Enhancements for Group Signatures
and
Side-Channel Attacks on Mobile Devices

by

Raphael Spreitzer

A PhD Thesis
Presented to the Faculty of Computer Science in Partial Fulfillment of the
Requirements for the PhD Degree

Assessors

Prof. Stefan Mangard (TU Graz, Austria)
Prof. Bart Preneel (KU Leuven, Belgium)

April 2017

Institute for Applied Information Processing and Communications (IAIK)
Faculty of Computer Science
Graz University of Technology, Austria

Abstract

In this thesis, we address important challenges in terms of privacy as well as security in mobile ecosystems. In the first part we focus on group signature schemes, which allow individuals to anonymously sign messages on behalf of a group. Although group signature schemes have already been proposed for various applications, the practical applicability of such mechanisms still faces delicate challenges and sometimes requires enhanced anonymity management mechanisms. In order to address these challenges, we propose a generic approach that adds controllable linkability to group signature schemes. This feature allows an authorized entity to determine whether two group signatures have been issued by the same anonymous signer. Based on this feature we also propose a novel revocation mechanism that can be generically applied to existing group signature schemes. Our proposed revocation mechanism is transparent for signers and allows immediate revocation of group members. Overall our revocation mechanism extends the existing portfolio of revocation mechanisms and aims to deal with the major bottleneck of revocation in group signature schemes.

Besides privacy-enhancing technologies, we also consider the platform security of mobile devices in this thesis. As mobile devices are extensively used for many different purposes, the information stored and processed must be protected properly. Even though dedicated protection mechanisms are implemented on all software layers, side-channel attacks often allow to bypass such protection mechanisms. We analyze and propose different side-channel attacks on mobile devices. Thereby, we aim to improve the understanding of security and privacy implications of specific components and features on these multi-purpose platforms. More specifically, we propose enhancements for cache-timing attacks that significantly reduce the effective key size of the Advanced Encryption Standard. We also propose a novel sensor-based attack that exploits the ambient-light sensor to infer user input on touchscreen devices. Our practical investigations confirm that not only motion sensors (accelerometer and gyroscope), but also ambient sensors can be used to reliably infer a user's PIN input. In addition, we show that seemingly harmless information provided by the operating system, such as the data-usage statistics that capture the amount of incoming and outgoing network traffic, allow to infer sensitive information. We demonstrate a practical attack that exploits the data-usage statistics to infer a user's browsing behavior with a high accuracy, even when the traffic is routed through the Tor network.

Acknowledgements

I would like to thank all people who supported me in writing this thesis. First of all, I would like to thank my supervisor Stefan Mangard for his valuable input and insightful discussions on side-channel attacks throughout my PhD studies. Special thanks also go to Daniel Slamanig for sharing his knowledge on privacy-enhancing technologies and public key cryptography in general. As supervisors, mentors, and co-authors, Stefan and Daniel deserve special credit for sharing their expertise as well as for their guidance and patience. In addition, I would like to thank my assessor Bart Preneel for giving important feedback.

Furthermore, I would like to thank Jörn-Marc Schmidt for providing me with the required freedom to explore and find a proper research topic in the beginning, and I would also like to thank Thomas Plos for accompanying my first research steps and for teaching me how to write a research paper. In addition, I am also very grateful for the manifold discussions with Karl-Christian Posch.

Besides I would like to thank all people who influenced this thesis, either through discussions or even as co-authors. Especially I want to thank David Derler and Thomas Unterluggauer for fruitful collaborations and writing papers on group signatures. I would also like to thank Benoît Gérard, Simone Griesmayr, Daniel Gruss, Thomas Korak, Clémentine Maurice, and Veelasha Moonsamy for the interesting discussions and their collaborations in various papers on side-channel attacks.

I am also very grateful for lots of interesting discussions with all members of the former SEnSE group, Christoph Dobraunig, Maria Eichlseder, Michael Hutter, Mario Kirschbaum, Florian Mendel, Tomislav Nad, Martin Schläffer, and Erich Wenger, as well as all members of the Secure Systems group, Hannes Gross, Patrick Klampfl, Moritz Lipp, Peter Peßl, Robert Schilling, Michael Schwarz, Samuel Weiser, and Mario Werner. Without these discussions, both private as well as research related, this thesis would not have been possible.

Last but not least, finishing a PhD thesis requires more than just a great environment for being productive. Hence, I would like to express my sincere gratitude to my whole family and friends. Especially my parents Christian and Jutta as well as my two brothers Gabriel and Michael who always supported me in any possible way. Fortunately, I was able to convince many of my friends that privacy is important and, hence, I want to honor all of them anonymously. Cheers!

Raphael Spreitzer
Graz, April 2017

Table of Contents

Abstract iii

Acknowledgements v

List of Tables xi

List of Figures xiii

Acronyms xv

1 Introduction 1
- 1.1 Contributions and Outline 3
  - 1.1.1 Enhancements for Group Signature Schemes 3
  - 1.1.2 Side-Channel Attacks on Mobile Devices 4
- 1.2 Other Contributions 5

I Enhancements for Group Signature Schemes 7

2 Group Signature Schemes 9
- 2.1 Introduction 10
- 2.2 Mathematical Preliminaries 11
- 2.3 Public Key Encryption 12
- 2.4 Digital Signature Scheme 13
- 2.5 Zero-Knowledge Proofs and Σ Protocols 14
- 2.6 Model for Dynamic Group Signatures 15
- 2.7 Sign-and-Encrypt-and-Prove Paradigm 17
- 2.8 Threshold Secret Sharing 17

3 Controllable Linkability 19
- 3.1 Introduction 20
- 3.2 Model for GSs with Controllable Linkability 21
- 3.3 Public Key Encryption and Equality Tests 22
  - 3.3.1 Public Key Encryption in GSSs 22
  - 3.3.2 Trapdoor Equality Test for Public-Key Encryption 23
  - 3.3.3 Modified All-Or-Nothing PKE with Equality Tests 24
- 3.4 Adding Controllable Linkability to PB-GSSs 28

3.4.1 Security Analysis 30
3.5 Comparison with Other Approaches 35
3.5.1 Linkable Group Signatures 35
3.5.2 Traceable Signatures 35
3.5.3 Public Key Anonymous Tag Systems 36
3.5.4 Verifier Local Revocation 37
3.5.5 Comparison 37

4 Linking-Based Revocation 39
4.1 Introduction 40
4.2 State-of-the-Art in Revocation 41
4.3 Building Blocks for GSs with LBR 45
4.3.1 High-Level Idea of GSs with LBR 46
4.3.2 Constant-Time Revocation Checks 46
4.3.3 Distributed Controllable Linkability 47
4.4 GSs with Linking-Based Revocation 49
4.4.1 Discussion and Security 49
4.5 Applying Linking-Based Revocation 52
4.6 Conclusion 54

II Side-Channel Attacks on Mobile Devices 55

5 Taxonomy of Side-Channel Attacks on Mobile Devices 57
5.1 Introduction 58
5.2 Background and Taxonomy 60
5.2.1 A Primer on Smartphone Security 60
5.2.2 Concept of Side-Channel Attacks 60
5.2.3 Types of Side-Channel Information Leaks 61
5.2.4 Software-only Side-Channel Attacks 62
5.3 A New Categorization System 62
5.4 Local Side-Channel Attacks 62
5.4.1 Passive Attacks 63
5.4.2 Active Attacks 65
5.5 Vicinity Side-Channel Attacks 66
5.5.1 Passive Attacks 67
5.5.2 Active Attacks 68
5.6 Remote Side-Channel Attacks 68
5.6.1 Passive Attacks 68
5.6.2 Active Attacks 72
5.7 Trend Analysis 73
5.8 Discussion of Countermeasures 75
5.8.1 Local Side-Channel Attacks 75
5.8.2 Vicinity Side-Channel Attacks 76
5.8.3 Remote Side-Channel Attacks 76
5.9 Conclusion 78

6 Time-Driven Cache Attacks 79
6.1 Introduction . . . 80
6.2 Background . . . 81
  6.2.1 Advanced Encryption Standard . . . 81
  6.2.2 CPU Caches . . . 81
  6.2.3 Seminal Work: Bernstein's Timing Attack . . . 82
6.3 Analysis of the Divide Part . . . 82
  6.3.1 Attacking Different Key-Chunk Sizes . . . 83
  6.3.2 Minimum Timing Information . . . 84
6.4 Analysis of the Conquer Part . . . 85
  6.4.1 Combining Information from the Divide Part . . . 85
  6.4.2 Evaluating the Key-Search Complexity . . . 86
  6.4.3 Choosing Thresholds and Combination Functions . . . 86
6.5 Experimental Results . . . 87
  6.5.1 Setup and Methodology . . . 87
  6.5.2 Attacking Different Key-Chunk Sizes . . . 88
  6.5.3 Minimum Timing Information . . . 90
  6.5.4 Summary of Practical Results . . . 91
6.6 Conclusion . . . 92

7 Exploiting the Ambient-Light Sensor 93
7.1 Introduction . . . 94
7.2 Ambient-Light Sensor . . . 94
7.3 RGB(W) Sensor . . . 97
7.4 Attack Scenario . . . 97
  7.4.1 Training Phase . . . 98
  7.4.2 Exploitation Phase . . . 98
  7.4.3 Security Implications . . . 99
  7.4.4 Observations and Assumptions . . . 99
7.5 Attack Approach . . . 100
  7.5.1 Gathering the Required Data . . . 100
  7.5.2 Detecting PIN Inputs . . . 101
  7.5.3 Determining the Unknown PIN . . . 102
7.6 Evaluation and Results . . . 102
  7.6.1 Comparison of Classification Algorithms . . . 103
  7.6.2 Comparison of Feature Vectors . . . 105
  7.6.3 Guessing PINs Based on Their Probability . . . 105
  7.6.4 Impact of Different Input Methods . . . 106
  7.6.5 Impact of the Sampling Frequency . . . 108
7.7 Limitations . . . 108
7.8 Countermeasures . . . 109
  7.8.1 UI and API Modifications . . . 109
  7.8.2 Rethinking the Permission Model . . . 110
  7.8.3 Application Analysis . . . 110
  7.8.4 User Behavior . . . 111
7.9 Comparison With Related Work . . . 111

7.10 Conclusion and Future Work 112

8 Exploiting the Data-Usage Statistics 113

8.1 Introduction 114
8.2 Background and Related Work 115
  8.2.1 On the Wire 115
  8.2.2 Shared Resources and Software Execution 116
8.3 Android Data-Usage Statistics 116
  8.3.1 Usage Statistics in a Controlled Scenario 117
  8.3.2 Usage Statistics for Real Websites 118
  8.3.3 Usage Statistics in the Tor Setting 118
  8.3.4 Usage Statistics for Mobile Connections 119
  8.3.5 API and /proc Support 120
8.4 Adversary Model and Scenario 121
  8.4.1 Possible Attack Scenarios 122
  8.4.2 Assumptions 122
8.5 Attack Description 123
  8.5.1 Gathering Traffic Signatures 123
  8.5.2 Classification 124
8.6 Evaluation and Results 125
  8.6.1 Intra-Day Classification Rate 126
  8.6.2 Classification Rate for Tor 127
  8.6.3 Inter-Day Classification Rate 129
  8.6.4 Scalability for Larger World Sizes 130
  8.6.5 Comparison with Related Work 130
8.7 Discussion of Countermeasures 133
  8.7.1 Existing Countermeasures 133
  8.7.2 Discussion 134
8.8 Conclusion 135

