could then isolate and detect the user's traffic.

The threat models are not the same for all anonymity services; what is considered a threat to one service may not be applied to another anonymity service. Even when they share the same threat to a certain saturation point, the level of the risk is not always the same. Therefore, to measure the anonymity of any anonymity service, the threat model should be taken into consideration, based on the environment or the purpose for which the anonymity service is used.

In summary, evaluating the level of anonymity should be done in a comprehensive way that takes into consideration the purpose, the design, and the environment. Thus, these five factors: the level of information available to the service provider, the obfuscation options, application anonymity, the authority and the logs, and the threat models are used in this research to measure the effectiveness of any anonymity service.

5 Evaluation

This section discusses how the aforementioned factors can be used to measure the anonymity of Tor, JonDonym, and I2P.

5.1 Factor Calculation

As a first step, we quantify these factors, so they are grouped into three categories, Table 1. These categories (High, Mid, and Low) are converted into numerical values as 100, 67 and 33, respectively. These numbers are chosen to simplify the representation of the three categories into three intervals. The High range is between 68–100 and represented by 100. The Mid range is between 34–67 and represented by 67, and so on. The exception is for the obfuscation, where it is labeled as “Yes” or “No”, depending on it is used or not. The reason is that some of the anonymity systems involve obfuscation techniques, and others do not. Therefore, the value is set to 100 (No) and 0 (Yes). The higher the values for these factors, the lower the anonymity level of the system. For example, a 100 in the Threat model factor is applied whenever the threat in the case under study is very strong (i.e., highly probable). The three categories are represented by 100, 67, and 33 as an approximation for High, Mid, and Low ranges. It is possible to expand this step to improve the accuracy of quantification of the factors by: (1) Instead of using three levels; the factors could be evaluated as a scale, for example, from 10 to 100, (2) Furthermore, each value on the scale could represent the level of the anonymity of the factor in a predefined way. This way, the value of the factors is determined more accurately. For example, if we apply the extended scale to the “Threat Models” factor, then the values will include more intervals from 10 to 100 instead of 33, 67, and 100. The threats or attacks on the anonymity systems should be ordered to match the scale from 10 to 100. This requires the study and evaluation of all possible threats on the anonymity systems and their applicability. This way, the scale has predefined values for every possible threat against the anonymity systems in the threat models factor. The same step could be used for the other factors.

Table 1. Proposed anonymity factors

Level of information	High	Mid	Low
	100	67	33
Obfuscation	Yes	No	
	0	100	
Authority and log	High	Mid	Low
	100	67	33
Application configuration	Low security configuration	Mid security configuration	High security configuration
	100	67	33
Threat model	Low cost	Mid cost	High cost
	100	67	33

5.2 Weight Calculation

Given that the weights of the factors may vary from one evaluation environment to another, quantifying these factors to measure anonymity is necessary but

insufficient by itself. Also, the weights of the factors have to be considered. Therefore, the “Pairwise Comparison” technique is employed to evaluate the weight of these factors. Each one of the factors is compared with all other factors; then, the weight for the factor is calculated based on these comparisons. The higher the weight of a factor, the more important it becomes for the anonymity of a given service. Calculating the weights is performed until all factors are evaluated comparatively, as shown in Table 2.

Table 2. Calculating the weights

γ_1					
γ_2	$2 \gamma_1$				
γ_3	$\gamma_1 \gamma_3$	γ_3			
γ_4	γ_4	γ_4	$\gamma_3 \gamma_4$		
γ_5	γ_1	$\gamma_2 \gamma_5$	$\gamma_3 \gamma_5$	$\gamma_4 \gamma_5$	

The first column in the table represents the five factors. The first factor “level of information available to the service provider” is presented by γ_1 . The second factor is presented by γ_2 and so on. The second column shows the importance of γ_1 compared to all the other factors. The third column shows the importance of γ_2 compared to all other factors except γ_1 , this is because the comparison between γ_2 and γ_1 already done in the second column. The comparison continues till all the factors compared with each others. Each cell starting from the second column shows the result of the comparison between two factors; the most important factor is shown in the cell, however if both have similar importance then both appear in the cell.