9 Conclusions 137

Bibliography 139

Index 167

Author Index 169

Author's CV 179

List of Tables

3.1 Applicability of related concepts to achieve controllable linkability. 38

4.1 Comparison of revocation mechanisms. 44

6.1 Device specifications of the test devices. 87
6.2 Sample results on the Samsung Galaxy SII. 91
6.3 Sample results on the Google Nexus S. 92

7.1 Sampling rates on the Samsung Galaxy SIII. 95
7.2 Input methods of three users. 107
7.3 Comparison of related work targeting a set of 50 PINs. 111

8.1 Test devices and configurations. 120
8.2 Google websites that have been merged. 127
8.3 Websites with the worst classifications in the inter-day setting. 130
8.4 Comparison of website fingerprinting attacks in the closed-world setting. 132

List of Figures

3.1 Experiment for LO-linkability. 32
3.2 Experiment for JP-unforgeability. 32
3.3 Experiment for E-linkability. 33

4.1 Naive instantiation of linking-based revocation. 46
4.2 Schematic of our secure instantiation of linking-based revocation. 51

5.1 Scope of attacks for smart cards, cloud infrastructures, and smartphones. 59
5.2 General notion of passive (⟶) and active (←--) side-channel attacks. 61
5.3 Categorization of side-channel information leaks. 61
5.4 Classification of side-channel attacks: (1) active vs passive, (2) logical vs physical, (3) local vs vicinity vs remote. 74

6.1 Possibilities to combine different key chunks: combination of bytes within a row (left) and combination of bytes within a column (right). 83
6.2 Rank evolution for one-byte key chunks. 88
6.3 Rank evolution for two-byte key chunks. 88
6.4 One-byte chunk signatures for the study phase of one run. . . . 89
6.5 One-byte chunk signatures for the attack phase of one run. . . . 89
6.6 Two-byte chunk signatures for the study phase of one run. . . . 89
6.7 Two-byte chunk signatures for the attack phase of one run. . . . 89

7.1 (1) Proximity and ambient-light sensor as well as (2) front camera. 94
7.2 PIN input mask to gather test samples. 95
7.3 Light-sensor information for five consecutive PIN inputs (1-5-9-0). 96
7.4 Schematic illustrating the tilts and turns of the smartphone. . . 96
7.5 RGB(W)-sensor information for five consecutive PIN inputs (1-7-3-0). 97
7.6 Average classification rate over multiple runs with a set of 15 PINs each. 104
7.7 Average classification rate over multiple runs with a set of 50 PINs each. 104

7.8 Average classification rate over multiple runs with a set of 15 PINs each. 105
7.9 Average classification rate over multiple guesses on a set of 50 PINs. 106
7.10 Average classification rate for different input methods on 15 PINs. 107
7.11 Average classification rate for three specific users on a set of 30 PINs. 108
7.12 Average classification rate for different sampling rates on a set of 15 PINs. 109

8.1 TCP packet lengths according to `tcpdump` on the server and data-usage statistics on the smartphone. 118
8.2 Data-usage statistics for the inbound traffic of three samples per website. 119
8.3 Traditional website fingerprinting attack considering a network attacker. 121
8.4 Client-side website fingerprinting attack exploiting a local side channel. 121
8.5 Confusion matrix for the 15 most popular websites in the US. . 126
8.6 Confusion matrix for the 100 most popular websites globally with *google*.** pages merged. 127
8.7 Classification rates considering the k most probable websites returned from the classifier. 128
8.8 Decreasing accuracy for samples captured Δ days after the training data. 129

Glossary

ADB	Android Debug Bridge
AES	Advanced Encryption Standard
AoN-PKEET	All-or-Nothing Public Key Encryption with Equality Tests
ATM	Automated Teller Machine
BBS	Boneh Boyen Shacham
BSZ	Bellare Shi Zhang
BYOD	Bring Your Own Device
CDH	Computational Diffie-Hellman Assumption
CL	Camenisch Lysyanskaya
co-CDH	Computational co-Diffie-Hellman Assumption
CSS	Cascading Style Sheets
DAA	Direct Anonymous Attestation
DDH	Decisional Diffie-Hellman Assumption
DLIN	Decision LINear Problem
DS	Digital Signature
EM	Electromagnetic
EUF-CMA	Existentially Unforgeable under Chosen Message Attacks
FAPI	Fixed Argument Pairing Inversion Problem
GM	Group Manager
GS	Group Signature
GSS	Group Signature Scheme
HTML	Hypertext Markup Language
IND-CCA	Indistinguishable under Chosen Ciphertext Attacks
IND-CPA	Indistinguishable under Chosen Plaintext Attacks

IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
KNN	K-Nearest Neighbor
LA	Linking Authority
LBR	Linking-Based Revocation
LED	Light-Emitting Diode
NIZKPK	Non-Interactive Zero-Knowledge Proof of Knowledge
OCSP	Online Certificate Status Protocol
OS	Operating System
OW-CCA	One-Wayness under Chosen Ciphertext Attacks
OW-CPA	One-Wayness under Chosen Plaintext Attacks
PB-GSS	Pairing-Based Group Signature Scheme
PET	Privacy-Enhancing Technology
PKE	Public Key Encryption
PKEET	Public Key Encryption with Equality Tests
PPT	Probabilistic Polynomial Time
q-SDH	q-Strong Diffie-Hellman Assumption
RA	Revocation Authority
RGBW	Red, Green, Blue, and White
RIG	Runtime-Information Gathering
RL	Revocation List
SEP	Sign-and-Encrypt-and-Prove
SoK	Signature of Knowledge
SSH	Secure Shell
SSL	Secure Sockets Layer
SXDH	Symmetric eXternal Diffie-Hellman Assumption
TLS	Transport Layer Security
UID	User Identifier
URL	Uniform Resource Locator
VLR	Verifier Local Revocation
WEP	Wired Equivalent Privacy

WPA	Wi-Fi Protected Access
XSGS	eXtremely Short Group Signature

1

Introduction

For me, privacy and security are really important.
We think about it in terms of both: You can't have privacy without security.
— Larry Page

Mobile devices, such as smartphones and tablet computers, are already widely employed and represent an integral part of our everyday lives. Due to this tight integration of mobile devices into our lives, both private and business, these devices inevitably store and process sensitive information. Hence, appropriate mechanisms must be implemented in order to protect this data as well as our privacy against a wide variety of stakeholders and, in the worst case, even adversaries. In addition, besides protecting the data on mobile devices, there is also the urgent need to protect our privacy when interacting with service providers of any kind via the Internet but also in less obvious scenarios that involve electronic devices, for example, when using public transport systems.

In order to cope with these challenges and to reduce the amount of revealed information, so-called privacy-enhancing technologies have gained increasing attention among the scientific community. For example, in some scenarios there is no need that a service provider is able to identify specific users. Instead, a service provider should only be convinced that a user is indeed a genuine user of the system. Therefore, concepts such as group signature schemes [CvH91, BMW03, BSZ05, KY05] have been introduced, which allow a user to prove affiliation to a specific group without revealing further information. Such privacy-preserving mechanisms enable interesting applications like, for example, to prove possession of a valid ticket for a public transport system without revealing any identifying information [IVP$^+$13]. The service provider can then verify that passengers are in possession of valid tickets and are indeed allowed to use the service, but in general the service provider is not able to identify specific passengers. Such privacy-preserving mechanisms in transportation systems are also envisioned by the European Union within the EU Directive 2010/40 [Eur10] by encouraging “the use of anonymous data”, anonymization, and data minimization.[1]

The proliferation of such privacy-preserving mechanisms, however, leads to new challenges for specific stakeholders such as service providers. For example, even though users are anonymous among a group of people, service providers might want to ensure that users can be held accountable for inappropriate actions. Therefore, group signature schemes employ an authorized entity that is able to identify a group member in case of misbehavior. Furthermore, in practical applications the need for revocation mechanisms arises in case of misbehavior, meaning the possibility to prevent specific users from proving membership affiliation again. However, revocation in the context of group signature schemes is a non-trivial task as the privacy of users should be protected and the revocation of one specific user should neither affect the privacy nor the signing capabilities of other users. In fact, revocation has been identified as the major bottleneck of state-of-the-art group signature schemes [MFG$^+$12].

Besides accountability and revocation, service providers are also interested in enhancing the efficiency of their services while still preserving the privacy of single users. For instance, service providers of public transport systems might be interested in analyzing the traveling patterns without compromising the privacy of passengers. In order to deal with these open challenges and to enable privacy-preserving data mining applications, so-called enhanced anonymity management mechanisms need to be researched and improved. We address these issues within the first part of this thesis.

Nevertheless, considering the implementation of privacy-preserving mechanisms, the system security of the employed platforms must also be taken into account. As history has taught us, even though specific mechanisms or primitives are considered as being secure and privacy preserving in theory, we have to cope with specific challenges in practice. A prominent example of such challenges are so-called side-channel attacks [Koc96] that exploit unintended information leakage of specific implementations or devices in order to extract secret information such as cryptographic keys.

Traditional side-channel attacks [MOP07] targeting smart cards usually require expensive equipment in order to gather the leaking information and to extract the secret information of these computing devices. In contrast, the ever increasing number of features present in today’s mobile devices, such as smartphones and tablet computers, already represent a plethora of side channels posing a significant threat to the users’ privacy and security. Even though modern operating systems employ specific mechanisms, such as sandboxing and

---

[1] “Anonymous data” seems to be hard to achieve. For instance, Narayanan and Shmatikov [NS06a] impressively demonstrated the de-anonymization of an individual’s record from an “anonymized” dataset of movie ratings published by Netflix. They conclude that any anonymized dataset can be de-anonymized again by relying on auxiliary information of individuals.

permission systems, to protect applications against each other and to protect sensitive resources, specific features as well as shared resources can be exploited to steal sensitive information from other applications (cf. [JS12, ZDH$^+$13]).

Although platform security itself is already a well-studied topic, it must be noted that smartphone security is different from traditional platform security as, for example, platform security in the context of personal desktop computers or smart cards. First, due to the mobility and portability of these devices, they are always turned on and carried around at all times and, thus, are also tightly integrated into our everyday lives. Second, additional software can be installed easily by means of established application markets and, hence, malware can also be spread easily and extremely fast. Third, these devices include far more features and sensors, such as an accelerometer, a gyroscope, a GPS sensor etc., which are not present on traditional platforms. Fourth, in order to decrease the overall number of devices carried around, employees tend to use their private devices to process corporate data and to access corporate infrastructure—also known as bring your own device (BYOD)—which clearly demonstrates the importance of secure computing platforms.

In a world where mobile and smart devices are tightly integrated into our everyday lives, we need to advance both privacy as well as security in order to pave the way for an environment that can be considered secure and privacy preserving. More specifically, in order to preserve the privacy of individuals we need to advance the field of privacy-preserving mechanisms. However, at the same time we also need to thoroughly study and understand security and privacy implications of the corresponding computing platforms. For instance, understanding security and privacy implications of specific features of smart and mobile devices is crucial.

1.1 Contributions and Outline

In this thesis, we aim to tackle the challenges of privacy and security. In Part I, we present an enhanced anonymity management mechanism for group signature schemes that also lends itself for an efficient revocation mechanism. In Part II, we perform investigations regarding the platform security of today's smartphones by focusing on side-channel attacks.

1.1.1 Enhancements for Group Signature Schemes

Chapter 2 provides an introduction to group signature schemes and outlines the principle of group signature schemes as well as the required preliminaries.

Chapter 3 introduces a promising generalization of a feature for group signatures denoted as controllable linkability, which has been published together with Daniel Slamanig and Thomas Unterluggauer [SSU14]. Controllable linkability allows an entity in possession of a trapdoor to link two signatures, *i.e.*, to determine whether two signatures have been produced by the same signer but it does not allow to identify the signers. In this paper [SSU14] we proposed a generic construction that allows to extend group signature schemes with controllable linkability. In order to enable such an extension we introduce a modification of all-or-nothing public key encryption with equality tests.

Chapter 4 demonstrates how to employ the concept of controllable linkability to implement the most efficient and flexible revocation mechanism for group signature schemes. The idea is that an always-online revocation authority can be contacted for the revocation check. The revocation authority performs the revocation check by anonymously linking a given signature against a revocation list. Furthermore, we introduce distributed controllable linkability such that multiple authorities must cooperate to link signatures, which reduces the trust assumption in these authorities. This work has been done together with Daniel Slamanig and Thomas Unterluggauer [SSU16].

1.1.2 Side-Channel Attacks on Mobile Devices

Chapter 5 starts with a general introduction to side-channel attacks. In an attempt to enable a more systematic investigation of side-channel attacks, we also refine the existing classification system in order to be applicable for modern side-channel attacks. Based on this classification system we survey existing side-channel attacks. This work has been done together with Veelasha Moonsamy, Thomas Korak, and Stefan Mangard [SMKM16].

Chapter 6 introduces our practical investigations of cache-based side-channel attacks on mobile devices. We investigate specific enhancements for Bernstein's cache-timing attack [Ber05] and we show that cache-based side-channel attacks indeed represent a real threat for today's smartphones and mobile platforms. More specifically, these enhancements demonstrate that cache-timing attacks can be used to significantly reduce the effective key size of the Advanced Encryption Standard (AES) on mobile devices. These investigations have been published together with Benoît Gérard [SG14].

Chapter 7 introduces a novel sensor-based attack that exploits the ambient-light sensor to infer a user's personal identification number (PIN) input. This is the first work demonstrating that—besides motion sensors—also ambient sensors can be used to infer sensitive user input. This work has been published in [Spr14].

Chapter 8 demonstrates how the official Android API and the `/proc` file system can be exploited in order to infer a user's browsing behavior. More specifically, we exploit the information leakage of the publicly available data-usage statistics on Android. While these statistics seem to be innocuous at first glance, we show that this information can be used to infer a user's visited websites even if the network traffic is routed through the anonymity network Tor. This work has been done together with Simone Griesmayr, Thomas Korak, and Stefan Mangard [SGKM16].

Chapter 9 concludes this thesis.

1.2 Other Contributions

Other scientific contributions and papers that were published during the PhD studies, but are not incorporated in this thesis, are as follows.

Group Signature Schemes and Privacy-Preserving Protocols

- Model and concrete instantiation for group signature schemes with verifiable controllable linkability [BDSS16]
- Considerations for a privacy-preserving protocol for automatic fare collection systems [ZGS$^+$15, ZSG$^+$16]
- Privacy-preserving authentication mechanism for IoT devices [GHSS15]
- Practical investigation of group signatures on resource-constrained devices [SS14]

Cache Attacks

- Investigation of Flush+Reload attacks on ARM [LGS$^+$16]
- Automated approach to launch Flush+Reload attacks [GSM15]
- Investigation of time-driven cache attacks on ARM [SP13b]
- Attack against disaligned AES T-tables on ARM [SP13a]

Part I

Enhancements for Group Signature Schemes

2

Group Signature Schemes

Birds of a feather flock together.
— Lewis Carrol, Alice in Wonderland

In this chapter, we introduce some background knowledge on group signatures. We start with an introduction to group signatures in Section 2.1 and discuss mathematical preliminaries in Section 2.2. Furthermore, we discuss public key encryption in Section 2.3, digital signatures in Section 2.4, and the principle of zero-knowledge proofs in Section 2.5. We recall the most popular security model for group signatures in Section 2.6, and the sign-and-encrypt-and-prove paradigm used to construct group signatures in Section 2.7. Finally, we recall the concept of threshold secret sharing in Section 2.8. Parts of this chapter are taken verbatim from [SSU14, SSU16].

Publication Data and Contribution

Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security – ISC 2014*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014

Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group Signatures with Linking-Based Revocation: A Pragmatic Approach for Efficient Revocation Checks. In *Conference on Cryptology & Malicious Security – Mycrypt 2016*, 2016. In press

Contribution: Main author; Initial idea together with Thomas Unterluggauer; Security proofs mainly developed by Daniel Slamanig.

2.1 Introduction

Two developments among today's society lead to two contrary requirements. On the one hand, users need to authenticate when interacting with service providers. In particular, service providers want to ensure that only genuine users can use their services. On the other hand, users want to protect their privacy and, thus, aim to reduce the amount of revealed personal information.[1] Indeed, in many scenarios like public transport or subscription-based services there is no need for a service provider to uniquely identify single users but only to ensure that they are allowed to use the respective service. In such scenarios, privacy-enhancing technologies (PETs) can be employed. One such popular cryptographic PET are group signature schemes.

The concept of group signature schemes (GSSs) has been introduced by Chaum and van Heyst [CvH91] and allows members within a predefined group to sign messages on behalf of the group anonymously. Thereby, signers implicitly prove membership affiliation anonymously. Verifiers in turn can determine whether a signature indeed has been produced by a member of the group, but are not able to determine the actual identity of a signer. However, in case of dispute the so-called group manager (GM) is able to open a given group signature (GS) in order to determine the identity of the actual signer.[2]

Early works of GSSs considered only a static setting [BMW03], where the group is fixed at the setup time, whereas more recent constructions consider dynamic groups [BSZ05, KY05], *i.e.*, new members can be added to the group and removed from the group over time. For both, the static as well as the dynamic setting, there exist constructions based on the sign-and-encrypt-and-prove (SEP) paradigm [CS97]. In schemes following this paradigm, a user on joining a group receives a signature from the GM, which is known as the membership certificate. A group signature produced by a user is then an encryption of the membership certificate of the user under the GM's public key and a non-interactive zero-knowledge proof of knowledge of well-formedness of the respective values.

Today, pairing-based group signature schemes (PB-GSSs) are prevalent, since they are far more efficient in terms of bandwidth and computational complexity than earlier constructions based on the strong RSA assumption. In the following subsections we introduce the required preliminaries for pairing-based group signature schemes following the SEP paradigm.

---

[1] Gürses [Gür10] defined three privacy paradigms: privacy as confidentiality, privacy as control, and privacy as practice. According to these definitions, we refer to privacy as "confidentiality" since users want to hide specific information from service providers.

[2] The role of the GM can be considered in the context of "privacy as control" [Gür10] because of the ability to define what happens with the data, *i.e.*, revealing the actual identity is possible. Although in this case it is the concept of group signatures that restricts access to this "confidential" data, *i.e.*, the identity of users, rather than the users themselves.

2.2 Mathematical Preliminaries

Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle \hat{g_2} \rangle$, and $\mathbb{G}_T$ be cyclic groups of prime order p. We write elements in $\mathbb{G}_1$ as g, h, etc., and we write elements in $\mathbb{G}_2$ as $\hat{g}$, $\hat{h}$, etc. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a map, such that the following properties hold:

1. Bilinearity: $e(u^a, \hat{v}^b) = e(u, \hat{v})^{ab}$ for all $u \in \mathbb{G}_1$, $\hat{v} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p^*$.
2. Non-degeneracy: $e(g_1, \hat{g_2}) \neq 1$.
3. Computability: e is efficiently computable.

Usually, $\mathbb{G}_1$ and $\mathbb{G}_2$ are subgroups of points on elliptic curves defined over a finite field $\mathbb{F}_q$ and an extension field $\mathbb{F}_{q^k}$, respectively. $\mathbb{G}_T$ usually represents a subgroup of the multiplicative group $\mathbb{F}_{q^k}^*$. The parameter q denotes the size of the underlying field and k denotes the embedding degree such that $p | q^k - 1$, with k minimal.

If $\mathbb{G}_1 = \mathbb{G}_2$, then e is called symmetric (Type 1) and asymmetric (Type 2 or Type 3) otherwise. For Type 2 pairings there exists an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$, whereas for Type 3 pairings no such efficient isomorphism is known.

Furthermore, a function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ is called negligible if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. In the remainder of this thesis, we use ϵ to denote such a negligible function.

For the sake of completeness, we state the following cryptographic assumptions that are referred to throughout this thesis.

Definition 1 (Computational Diffie-Hellman Assumption (CDH))**.** *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, such that $\log_2 p = \kappa$, and g a generator. We say that the CDH assumption holds, if for all probabilistic polynomial time (PPT) adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[r, s \xleftarrow{R} \mathbb{Z}_p, h \leftarrow \mathcal{A}(g, g^r, g^s) \ : \ h = g^{rs}\right] \leq \epsilon(\kappa).$$

Definition 2 (Computational co-Diffie-Hellman Assumption (co-CDH))**.** *Let $\mathbb{G}_1 = \langle g \rangle$, and $\mathbb{G}_2 = \langle \hat{g} \rangle$ be two cyclic groups of prime order p, such that $\log_2 p = \kappa$, and g and $\hat{g}$ their respective generators. We say that the co-CDH assumption holds, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[r \leftarrow \mathbb{Z}_p, \hat{h} \leftarrow \mathcal{A}(g, g^r, \hat{g}) \ : \ \hat{h} = \hat{g}^r\right] \leq \epsilon(\kappa).$$

Definition 3 (Computational co-Diffie-Hellman Assumption (co-CDH*) [CM11])**.** *Let $\mathbb{G}_1 = \langle g \rangle$, and $\mathbb{G}_2 = \langle \hat{g} \rangle$ be two cyclic groups of prime order p, such that $\log_2 p = \kappa$, and g and $\hat{g}$ their respective generators. We say that the co-CDH* assumption holds, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[r \leftarrow \mathbb{Z}_p, k \leftarrow \mathcal{A}(h, g, g^r, \hat{g}, \hat{g}^r) \ : \ k = h^r\right] \leq \epsilon(\kappa).$$

Definition 4 (Decisional Diffie-Hellman Assumption (DDH))**.** *Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p, such that $\log_2 p = \kappa$, and g a generator. We say that the DDH assumption holds, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{ll} b \xleftarrow{R} \{0,1\}, & \\ r,s,t \xleftarrow{R} \mathbb{Z}_p, & : \ b = b^* \\ b^* \leftarrow \mathcal{A}(g, g^r, g^s, g^{b\cdot(rs)+(1-b)\cdot t}) & \end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

Definition 5 (q-Strong Diffie-Hellman Assumption (q-SDH) [BB08])**.** *Let $\mathbb{G}_1 = \langle g \rangle$, and $\mathbb{G}_2 = \langle \hat{g} \rangle$ be two cyclic groups of prime order p, such that $\log_2 p = \kappa$, and g and $\hat{g}$ their respective generators in a Type 3 pairing setting. We say that the q-SDH assumption holds, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{ll} \gamma \xleftarrow{R} \mathbb{Z}_p, & \\ (x,h) \leftarrow \mathcal{A}(g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q}, \hat{g}, \hat{g}^{\gamma}) & : \ h = g^{\frac{1}{\gamma+x}} \end{array}\right] \leq \epsilon(\kappa).$$