Table 3 shows the weights of the five factors after the comparison and their total value. Based on the weights value, the level of information, the application configuration, and the authority and log factors have the same weights (importance). The obfuscation has the lowest importance, compared to the other factors. The weights represent the importance of each factor compared to the other factors. Using the pairwise comparison helps in deciding how to rank or weight the factors compared to others.

Table 3. Final weights of the factors after pairwise comparison

γ_1	γ_2	γ_3	γ_4	γ_5	Total
4	1	4	4	3	16

5.3 Weighted Anonymity Factor

Equations 1 and 3 are applied after calculating the values of the factors and weights. Equation 2 is the total of the weights of the factors. WF in Eq. 1 is the Weighted Anonymity Factor, (f) represents the value of a factor and γ represents

the weight. n in Eq. 2 represents the number of factors. T_γ in Eq. 3 is the total weight.

$$WF = \gamma_1 f_1 + \gamma_2 f_2 + \gamma_3 f_3 + \gamma_4 f_4 + \gamma_5 f_5 \quad (1)$$

$$= \sum_{i=1}^n \gamma_i f_i \quad (2)$$

$$(T_\gamma) = \sum_{i=1}^n \gamma_i \quad (3)$$

The measurements may vary from one environment to another, where different factors are applied or when the numerical conversion is different than what is used in Table 1. To generalize measurements, Eq. 4 shows converting the calculated values of the weighted factors (WF) from Eq. 1 to a percentage by using the minimum and maximum value.

$$WF(\%) = \left(1 - \frac{WF - \text{Min}(WF)}{\text{Max}(WF) - \text{Min}(WF)}\right) * 100 \quad (4)$$

Equation 4 can be rewritten after calculating the weights to the form in Eq. 5.

$$WF(\%) = \left(1 - \frac{WF - 495}{1600 - 495}\right) * 100 \quad (5)$$

5.4 Evaluation Case Study

In this case study, three users participate to compare the levels of anonymity. It is important to note that the evaluation does not aim to identify the best anonymity service; it aims to evaluate the level of anonymity according to the environment in which these users use the anonymity services.

The first user (A) uses standalone Tor to browse Internet websites. She configures Chrome browser to work with Tor by setting the browser to access Tor via Socket. To increase the anonymity level, she adds Scramblesuit as an obfuscation option to her “torrc” file to access Tor via a bridge. She browses websites on the Internet which include a compromised web server by an attacker. The web server injects a code to force the browser to request images from another website that belongs to the attacker. The attacker aims to identify the user by forcing the browser to send requests without using the Tor network.

User (B) chooses to use JonDonym as an anonymity service. He does not have a technical background. All the settings are left as default. The only addition to the default setting is that he chooses to use the TCP/IP forwarder. The user (B) wants all the activities that he performs on the Internet to be anonymous. Therefore, he uses JonDoFox to browse all the Internet websites. He usually visits web sites such as news, videos, email, Internet shopping, and his bank account.

User (C) lives in a country where the Internet is censored and some websites are blocked. Therefore, he uses Tor to gain access to the blocked Internet blogs. The user (C) browses these blogs and participates on them via Tor. He is concerned about hiding his identity, so he uses the Internet via the company

network where he works. It seems that he is the only person who is using Tor in this company. The user organizes his time so that he only accesses Tor at the end of the day between 5–6 pm on weekdays.

Table 4. Evaluated factors for users (A), (B) and (C)

	Level of information	Obfuscation	Authority and log	Application configuration	Threat model
A	33	0	33	100	67
B	100	0	67	33	100
C	67	100	100	33	67

Table 4 shows how these scenarios are converted to measurable numeric values, using the proposed factors. Table 4 is calculated based on the given information about the scenarios above and how the users (A), (B) and (C) are using these anonymity services. For example, the user (C) did not include an obfuscation option when using the anonymity service; therefore, the obfuscation value is measured as 100. The user (A) prefers to use his favorite browsers instead of using the default Tor browser. Therefore, the possibility of a DNS leak is higher, especially when accessing suspicious websites or when using any application other than browsing. Based on that, user (A) gets 100 on the application configuration. Even though the user (B) uses some sort of obfuscation, he misses the fact that browsing any website already linked to his real identity such as his email or bank account, even while using an anonymity service, does not mean that he is anonymous. Furthermore, the information available to the exit node in this case is high, even if the information does not contain passwords. So, the level of information is evaluated as 100 in this case. The same applies to the user (C); he uses Tor at the same time daily from the same place where no one else is using Tor. Using Table 4 and Eq. 1, the weighted factors are calculated as follows:

$$WF = \gamma_1 f_1 + \gamma_2 f_2 + \gamma_3 f_3 + \gamma_4 f_4 + \gamma_5 f_5$$

$$WF = 4f_1 + f_2 + 4f_3 + 4f_4 + 3f_5$$

$$WF_A = 4 * 33 + 0 + 4 * 33 + 4 * 100 + 3 * 67$$

$$= 865$$

$$WF_A(\%) = \left(1 - \frac{865 - 495}{1600 - 495}\right)$$

$$= 66.5\%$$

$$WF_B = 4 * 100 + 0 + 4 * 67 + 4 * 33 + 3 * 100$$

$$= 1100$$

$$WF_B(\%) = \left(1 - \frac{1100 - 495}{1600 - 495}\right)$$

$$= 45.2\%$$

$$\begin{aligned}
 WF_C &= 4 * 67 + 100 + 4 * 100 + 4 * 33 + 3 * 67 \\
 &= 1101 \\
 WF_C(\%) &= \left(1 - \frac{1101 - 495}{1600 - 495}\right) \\
 &= 45.16\%
 \end{aligned}$$

Based on the above calculations, user (A) has a higher level of anonymity than either of users (B) or (C). The level of anonymity of user (A), (B) and (C) may change based on the anonymity services they use or even their behavior. The weighted factor method is designed to take into consideration the users' environment when evaluating the anonymity level. The factors themselves are parameters that could be adjusted based on the scenario for which an anonymity system is used.

6 Conclusion

In this paper, we propose and evaluate five factors that affect the level of privacy in anonymity services. Understanding these factors and knowing how to address them is an important step in improving users' privacy. To this end, three popular anonymity systems, namely Tor, JonDonym, and I2P, were used as case studies to analyze these factors. Our analysis showed that even though these systems aim to provide anonymity to their users, user information is visible to the operators of the services. Furthermore, the infrastructure and the browser settings vary from one system to another. The setting is configured based on the developers'/administrators' evaluation of possible threats. The same threat might be considered high in one system but low in another. We applied a measurable mechanism to evaluate the anonymity of a given situation based on the factors we proposed. This evaluation could be used on any anonymity system using different scenarios. Future research will continue to analyze other anonymity systems based on the proposed five factors and will evaluate them using the proposed approach under other adversarial conditions.

Acknowledgment. This research is partially supported by the Natural Science and Engineering Research Council of Canada (NSERC) grant, and is conducted as part of the Dalhousie NIMS Lab at <http://projects.cs.dal.ca/projectx/>. The first author would like to thank the Ministry of Higher Education in Saudi Arabia for his scholarship.

References

1. Dhiah el Diehn, A.-T., Pimenidis, L., Schomburg, J., Westermann, B.: Usability inspection of anonymity networks. In: 2009 World Congress on Privacy, Security, Trust and the Management of e-Business, CONGRESS 2009, pp. 100–109. IEEE (2009)
2. Chaum, D.: The dining cryptographers problem: unconditional sender and recipient untraceability. *J. cryptol.* **1**(1), 65–75 (1988)