Definition 6 ((Symmetric) eXternal Diffie-Hellman Assumption ((S)XDH))**.** *Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three distinct cyclic groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ a bilinear map. We say that the (S)XDH assumption holds, if the DDH assumption holds in $\mathbb{G}_1$ (and $\mathbb{G}_2$).*

Definition 7 (Decision LINear Assumption (DLIN) [BBS04])**.** *Let $\mathbb{G}_1 = \langle u \rangle = \langle v \rangle = \langle h \rangle$ be a cyclic group of prime order p, such that $\log_2 p = \kappa$, and u, v, h three generators. We say that the DLIN assumption holds, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{ll} b \xleftarrow{R} \{0,1\}, & \\ r,s,t \xleftarrow{R} \mathbb{Z}_p, & : \ b = b^* \\ b^* \leftarrow \mathcal{A}(u, v, h, u^r, v^s, h^{b\cdot(r+s)+(1-b)\cdot t}) & \end{array}\right] \leq \frac{1}{2} + \epsilon(\kappa).$$

2.3 Public Key Encryption

In a public key encryption (PKE) scheme, any entity can encrypt a message for a specific receiver by using the receiver's public key. In order to decrypt the message the receiver makes use of the corresponding private key. A conventional public key encryption scheme consists of the following PPT algorithms:

$\mathsf{KeyGen}(1^{\kappa})$: The key generation algorithm takes a security parameter κ, and generates a public-private key pair $(\mathsf{pk}, \mathsf{sk})$ used for encryption and decryption operations.

$\mathsf{Enc}(\mathsf{pk}, m)$: The encryption algorithm takes the public key pk, and a message m, and returns the encryption c of m under pk.

$\mathsf{Dec}(\mathsf{sk}, c)$: The decryption algorithm takes the private key sk, and a ciphertext c, and returns the message m.

Correctness of a public key encryption scheme is defined as follows.

Definition 8 (Correctness)**.** *A PKE scheme is correct, if it holds that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ m \xleftarrow{R} \mathcal{M}\end{array} \quad : \quad m = \mathsf{Dec}(\mathsf{sk},\mathsf{Enc}(\mathsf{pk},m))\right] = 1$$

where $\mathcal{M}$ represents the message space.

Weak security notions that we will refer to in this thesis are one-wayness under chosen plaintext attacks (OW-CPA) or one-wayness under chosen ciphertext attacks (OW-CCA). A stronger security notion requires a PKE scheme to be either indistinguishable under chosen plaintext attacks (IND-CPA) or indistinguishable under chosen ciphertext attacks (IND-CCA2).

Definition 9 (OW-CPA/OW-CCA security)**.** *A PKE scheme is* OW-CPA/OW-CCA *secure, if for all PPT adversaries $\mathcal{A}$ and security parameters κ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ m \xleftarrow{R} \mathcal{M}, c \leftarrow \mathsf{Enc}(\mathsf{pk},m),\\ m^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},c)\end{array} \quad : \quad m^* = m\right] \leq \epsilon(\kappa)$$

where $\mathcal{A}$ is restricted in case of OW-CPA *security, i.e., $\mathcal{O}(\cdot) = \perp$, and $\mathcal{A}$ is unrestricted in case of* OW-CCA *security, i.e., $\mathcal{O}(\cdot) = \mathcal{O}_{\mathsf{Dec}}$, and $\mathcal{A}$ has not queried the decryption oracle for the challenge ciphertext c.*

Definition 10 (IND-CPA/IND-CCA2 security)**.** *A* PKE *scheme is* IND-CPA/IND-CCA2 *secure, if for all PPT adversaries $\mathcal{A}$ and security parameters κ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa),\\ (m_0,m_1,s) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}),\\ b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{pk},m_b)\\ b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk},c,s)\end{array} \quad : \quad b^* = b\right] \leq \frac{1}{2} + \epsilon(\kappa)$$

where $m_0, m_1 \in \mathcal{M}$, with $\mathcal{M}$ representing the message space, and

$$\begin{array}{lll}\mathcal{O}(\cdot) & = \perp & \textit{for } \mathsf{IND\text{-}CPA}\\ \mathcal{O}(\cdot) & = \mathcal{O}_{\mathsf{Dec}} & \textit{for } \mathsf{IND\text{-}CCA2}\end{array}$$

and $\mathcal{O}_{\mathsf{Dec}}$ represents the decryption oracle, and $\mathcal{A}$ has not queried the decryption oracle for the challenge ciphertext c.

2.4 Digital Signature Scheme

In a digital signature (DS) scheme, entities can sign messages by using their private keys and any entity can verify the validity of a resulting signature by relying on the corresponding public key. A conventional digital signature scheme consists of the following PPT algorithms:

$\mathsf{KeyGen}(1^\kappa)$**:** The key generation algorithm takes a security parameter κ, and generates a public-private key pair $(\mathsf{pk}, \mathsf{sk})$ used for signature verification and generation operations.

$\mathsf{Sign}(\mathsf{sk}, m)$**:** The sign algorithm takes the private key sk, and a message m, and returns the signature s of m under sk.

$\mathsf{Verify}(\mathsf{pk}, s, m)$**:** The verification algorithm takes the public key pk, a signature s, and the corresponding message m, and returns 1 if s is a valid signature of m under sk and 0 otherwise.

Correctness of a digital signature scheme is defined as follows.

Definition 11 (Correctness)**.** *A DS scheme is said to be correct, if it holds that:*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ m \xleftarrow{R} \mathcal{M}\end{array} \quad : \quad \mathsf{Verify}(\mathsf{pk}, \mathsf{Sign}(\mathsf{sk}, m), m) = 1\right] = 1$$

where $\mathcal{M}$ represents the message space.

We require a DS scheme to be existentially unforgeable under chosen message attacks, *i.e.*, a DS scheme must be $\mathsf{EUF\text{-}CMA}$ secure.

Definition 12 ($\mathsf{EUF\text{-}CMA}$ security)**.** *A DS scheme is $\mathsf{EUF\text{-}CMA}$ secure, if for all PPT adversaries $\mathcal{A}$ and security parameters κ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\mathsf{Pr}\left[\begin{array}{l}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \\ (m, s) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk}),\end{array} \quad : \quad \mathsf{Verify}(\mathsf{pk}, s, m) = 1\right] \leq \epsilon(\kappa)$$

where $\mathcal{O}_{\mathsf{Sign}}$ represents the signing oracle, and $\mathcal{A}$ has not queried the signing oracle for message m.

2.5 Zero-Knowledge Proofs and Σ Protocols

A zero-knowledge proof is an interactive proof where a prover $\mathcal{P}$ convinces a verifier $\mathcal{V}$ of the validity of some statement, with the additional property that $\mathcal{V}$ does not learn any other information apart from the truth of the statement. Besides proving the truth of a statement x, we sometimes also want $\mathcal{P}$ to prove knowledge of the corresponding witness w, which is denoted as zero-knowledge proof of knowledge. For an element $x \in L$, w represents a witness if $(x, w) \in R$, where $L = \{x \mid \exists w : (x, w) \in R\}$ represents the corresponding language and R some relation.

In general, zero-knowledge proofs of knowledge considered in this thesis follow a 3-move protocol (also denoted as Σ protocol) between $\mathcal{P}$ and $\mathcal{V}$ as follows. Both $\mathcal{P}$ and $\mathcal{V}$ are given x, and $\mathcal{P}$ additionally knows a witness w such that $(x, w) \in R$. $\mathcal{P}$ sends an initial message $a \leftarrow \mathcal{P}(x, w)$, $\mathcal{V}$ responds with a challenge c, and $\mathcal{P}$ responds with $z \leftarrow \mathcal{P}(x, w, a, c)$. Finally, $\mathcal{V}$ decides whether or not to accept the proof based on the transcript (a, c, z) for x. Σ protocols yield highly efficient zero-knowledge proofs of knowledge. We recall the definition of a Σ protocol below (cf. [HL10]).

Definition 13 (Σ protocol)**.** *A 3-move protocol* Π *is called a* Σ *protocol for relation* R *if the following requirements hold:*

Completeness: *If* $\mathcal{P}$ *and* $\mathcal{V}$ *follow the protocol on input* x *and private input* w *to* $\mathcal{P}$ *such that* $(x, w) \in R$*, then* $\mathcal{V}$ *always accepts.*

Special soundness: *There exists a polynomial-time extractor* $\mathcal{E}$ *that given any* x *and any pair of accepting transcripts* (a, c, z)*,* (a, c', z') *for* x*, where* $c \neq c'$*, outputs* w *such that* $(x, w) \in R$*.*

Special honest verifier zero knowledge: *There exists a PPT simulator* $\mathcal{S}$ *that given* x *and* c *outputs a transcript* (a, c, z) *with the same probability distribution as transcripts between honest* $\mathcal{P}$ *and* $\mathcal{V}$ *on common input* x*.*

The above defined property of special soundness implies soundness, *i.e.*, a verifier cannot be tricked into accepting false statements. The group signature schemes considered in this thesis rely on non-interactive versions of Σ protocols. By relying on the Fiat-Shamir transform [FS86], where the challenge c sent by the verifier $\mathcal{V}$ is replaced with the evaluation of a cryptographic hash function on the first message a of the prover (and potentially x as well as other information), such honest verifier zero-knowledge proofs can be made non-interactive. Additionally, Σ protocols made non-interactive via the Fiat-Shamir paradigm are simulation sound [FKMV12], *i.e.*, soundness holds even after seeing simulated proofs for true as well as false statements.

2.6 Model for Dynamic Group Signatures

Bellare, Shi, and Zhang [BSZ05] presented a formal model for dynamic group signature schemes—denoted as BSZ model—that also includes formal security properties for such schemes. The BSZ model defines two different authorities: (1) an opening authority who can open signatures, and (2) an issuing authority who is capable of issuing signing keys to group members. We denote the keys of these authorities as master opening key (mok), and master issuing key (mik), respectively. Other involved parties are group members and verifiers.

GSSs usually refer to the group manager as an authority in charge of issuing new certificates and opening signatures. However, the secret key for issuing certificates (mik) and the secret key for opening signatures (mok) are separated and, thus, the group manager is just a logical authority whose responsibilities might be split across two physical authorities, e.g., the issuing authority and the opening authority.

A dynamic GSS is a tuple $\mathcal{GS} = (\mathsf{GkGen}, \mathsf{UkGen}, \mathsf{Join}, \mathsf{Issue}, \mathsf{GSig}, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge})$ of PPT algorithms which are defined as follows (cf. [BSZ05]):

$\mathsf{GkGen}(1^\kappa)$**:** On input a security parameter κ, the algorithm generates the public parameters, and outputs a tuple (gpk, mok, mik) representing the group public key, the master opening key, and the master issuing key.

$\mathsf{UkGen}(1^\kappa)$**:** On input a security parameter κ, the algorithm generates a user's key pair ($\mathsf{usk_i}$, $\mathsf{upk_i}$).

$\mathsf{Join}(\mathsf{usk_i}, \mathsf{upk_i})$**:** On input a user's key pair $(\mathsf{usk_i}, \mathsf{upk_i})$, the algorithm interacts with the Issue algorithm and outputs the group signing key $\mathsf{gsk_i}$ of user i.

$\mathsf{Issue}(\mathsf{gpk}, \mathsf{mik}, \mathbf{reg})$**:** On input of the group public key gpk, the master issuing key mik, and the registration table **reg**, the algorithm interacts with the Join algorithm in order to add user i to the group by adding an entry for user i to the registration table **reg**.

$\mathsf{GSig}(\mathsf{gpk}, M, \mathsf{gsk_i})$**:** On input of the group public key gpk, a message M, and a user's secret key $\mathsf{gsk_i}$, the algorithm outputs a group signature σ.

$\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$**:** On input of the group public key gpk, a message M, and a signature σ, the algorithm verifies whether the signature σ is valid with respect to the message M and the group public key gpk, and outputs `true` if the verification succeeds and `false` otherwise.

$\mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M, \sigma, \mathsf{mok})$**:** On input of the group public key gpk, the registration table **reg**, a message M, a valid signature σ, and the master opening key mok, the algorithm accesses the registration table **reg** to find the unique ID of the corresponding signer of σ. If a signer has been identified, the algorithm returns the ID of the signer and a publicly verifiable proof τ for the corresponding claim. Otherwise the algorithm claims that no group member produced σ.

$\mathsf{Judge}(\mathsf{gpk}, M, \sigma, ID, \mathsf{upk}, \tau)$**:** On input of the group public key gpk, a message M, a valid signature σ, the claimed ID of the corresponding signer of σ as well as the corresponding public key upk, and a proof τ that the claimed ID is indeed the signer of σ, the algorithm determines whether or not the proof τ holds. It outputs `true` if it holds and `false` otherwise.

Security Properties for GSSs. For a GSS to be secure according to the BSZ model it needs to satisfy the following properties. For a formal description of the properties defined in the BSZ model we refer to [BSZ05].

- **Correctness:** Signatures generated by honest group members should be valid according to the GVf algorithm, and the Open algorithm should correctly identify the signer. Furthermore, the proof returned by the Open algorithm should be accepted by the Judge algorithm.

- **Anonymity:** The identity of the signer can only be determined by the authority in possession of the master opening key mok.

- **Traceability:** An adversary should not be able to produce a signature that either cannot be opened by the opening authority, or the opening authority identifies a signer but cannot generate a correct proof for this claim.
- **Non-frameability:** No entity should be able to produce a correct proof that an honest user generated a signature unless this entity indeed produced this signature.

2.7 Sign-and-Encrypt-and-Prove Paradigm

Group signature schemes following the sign-and-encrypt-and-prove (SEP) [CS97, BSZ05, KY05] paradigm consist of the following three building blocks.

1. A secure public key encryption scheme $\mathcal{AE} = (\mathsf{KeyGen}_e, \mathsf{Enc}, \mathsf{Dec})$.
2. A secure digital signature scheme $\mathcal{DS} = (\mathsf{KeyGen}_s, \mathsf{Sign}, \mathsf{Verify})$.
3. Non-interactive zero-knowledge proofs of knowledge (NIZKPKs), e.g., honest verifier zero-knowledge proofs of knowledge made non-interactive using the Fiat-Shamir transform [FS86] in the random oracle model (denoted as signatures of knowledge (SoK) subsequently).

The group public key gpk consists of the public encryption key pk_e, and the signature verification key pk_s. The master opening key mok is the decryption key sk_e, and the master issuing key mik is the signing key sk_s. During the execution of the Join protocol, a user i generates a secret x_i and sends $f(x_i)$ to the issuer, where $f(\cdot)$ is a one-way function applied to a secret x_i. The issuer returns a signature $\mathsf{cert} \leftarrow \mathsf{Sign}(\mathsf{sk}_s, f(x_i))$ as the user's certificate by signing $f(x_i)$ with the signing key sk_s.

A group signature $\sigma = (T, \pi)$ for a message M consists of a ciphertext $T \leftarrow \mathsf{Enc}(\mathsf{pk}_e, \mathsf{cert})$ and the following SoK π:

$$\pi \leftarrow \mathsf{SoK}\{(x_i, \mathsf{cert}) : \mathsf{cert} = \mathsf{Sign}(\mathsf{sk}_s, f(x_i)) \quad \wedge \quad T = \mathsf{Enc}(\mathsf{pk}_e, \mathsf{cert})\}(M).$$

This notation for signatures of knowledge has been introduced by Camenisch and Stadler [CS97], *i.e.*, SoK denotes a signature of knowledge of a pair (x_i, cert), which satisfies the relations after the colon, on the message M.

The membership certificate typically refers to the issuer's signature (cert) but might also be a commitment to a user's secret that has been signed by the issuer. Nevertheless, we always refer to T as an encryption of the membership certificate.

2.8 Threshold Secret Sharing

A (t, n)-threshold secret sharing scheme allows to distribute a secret s to n parties in a way such that it requires the cooperation of at least t parties to recover s, while any set of up to $t-1$ parties learns nothing about s. A simple and famous (t, n)-threshold scheme based on polynomial interpolation has been proposed by Shamir [Sha79]. Here, to share a secret s, one chooses a random polynomial $F(x) = s + \sum_{\ell=1}^{t-1} a_\ell \cdot x^\ell$ of degree $t-1$ over some finite field such that $F(0) = s$. Shares are of the form $(i, F(i))$ and any subset of size at most $t-1$ will learn no information about s. Given shares $(i, F(i)) \in \mathcal{I}$ such that $|\mathcal{I}| \geq t$, s can however be efficiently recovered via $s = F(0) = \sum_{i \in \mathcal{I}} c_i \cdot F(i)$, where $c_i = \prod_{j \in \mathcal{I}, j \neq i} \frac{j}{j-i}$ are the corresponding Lagrange coefficients. We will use this scheme to share a group element from a prime order group (cf. [BZ04]).

3

Controllable Linkability

Great results, can be achieved with small forces.
— Sun Tzu, The Art of War

In this chapter, we show how PB-GSSs based on the SEP paradigm that are secure in the BSZ model can be generically transformed to support controllable linkability. Basically, this transformation replaces the public key encryption scheme used for identity escrow within a group signature scheme with a modified all-or-nothing public key encryption with equality tests scheme (denoted AoN-PKEET*), instantiated from the respective public key encryption scheme. Our suggested AoN-PKEET* mechanism may also be of independent interest for other applications, but we use it to achieve controllable linkability. The corresponding trapdoor is given to the linking authority as a linking key, which allows to perform trapdoor-equality tests on the ciphertexts (being part of the group signature) without learning the respective plaintexts. If the underlying encryption scheme is IND-CCA2 secure, controllable linkability can be added for free, *i.e.*, it neither influences the signature size nor the computational costs for signers and verifiers in comparison to the scheme without this feature.

We start with a motivation for controllable linkability in Section 3.1 and we recall an adapted security model for this feature in Section 3.2. We propose our AoN-PKEET* scheme in Section 3.3 and we show how to apply it to GSSs in Section 3.4. We compare controllable linkability to similar enhanced anonymity management mechanisms in Section 3.5. Parts of this chapter are taken verbatim from [SSU14].

Publication Data and Contribution

Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security – ISC 2014*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014

Contribution: Main author; Initial idea together with Thomas Unterluggauer; Daniel Slamanig suggested to generalize this concept; Security proofs mainly developed by Daniel Slamanig.

3.1 Introduction

Over the years, various approaches for enhanced anonymity management mechanisms for group signatures, besides the standard opening feature, have been proposed. For example, in various applications and scenarios such as direct anonymous attestation (DAA) [BCC04] or data mining applications it may be desirable to link different signatures of the same anonymous signer under certain circumstances. Clearly, a naive approach to realize such a feature is to contact the group manager, who can, given two signatures, open both signatures and decide whether they have been produced by the same signer. However, involving the group manager for such tasks may not be desirable and, thus, extensions are required that allow to answer such questions without the group manager. Such an extension might allow to link signatures either publicly or by means of another dedicated but less powerful entity, *i.e.*, an entity that does not need to be in possession of the secret key of the group manager. More specifically, some schemes allow to publicly link signatures of users without identifying them [NFW99], or even allow the public tracing of signers who have produced a number of signatures above a certain threshold [Wei05].

Besides these authority-free linking approaches, there also exist schemes, and in particular the schemes proposed by Hwang et al. [HLC$^+$11, HLC$^+$13] that support controllable linkability. In these schemes, there exist dedicated linking authorities (LAs), *i.e.*, parties in possession of a so-called linking key, who are able to link two signatures by means of this key, whereas others are not able to do so. A LA thereby can only decide whether two given signatures have been issued by the same *unknown* signer and, thus, signers stay anonymous.

Hwang et al. [HLC$^+$11, HLC$^+$13] argued that the concept of controllable linkability might be useful in vehicular ad hoc networks (VANETs), for example, to prevent Sybil attacks by detecting multiple signatures from the same entity. Furthermore, this concept can be beneficial in the context of data mining, such that service providers can establish statistics regarding buying patterns, while still preserving the privacy of customers. Besides, controllable linkability can also be useful in the context of smart cities. For example, public transport systems can support anonymous traveling where a valid group signature represents showing of a valid ticket. However, service providers might also be interested in analyzing traveling patterns and to perform some kind of flow control analysis. Thereby, the concept of controllable linkability allows the service provider to efficiently link signatures (ticket showings), while passengers of the public transport system still remain anonymous.

The techniques which are the basis for our generic approach to controllable linkability have been inspired by the works of Hwang et al. [HLC$^+$11, HLC$^+$13], who realize controllable linkability for their variation of the BBS$^+$ GSS, *i.e.*, a membership certificate is a BBS$^+$ signature [ASM06]. However, their approach is tailored towards their variation of the BBS$^+$ GSS, meaning that they neither make their design intuition explicit nor do they rely on a general building block. In contrast, our approach relies on the general building block of the newly introduced AoN-PKEET* and is independent from a particular GSS.

3.2 Model for GSs with Controllable Linkability

Hwang et al. [HLC$^+$11, HLC$^+$13] introduced a model for GSSs with controllable linkability that builds upon the well-established BSZ model (cf. Section 2.6). Although, Hwang et al. use a weaker notion of anonymity where the adversary does not have access to an open oracle.

While the BSZ model logically splits the group manager into (1) an opening authority, and (2) an issuing authority, Hwang et al. add (3) a *linking authority* capable of linking signatures. This linking authority can determine whether or not two signatures have been issued by the same anonymous signer. We denote the keys of these authorities as master opening key (mok), master issuing key (mik), and master linking key (mlk), respectively. A GSS with controllable linkability is specified as a tuple $\mathcal{GS}$ = (GkGen, UkGen, Join, Issue, GSig, GVf, Open, Judge, Link). Subsequently, we present the GkGen algorithm, as it changes due to these modifications, as well as the additional Link algorithm.

$\mathsf{GkGen}(1^\kappa)$: On input a security parameter κ, the algorithm generates the public parameters and outputs a tuple (gpk, mok, mik, mlk), representing the group public key, the master opening key, the master issuing key, and the master linking key.