3. Clark, J., Van Oorschot, P.C., Adams, C.: Usability of anonymous web browsing: an examination of Tor interfaces and deployability. In: Proceedings of the 3rd Symposium on Usable Privacy and Security, pp. 41–51. ACM (2007)
4. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36467-6_5
5. Garlic routing (2014). <https://geti2p.net/en/docs/how/garlic-routing>
6. I2P's threat model: Harvesting attacks (2010). <https://geti2p.net/en/docs/how/threat-model>
7. HTTPS-everywhere extension. <https://www.eff.org/https-everywhere>
8. Anonymous surfing with JonDoFox (n.d.). <https://anonymous-proxy-servers.net/en/jondofox.html>
9. Jonymym InfoService (n.d.). <https://anonymous-proxy-servers.net/en/help/info-service.html>
10. Maymounkov, P., Mazières, D.: Kademia: a peer-to-peer information system based on the XOR metric. In: Druschel, P., Kaashoek, F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, pp. 53–65. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45748-8_5
11. McCoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining light in dark places: understanding the Tor network. In: Borisov, N., Goldberg, I. (eds.) PETS 2008. LNCS, vol. 5134, pp. 63–76. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70630-4_5
12. Murdoch, S.J.: Quantifying and measuring anonymity. In: Garcia-Alfaro, J., Lioudakis, G., Cuppens-Boualahia, N., Foley, S., Fitzgerald, W.M. (eds.) DPM/SETOP -2013. LNCS, vol. 8247, pp. 3–13. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54568-9_1
13. Ries, T., Panchenko, A., Engel, T., et al.: Comparison of low-latency anonymous communication systems: practical usage and performance. In: Proceedings of the Ninth Australasian Information Security Conference, vol. 116, pp. 77–86. Australian Computer Society, Inc. (2011)
14. Serjantov, A., Danezis, G.: Towards an information theoretic metric for anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-36467-6_4
15. The design and implementation of the Tor browser (2017). <https://www.torproject.org/projects/torbrowser/design/#idm29>
16. Tor Obfs3 (n.d.). <https://gitweb.torproject.org/pluggable-transport/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt>
17. Tor pluggable transports (n.d.). <https://www.torproject.org/docs/pluggable-transport.html.en>
18. Wendolsky, R., Herrmann, D., Federrath, H.: Performance comparison of low-latency anonymisation services from a user perspective. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 233–253. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75551-7_15
19. Winter, P., Pulls, T., Fuss, J.: ScrambleSuit: a polymorphic network protocol to circumvent censorship. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, pp. 213–224. ACM (2013)

Author Index

- Allard, Morgan 144
Asghar, Muhammad Rizwan 189
- Bindel, Nina 225
Blömer, Johannes 3
Blot, Elliott 53
Boussaha, Mohamed Recem 160
Buchmann, Johannes 225
- Cui, Shujie 189
- De Santis, Giulia 144
Deruyver, Aline 144
Doynikova, Elena 293
Dreier, Jannik 53
Duque Antón, Simon 259
- Ferrante, Alberto 242
Festor, Olivier 144
François, Jérôme 144
Fraunholz, Daniel 259
- Galbraith, Steven D. 189
Garcia-Alfaro, Joaquin 293
Gay, Richard 18
Gonzalez-Granadillo, Gustavo 293
Goux, Quentin 144
Graupner, Hendrik 173
Gudes, Ehud 35
Günther, Peter 3
- Hallé, Sylvain 160
Halpin, Harry 83
Hamdi, Fayçal 144
Hu, Jinwei 18
- Jha, Sumit Kumar 277
- Khaled, Abdelaziz 127
Khoury, Raphaël 160
Koshiba, Takeshi 95
Kotenko, Igor 293
Krämer, Juliane 225
Krentz, Konrad-Felix 173
Krummel, Volker 3
- Lafourcade, Pascal 53
Lagraa, Sofiane 144
Lahmadi, Abdelkader 144
Lammari, Nadira 144
Legrand, Véronique 144
Lipps, Christoph 259
Logrippio, Luigi 111
Löken, Nils 3
- Malek, Miroslaw 242
Mantel, Heiko 18, 225
Martinelli, Fabio 242
Mazaheri, Sogol 18
Meinel, Christoph 173
Mercaldo, Francesco 242
Milosevic, Jelena 242
Mueller, Johannes Karl Martin 259
- Navarro, Julio 144
- Parrend, Pierre 144
Pereira, Olivier 68
Potet, Marie-Laure 127
Pullum, Laura 277
Puys, Maxime 127
- Rabin, Alexey 35
Raj, Sunny 277
Ramanathan, Arvind 277
Rochet, Florentin 68
Russello, Giovanni 189
- Saha, Tushar Kanti 95
Schickel, Johannes 225
Schotten, Hans Dieter 259
Shahbar, Khalid 303
- Tikhomirov, Sergei 206
- Weber, Alexandra 225
Wiedling, Cyrille 68
- Zimmermann, Marc 259
Zincir-Heywood, A. Nur 303