$\mathsf{Link}(\mathsf{gpk}, M, \sigma, M', \sigma', \mathsf{mlk})$: On input of the group public key gpk, a message M and a corresponding signature σ, another message M' and a corresponding signature σ', as well as the master linking key mlk, the algorithm verifies both signatures by calling $\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$ and $\mathsf{GVf}(\mathsf{gpk}, M', \sigma')$. If both signatures are valid for messages M and M' under the group public key gpk, the algorithm uses mlk to determine whether σ and σ' have been produced by the same *unknown* signer. If both signatures are valid and can be linked to the same unknown signer, the algorithm returns `true` and `false` otherwise.

Properties for GSSs with Controllable Linkability. Hwang et al. also adapt the correctness property (cf. Section 2.6) and introduce new properties to model the newly introduced feature of controllable linkability.

- **Correctness**: Signatures generated by honest group members should be valid, the Open algorithm should correctly identify the signer, and the proof returned by the Open algorithm should be accepted by the Judge algorithm. Furthermore, the Link algorithm should correctly link two signatures from the same unknown signer.

- **Linkability**: The authority in possession of the master linking key mlk should neither be able to gain any useful information for opening a signature (*link-only linkability*) nor for generating a Judge proof τ (*judge-proof unforgeability*). Furthermore, colluding parties—including users, the linking authority, and/or the opening authority—should not be able to generate pairs of messages and signatures (M, σ) and (M', σ') that violate the correctness property mentioned above (*enforced linkability*).

3.3 Public Key Encryption and Equality Tests

In Section 3.3.1, we discuss how public key encryption is used in GSSs following the SEP paradigm. In Section 3.3.2, we recall existing public key encryption schemes supporting trapdoor equality tests. Finally, we introduce our modified primitive to be applicable for controllable linkability in Section 3.3.3.

3.3.1 Public Key Encryption in GSSs

In PB-GSSs following the SEP paradigm, the used encryption scheme depends on the bilinear map setting, *i.e.*, the type of the pairing, and whether the construction targets to achieve weak or full anonymity. The former issue is concerned with the DDH problem in the used groups. If one assumes the XDH assumption to hold in the group used for encryption, *i.e.*, DDH is assumed to be hard in the group $\mathbb{G}_1$, then one can use standard IND-CPA secure ElGamal encryption. If one assumes the DDH problem to be easy in $\mathbb{G}_1$ and $\mathbb{G}_2$, then one usually relies on linear encryption variants [BBS04, HLC$^+$13] of ElGamal encryption that are IND-CPA secure under the DLIN (or some related) assumption.

Whether the construction targets weak or full anonymity depends on whether or not the adversary is allowed to access an Open oracle, *i.e.*, a decryption oracle. Weak anonymity means that anonymity is guaranteed as long as the adversary does not have access to an Open oracle. Hence, the use of IND-CPA secure encryption schemes yields weak anonymous group signatures (CPA-full-anonymity). For full anonymity (CCA-full-anonymity), *i.e.*, anonymity that holds even in case the adversary has access to an Open oracle, one needs to rely on IND-CCA2 secure encryption schemes. For example, Delerablée and Pointcheval [DP06] tweak the weak anonymous BBS variant [BBS04] by replacing linear ElGamal with standard IND-CPA secure ElGamal (relying on the XDH assumption) and turn IND-CPA secure ElGamal into an IND-CCA2 secure encryption scheme using the twin encryption paradigm [NY90, RS91] to achieve CCA-full-anonymity.

3.3.2 Trapdoor Equality Test for Public-Key Encryption

At the heart of our generic construction to achieve controllable linkability is a means to extend IND-CPA/IND-CCA2 secure public key encryption schemes with a feature that allows a dedicated party (holding a trapdoor) to check whether two ciphertexts under the same public key contain the same message, but without being able to decrypt ciphertexts, *i.e.*, still providing one-wayness (OW) against trapdoor holders. A naive approach would be to give away the private key as a trapdoor and for two given ciphertexts one could simply decrypt and compare the plaintexts. However, this would allow to recover encrypted messages, which is not desired as it would give too much power to the linking authority.

Our idea is related to the concept of probabilistic public key encryption with equality tests (PKEET) [YTHW10], but differs in that their equality tests are public, *i.e.*, not only feasible for one holding a trapdoor, and need to work for ciphertexts under different public keys. Consequently, their construction does not satisfy any meaningful notion of indistinguishability. However, in our approach indistinguishability supported by the underlying encryption scheme still needs to hold for all parties except the one holding the trapdoor, who clearly can test against any possible message. However, if the messages are not guessable, *i.e.*, the messages are randomly sampled from a message space that has large enough min entropy, messages can still be hidden from the party holding the trapdoor and, thus, provide OW-CPA security.

Tang [Tan12b] added an authorization mechanism to PKEET—denoted as all-or-nothing PKEET (AoN-PKEET)—in order to authorize entities to perform plaintext equality tests. This idea is related to our idea, but their focus is on allowing a semi-trusted proxy to compare ciphertexts for two distinct parties by obtaining a trapdoor from each party. This approach also works if the two users are identical, in which case only one trapdoor is required. However, firstly they target applications for searchable encryption and consequently their Type-I adversary (representing the proxy holding the trapdoor(s)) is very powerful, *i.e.*, they require OW-CCA security, and also for an outsider who does not know the trapdoor(s) (Type-II adversary), they always require IND-CCA2 security. Secondly, due to their focus on comparison of ciphertexts from distinct users, their construction is quite involved and requires a quite costly variant of double encryption for each user. Besides inefficiency, the most important difference between the AoN-PKEET construction and our approach is that we need compatibility with efficient proofs of knowledge of encrypted messages, which are not possible in [Tan12a, Tan12b] as these instantiations involve encrypting hashes of messages. Thus, these approaches are not applicable to our setting as it cannot be efficiently proven that the correct hash of the message (membership certificate) has been included by the user and thus identity escrow is not that efficiently possible [JKO13] to be of practical relevance for GSSs.

3.3.3 Modified All-Or-Nothing PKE with Equality Tests

In brief, our approach can be seen as a restricted version of AoN-PKEET, as we only allow comparison of ciphertexts of the same user, *i.e.*, under the same public key. Furthermore, against a Type-I adversary (the trapdoor holder) we do not require $\mathsf{OW\text{-}CCA}$ but $\mathsf{OW\text{-}CPA}$ security[1] and against Type-II adversaries (outsiders) either $\mathsf{IND\text{-}CPA}$ or $\mathsf{IND\text{-}CCA2}$ security (depending on the underlying public key encryption scheme). Additionally, our construction requires efficient zero-knowledge proofs of knowledge about messages encrypted in ciphertexts, which is clearly true for all encryption schemes used in PB-GSSs following the SEP paradigm. As already mentioned, the constructions by Tang [Tan12a, Tan12b] do not allow such efficient proofs of knowledge of encrypted messages, and thus renders these approaches not applicable to our setting.

We denote our modified scheme as an AoN-PKEET* scheme. Subsequently, we first present the formal model, and then discuss possible instantiations.

Formal Model

An AoN-PKEET* scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aut}, \mathsf{Com})$ is a conventional public key encryption scheme augmented by two PPT algorithms $(\mathsf{Aut}, \mathsf{Com})$ as follows.

$\mathsf{Aut}(\mathsf{sk})$: The authorization algorithm takes a private key sk, and outputs a trapdoor tk that allows to perform the equality tests.

$\mathsf{Com}(c,c',\mathsf{tk})$: The comparison algorithm takes two ciphertexts c and c' (of two messages m and m') produced under pk, and a trapdoor tk produced with the corresponding sk, and outputs `true` if $m = m'$ and `false` otherwise.

Security Definition

For our modified setting, the *soundness* definition of [Tan12b] reduces to the fact that besides correctness of the public key encryption scheme, we have that for all $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa)$ we require that $\mathsf{Com}(\mathsf{Enc}(\mathsf{pk}, m), \mathsf{Enc}(\mathsf{pk}, m'), \mathsf{Aut}(\mathsf{sk})) =$ `true` if and only if $m = m'$. Against a Type-I adversary, we require $\mathsf{OW\text{-}CPA}$ security, which is defined as follows.

Definition 14 ($\mathsf{OW\text{-}CPA}$ security)**.** *An* AoN-PKEET* *scheme is* $\mathsf{OW\text{-}CPA}$ *secure, if for all PPT adversaries* $\mathcal{A}$ *and security parameters* κ *there is a negligible function* $\epsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{ll} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{sk}), & \\ m \xleftarrow{R} \mathcal{M}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, m), & : \quad m^* = m \\ m^* \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{tk}, c) & \end{array}\right] \leq \epsilon(\kappa).$$

Against a Type-II adversary, we require either $\mathsf{IND\text{-}CPA}$ or $\mathsf{IND\text{-}CCA2}$ security (depending on the used public key encryption scheme) as it is defined for conventional public key encryption schemes.

---

[1] Assuming that the opener cannot be used as a decryption oracle is reasonable, however, we may also extend the approach to $\mathsf{OW\text{-}CCA}$ security.

Definition 15 ($\mathsf{IND\text{-}CPA/IND\text{-}CCA2}$ security)**.** *An* AoN-PKEET* *scheme is* $\mathsf{IND\text{-}CPA/IND\text{-}CCA2}$ *secure, if for all PPT adversaries* $\mathcal{A}$ *and security parameters* κ *there is a negligible function* $\epsilon(\cdot)$ *such that:*

$$\Pr\left[\begin{array}{l}(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{sk}),\\ (m_0, m_1, s) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}),\\ b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\\ b^* \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pk}, c, s)\end{array} \quad : \quad b^* = b\right] \leq \frac{1}{2} + \epsilon(\kappa)$$

where $m_0, m_1 \in \mathcal{M}$*, and*

$$\begin{array}{lll}\mathcal{O}(\cdot) & = \perp & \textit{for } \mathsf{IND\text{-}CPA}\\ \mathcal{O}(\cdot) & = \mathcal{O}_{\mathsf{Dec}} & \textit{for } \mathsf{IND\text{-}CCA2}\end{array}$$

and $\mathcal{O}_{\mathsf{Dec}}$ *represents the decryption oracle, and* $\mathcal{A}$ *has not queried the decryption oracle for the challenge ciphertext* c*.*

Definition 16. *An AoN-PKEET* scheme is called secure if it is sound, provides* $\mathsf{OW\text{-}CPA}$ *security against Type-I adversaries (trapdoor holders) and if the underlying encryption scheme provides* $\mathsf{IND\text{-}CPA/IND\text{-}CCA2}$ *security against Type-II adversaries (outsiders).*

Constructions

Subsequently, we elaborate how an AoN-PKEET* scheme can be instantiated with various public key encryption schemes that are often used in PB-GSSs.

ElGamal Encryption. We consider ElGamal encryption in $\mathbb{G}_1$ in a bilinear map setting $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that the DDH assumption holds in $\mathbb{G}_1$. Let the private key be $\xi \in \mathbb{Z}_p^*$, the public key be $h \leftarrow g^\xi \in \mathbb{G}_1$, and a ciphertext for a message $m \in \mathbb{G}_1$ be $(T_1, T_2) \leftarrow (g^\alpha, mh^\alpha) \in \mathbb{G}_1^2$ for random $\alpha \in \mathbb{Z}_p^*$. Decryption works by computing $m \leftarrow T_2/(T_1^\xi)$. The algorithms Aut and Com work as follows.

$\mathsf{Aut}(\xi)$**:** Given the secret key ξ, the trapdoor is computed as $\mathsf{tk} \leftarrow (\hat{r}, \hat{t} = \hat{r}^\xi)$ for a random $\hat{r} \in \mathbb{G}_2$.

$\mathsf{Com}(T, T', \mathsf{tk})$**:** Given a ciphertext $(T_1, T_2) = (g^\alpha, mh^\alpha)$, another ciphertext $(T_1', T_2') = (g^{\alpha'}, m'h^{\alpha'})$, and a trapdoor $\mathsf{tk} = (\hat{r}, \hat{t} = \hat{r}^\xi)$ check:

$$\frac{e(T_2, \hat{r})}{e(T_1, \hat{t})} \stackrel{?}{=} \frac{e(T_2', \hat{r})}{e(T_1', \hat{t})}.$$

If the check holds, *i.e.*, we have $e(m, \hat{r}) = e(m', \hat{r})$, then return `true` and `false` otherwise.

Linear Encryption. We consider linear encryption [BBS04] in $\mathbb{G}_1$ in a bilinear map setting $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that the DDH problem may be easy in $\mathbb{G}_1$ as well as $\mathbb{G}_2$ but the DLIN assumption holds. Let the private key be $(\xi, \mu) \in (\mathbb{Z}_p^*)^2$ and the public key be $(u, v, h) \in \mathbb{G}_1^3$ such that $u^\xi = v^\mu = h$. Therefore, choose a random h and compute $u \leftarrow h^{1/\xi}$ and $v \leftarrow h^{1/\mu}$. A ciphertext for a message $m \in \mathbb{G}_1$ is $(T_1, T_2, T_3) \leftarrow (u^\alpha, v^\beta, mh^{\alpha+\beta}) \in \mathbb{G}_1^3$ for random $\alpha, \beta \in \mathbb{Z}_p^*$. Decryption works in $\mathbb{G}_1$ by computing $m \leftarrow T_3/(T_1^\xi T_2^\mu)$. The algorithms Aut and Com work as follows.

$\mathsf{Aut}(\xi, \mu)$**:** Given the secret key (ξ, μ), the trapdoor is computed as $\mathsf{tk} \leftarrow (\hat{r}, \hat{s} = \hat{r}^\xi, \hat{t} = \hat{r}^\mu)$ for a random $\hat{r} \in \mathbb{G}_2$.

$\mathsf{Com}(T, T', \mathsf{tk})$**:** Given a ciphertext $(T_1, T_2, T_3) = (u^\alpha, v^\beta, mh^{\alpha+\beta})$, another ciphertext $(T_1', T_2', T_3') = (u^{\alpha'}, v^{\beta'}, m'h^{\alpha'+\beta'})$, and a trapdoor $\mathsf{tk} = (\hat{r}, \hat{s} = \hat{r}^\xi, \hat{t} = \hat{r}^\mu)$ check:

$$\frac{e(T_3, \hat{r})}{e(T_1, \hat{s}) \cdot e(T_2, \hat{t})} \overset{?}{=} \frac{e(T_3', \hat{r})}{e(T_1', \hat{s}) \cdot e(T_2', \hat{t})}.$$

If the check holds, *i.e.*, we have $e(m, \hat{r}) = e(m', \hat{r})$, then return `true` and `false` otherwise.

Security of the Constructions

We analyze AoN-PKEET* instantiated with ElGamal encryption under the (S)XDH assumption and with linear encryption under the DLIN assumption below. We rely on the use of Type 2 pairings ($\mathbb{G}_1 \neq \mathbb{G}_2$ with an efficient isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$), Type 3 pairings ($\mathbb{G}_1 \neq \mathbb{G}_2$ without an efficient isomorphism), and Type 1 pairings ($\mathbb{G}_1 = \mathbb{G}_2$), respectively.

Lemma 1. *AoN-PKEET* based on ElGamal in $\mathbb{G}_1$ in an XDH setting (Type 2 setting) is secure under the co-CDH assumption.*

Proof (Lemma 1). Obviously, AoN-PKEET* based on ElGamal encryption satisfies the soundness property and is $\mathsf{IND\text{-}CPA}$ secure against Type-II adversaries. What remains to be argued about is the $\mathsf{OW\text{-}CPA}$ security against Type-I adversaries. Given an adversary $\mathcal{A}$ that breaks $\mathsf{OW\text{-}CPA}$ security, we show how to construct an adversary $\mathcal{B}$ against the co-CDH problem. Let $(g_1, g_1^a, \hat{g_2}, \hat{g_2}^b)$ be a co-CDH problem instance given to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk} \leftarrow \psi(\hat{g_2}^b)$ and $\mathsf{tk} \leftarrow (\hat{g_2}^w, (\hat{g_2}^b)^w)$ for a random $w \in \mathbb{Z}_p^*$, chooses a random $h \in \mathbb{G}_1$, and runs $\mathcal{A}$ on pk and $c \leftarrow (g_1^a, h) \in \mathbb{G}_1^2$. If $\mathcal{A}$ manages to output $m^* = h/g_1^{ab}$, then $\mathcal{B}$ outputs $h/m^* = g_1^{ab}$ which is a valid solution to the co-CDH problem instance. $\square$

Lemma 2. *AoN-PKEET* based on ElGamal in $\mathbb{G}_1$ in an SXDH setting (Type 3 setting) is secure under the co-CDH* assumption.*

Proof (Lemma 2). Again, AoN-PKEET* based on ElGamal encryption satisfies the soundness property and is $\mathsf{IND\text{-}CPA}$ secure against Type-II adversaries.

What remains to be argued about is the $\mathsf{OW\text{-}CPA}$ security against Type-I adversaries. Given an adversary $\mathcal{A}$ that breaks $\mathsf{OW\text{-}CPA}$ security, we show how to construct an adversary $\mathcal{B}$ against the co-CDH* problem. Let $(g_1, g_1^a, g_1^b, \hat{g_2}, \hat{g_2}^b)$ be a co-CDH* problem instance given to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk} \leftarrow g_1^b$ and $\mathsf{tk} \leftarrow (\hat{g_2}^w, (\hat{g_2}^b)^w)$ for a random $w \in \mathbb{Z}_p^*$, chooses a random $h \in \mathbb{G}_1$, and runs $\mathcal{A}$ on pk and $c \leftarrow (g_1^a, h) \in \mathbb{G}_1^2$. If $\mathcal{A}$ manages to output $m^* = h/g_1^{ab}$, then $\mathcal{B}$ outputs $h/m^* = g_1^{ab}$ which is a valid solution to the co-CDH* problem instance. ☐

In addition, for the security of this construction we also need to consider the value $\mathsf{t} = e(m, \hat{r})$ where $\hat{r} = \hat{g_2}^w$ is available to $\mathcal{A}$. Therefore, note that computing $m \in \mathbb{G}_1$ when given the values $\mathsf{t} \in \mathbb{G}_T$ and $\hat{r} \in \mathbb{G}_2$ is the fixed argument pairing inversion 2 (FAPI-2) problem and an adversary against the FAPI-2 problem implies an adversary against the CDH problem [GHV08] and the co-CDH problem in the ElGamal-based Type 2 setting above.

Lemma 3. *AoN-PKEET* based on linear encryption in* $\mathbb{G}_1$ *is secure under the CDH assumption.*

Proof (Lemma 3). Obviously, AoN-PKEET* based on linear encryption satisfies the soundness property and is $\mathsf{IND\text{-}CPA}$ secure against Type-II adversaries. What remains to be argued about is the $\mathsf{OW\text{-}CPA}$ security against Type-I adversaries. Given an adversary $\mathcal{A}$ that breaks $\mathsf{OW\text{-}CPA}$ security, we show how to construct an adversary $\mathcal{B}$ against the CDH problem in $\mathbb{G}_1$. Let (g, g^a, g^b) be a CDH problem instance given to $\mathcal{B}$. Unlike the ElGamal case, $\mathcal{A}$ can verify whether $\mathsf{pk} = (u, v, h)$ and $\mathsf{tk} = (\hat{r}, \hat{r}^\xi, \hat{r}^\mu)$ are consistent, *i.e.*, by checking whether $e(u, \hat{r}^\xi) = e(v, \hat{r}^\mu) = e(h, \hat{r})$ holds. Thus, $\mathcal{B}$ randomly chooses $x, y, z \in \mathbb{Z}_p^*$ and sets $\mathsf{pk} \leftarrow (u, v, h) = (g^x, g^y, g^b)$ as well as $\mathsf{tk} \leftarrow (g^z, h^{z/x}, h^{z/y})$, chooses a random $k \in \mathbb{G}_1$ and runs $\mathcal{A}$ on pk and $c \leftarrow ((g^a)^x, v^\beta, k)$ for random $\beta \in \mathbb{Z}_p^*$. Note that for $\mathcal{A}$ this simulation is indistinguishable from the real game. If $\mathcal{A}$ manages to output $m^* = k/h^{\alpha+\beta}$, then $\mathcal{B}$ outputs $(k/m^*)/(h^\beta) = h^\alpha = g^{ab}$ which is a valid solution to the CDH problem instance. ☐

Below we briefly mention other schemes that may also be used as the underlying encryption scheme for AoN-PKEET* schemes.

Double ElGamal Encryption: Some GSSs [NS04b, DP06] rely on double ElGamal encryption, which relies on the Naor-Yung paradigm [NY90] to transform $\mathsf{IND\text{-}CPA}$ secure ElGamal into an $\mathsf{IND\text{-}CCA2}$ secure variant in the random oracle model. The idea is to encrypt a message m twice under two independent public keys $(h_1 = g^{\xi_1})$ and $(h_2 = g^{\xi_2})$ using independent random coins α and β to obtain $(T_1, T_2) \leftarrow (g^\alpha, mh_1^\alpha)$ and $(T_3, T_4) \leftarrow (g^\beta, mh_2^\beta)$. Then, one computes a non-interactive zero-knowledge proof of knowledge (via Fiat-Shamir) that the two ciphertexts (T_1, T_2) and (T_3, T_4) contain the same message m, which can be realized by providing the following proof of knowledge (PoK) of the values (α, β):

$$\mathsf{PoK}\{(\alpha, \beta) : T_1 = g^\alpha \wedge T_3 = g^\beta \wedge T_2/T_4 = h_1^\alpha / h_2^\beta\}.$$

When used in a bilinear map setting (and assuming XDH as done in [DP06]), the trapdoor equality test works analogously to standard ElGamal and the trapdoor can either be $\mathsf{tk} = (\hat{r}, \hat{t} = \hat{r}^{\xi_1})$ or $\mathsf{tk} = (\hat{r}, \hat{t} = \hat{r}^{\xi_2})$.

Double Linear Encryption: The same approach used to turn standard ElGamal into an IND-CCA2 secure variant in the random oracle model can also be applied to linear encryption (when one does not want to rely on the XDH assumption) and the trapdoor equality test works analogously.

Cramer-Shoup Encryption: Our trapdoor equality test also works for the IND-CCA2 secure Cramer-Shoup encryption scheme [CS98], as the first and the third element of the ciphertext form a standard ElGamal ciphertext.

3.4 Adding Controllable Linkability to PB-GSSs

In order to generically add controllable linkability to PB-GSSs following the SEP paradigm, we replace the used public key encryption scheme with its AoN-PKEET* version. Then, the additionally required linking key mlk is the trapdoor tk computed by Aut, which is given to the LA. When given two message-signature pairs (M, σ) and (M', σ'), where the signatures $\sigma = (T, \pi)$ and $\sigma' = (T', \pi')$ contain ciphertexts T and T' as well as the non-interactive proofs π and π', the LA runs $\mathsf{Com}(T, T', \mathsf{mlk})$ to decide whether the two signatures have been produced by the same *unknown* signer.

In order to convert a PB-GSS $\mathcal{GS}$ = (GkGen, UkGen, Join, Issue, GSig, GVf, Open, Judge) following the SEP paradigm into a PB-GSS with controllable linkability, we have to add a linking authority (LA) as well as an additional algorithm Link (cf. Section 3.2). Below we present the required modifications to the GkGen algorithm as well as the Link algorithm:

$\mathsf{GkGen}(1^\kappa)$**:** On input a security parameter κ, the algorithm generates the public parameters and outputs a tuple $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk})$. Therefore, it runs $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KeyGen}(1^\kappa)$ of the AoN-PKEET* scheme, sets $\mathsf{mok} = \mathsf{sk}_e$, and integrates pk_e into gpk. Finally, it computes the trapdoor $\mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{mok})$, and sets $\mathsf{mlk} = \mathsf{tk}$. All remaining steps remain unchanged.

$\mathsf{Link}(\mathsf{gpk}, M, \sigma, M', \sigma', \mathsf{mlk})$**:** On input of the group public key gpk, a message M and a corresponding signature σ, another message M' and a corresponding signature σ', as well as the master linking key mlk, the algorithm first verifies both signatures via the GVf algorithm. If any of these two verifications fails, the algorithm outputs `false`. Otherwise, the algorithm extracts the ciphertexts T and T' from σ and σ', runs $\mathsf{Com}(T, T', \mathsf{mlk})$, and outputs whatever Com outputs.

Irrespective of the actual value that is being encrypted in the context of group signatures, we always refer to this value as an encryption of the membership certificate. In order to prevent the LA from efficiently guessing messages and thereby to link the ciphertexts of (guessed) messages against unknown ciphertext messages, we rely on the following assumption.

Assumption 1. *Honestly computed membership certificates (used for identity escrow) of users are uniformly distributed over the respective group and are unknown to the linking authority.*[2]

For all GSSs based on BBS variants [BBS04], membership certificates are of the form $A = g_1^{\frac{1}{x_i+\mathsf{mik}}}$ where g_1 may be the product of other group elements representing a BB signature [BB04]. In [DY05] it is shown that this represents a verifiable random function (VRF). Basically, in such a setting an adversary controlling the input should not be able to distinguish the output of the VRF from uniformly sampled strings of equal size. However, in our setting we do not want to realize a pseudo-random function, but we only require that A is uniformly distributed over the group for uniformly at random sampled messages (that are not known to the linking authority). In case of BB signatures, A will be a random element of the group if the unknown value of x_i is chosen uniformly at random from the integers in the order of the group.

The CL GSS proposed by Camenisch and Lysyanskaya [CL04] uses Cramer-Shoup encryption in $\mathbb{G}_T$ as they rely on Type 1 pairings, which would not work with our approach. However, it can be adapted to asymmetric pairings (cf. [PS16]) such that the encryption scheme no longer needs to work on elements in $\mathbb{G}_T$, and the scheme must also be lifted to the BSZ model. Irrespective of the used paring setting the membership certificate to be encrypted is not a signature (as in BBS variants), but a commitment to a user's secret is encrypted. As this secret is chosen uniformly at random, our assumption also holds.

Relation to Hwang et al.: The feature of controllable linkability for group signature schemes proposed by Hwang et al. [HLC$^+$11, HLC$^+$13] inspired us to our general transformation, but their construction differs in the following aspects. They use two ciphertexts (sharing the same randomness), one for opening and one for linking, and consequently they require an additional ciphertext in the group signature. Recall, that in the BBS scheme a membership certificate is of the form $A_i = g_1^{\frac{1}{x_i+\mathsf{mik}}}$ for x_i randomly chosen by the issuer and when requiring non-frameability (strong exculpability) $A = (g_1 h^{-z_i})^{\frac{1}{x_i+\mathsf{mik}}}$ where z_i is randomly chosen by the user and given in the form of a commitment h^{-z_i} to the issuer. The construction of Hwang et al. [HLC$^+$11] requires an additional issuer-chosen element y_i and their membership certificates are of the form $A = (g_1 h^{-z_i} h'^{-y_i})^{\frac{1}{x_i+\mathsf{mik}}}$. In order to realize controllable linkability, a user encrypts g^{y_i} and provides an additional non-interactive zero-knowledge proof of knowledge that the unrevealed value y_i in the second ciphertext is included in the membership certificate and linking basically represents a plaintext equality check based on ElGamal encryption. The construction in [HLC$^+$13] works analogously but uses another encryption scheme. Consequently, they apply their

---

[2] The latter case rules out trivial cases where the membership certificate that is encrypted is a well known public key. In this case the user could always commit to his secret using a base different from that in the public key and prove the equality of the respective discrete logarithms during joining.

equality test to allow for controllable linkability not directly on the membership certificate but on the second ciphertext that encrypts g^{y_i} and, thus, they do not require Assumption 1 as they satisfy this by construction. The use of an additional ciphertext as well as an additional non-interactive proof of knowledge, however, makes the group signature more expensive from a computational as well as a bandwidth perspective. Hence, similar as in case of some IND-CCA2 secure encryption schemes the GSSs get more complex, although the schemes by Hwang et al. still only support weak anonymity, *i.e.*, (CPA-full-anonymity), whereas IND-CCA2 secure encryption schemes support full anonymity, *i.e.*, (CCA-full-anonymity). Thus, although their construction may also be converted into a generic transformation, it seems more natural to us to use a single ciphertext (which is already available in group signatures) as in our proposed generic transformation and also base the linking decision on the encryption of the membership certificate.

3.4.1 Security Analysis

A nice feature of our generic transformation is that the linking key mlk is independent from the issuing key mik that is required to issue membership certificates. Hence, the linking key does not constitute any danger in the sense that the LA may impersonate the issuer. Since mlk is only related to mok, *i.e.*, the public key encryption scheme used to encrypt membership certificates, we only need to ensure that LAs are not able to trace users, although LAs are able to check whether two signatures have been issued by the same unknown group member. Consequently, all proofs of the security properties of PB-GSSs not related to the property of linkability remain valid as well as untouched by our generic transformation. Thus, we only need to investigate the security properties related to linkability that have been introduced by Hwang et al. [HLC$^+$11, HLC$^+$13].

Security Notions for Controllable Linkability. In GSSs with controllable linkability [HLC$^+$11, HLC$^+$13], the related security issues are covered by the *linkability* property that consists of three separate security notions. For the subsequent discussion we define the following sets and oracles. **CU**, **HU**, and **GSet** represent the sets of corrupted users, honest users, and the set of message-signature pairs generated via queries to the challenge oracle, respectively. Furthermore, **reg** denotes the list of transcripts generated by the Join process. Besides, the following oracles are used (cf. [BSZ05]).

`AddU(i)`**:** The add user oracle allows to add a user i to the group of honest users (**HU**). Therefore, it executes UkGen, Join, and Issue. In the end, user i is in possession of a key pair ($\mathsf{usk_i}$, $\mathsf{upk_i}$), and a signing key $\mathsf{gsk_i}$. The registration information is recorded in **reg**, and $\mathsf{upk_i}$ is published.

`CrptU(i, upk)`**:** The corrupt user oracle allows to set a user's identifiable information, *i.e.*, the public key corresponding to the user's private key, and the user is added to **CU**.

`SndToU(i, M)`: The send to user oracle (by a corrupted issuer) allows an adversary to interact with an honest user i through the (not necessarily honest) execution of the Issue algorithm.

`USK(i)`: The user secret key oracle allows to retrieve the private keys ($\mathsf{usk_i}$, $\mathsf{gsk_i}$) of a user i and thereby turns this user into a corrupted user by adding her to **CU**.

`RReg(i)`: The read registration table oracle allows to read a registration table entry for user i.

`WReg(i, M)`: The write registration table oracle allows to overwrite a registration table entry for user i with M.

`GSig(i, M)`: The group signing oracle allows to query a group signature for a given (honest) user i and a given message M.

`Open(M, σ)`: The open oracle allows to retrieve the signer of a given message-signature pair (M, σ), unless the given signature has been generated via the challenge oracle.

$\mathtt{Ch_b(i_0, i_1, M)}$: The challenge oracle receives two honest users (i_0, i_1) and a message M, and returns a group signature for one of the two users, who is randomly chosen by bit b. The challenge signature is recorded in **GSet**.

In the following we present the ideas as well as the formal properties required for controllable linkability, namely link-only linkability (LO-linkability), judge-proof unforgeability (JP-unforgeability), and enforced linkability (e-linkability) as defined in [HLC$^+$11, HLC$^+$13].

LO-linkability: LO-linkability captures that a linking key should be used only for linking signatures, not for gaining useful information for opening.

JP-unforgeability: JP-unforgeability captures that a linking key cannot be used for generating a judge proof.

E-linkability: E-linkability captures that colluding users should not be able to generate two message-signature pairs satisfying any of the following two conditions (even with the help of the linking authority or the opening authority): (1) Open yields identical identities which are successfully judged, while Link outputs `false` or (2) identities output by Open are different and both are successfully judged, while Link outputs `true`.

Definition 17 (LO-Linkability). *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to provide link-only linkability (LO-linkability) if for any adversary $\mathcal{A}$ and any $\kappa \in \mathbb{N}$, $\Pr[\mathsf{Exp}_{\mathcal{GS},\mathcal{A}}^{\mathsf{LO-link}}(\kappa) = 1] \leq \epsilon(\kappa)$, where the experiment is defined in Figure 3.1.*

```
**Experiment** $\mathsf{Exp}^{\mathsf{LO-link}}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
  $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(1^\kappa)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
  $(i_0, i_1, M) \leftarrow \mathcal{A}^{\mathtt{SndToU},\mathtt{AddU},\mathtt{GSig},\mathtt{Open},\mathtt{USK},\mathtt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk})$;
  $b \xleftarrow{R} \{0,1\}$, $\sigma \leftarrow \mathsf{Ch_b}(i_0, i_1, M)$;
  $b' \leftarrow \mathcal{A}^{\mathtt{SndToU},\mathtt{AddU},\mathtt{GSig},\mathtt{Open},\mathtt{USK},\mathtt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, i_0, i_1, M, \sigma)$;
  If all of the following conditions hold, then return 1 and 0 otherwise
    $i_0, i_1 \in \mathbf{HU}$;
    $\mathtt{GSig}(i_0, \cdot)$, $\mathtt{GSig}(i_1, \cdot)$, and $\mathtt{Open}(M, \sigma)$ have not been queried;
    $b' = b$
```

Figure 3.1: Experiment for LO-linkability.

Definition 18 (JP-unforgeability)**.** *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to provide judge-proof unforgeability (JP-unforgeability) if for any adversary $\mathcal{A}$ and any $\kappa \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{JP-UF}}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1] \leq \epsilon(\kappa)$, where the experiment is defined in Figure 3.2.*

```
**Experiment** $\mathsf{Exp}^{\mathsf{JP-UF}}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
  $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(1^\kappa)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
  $(i, M) \leftarrow \mathcal{A}^{\mathtt{SndToU},\mathtt{AddU},\mathtt{WReg},\mathtt{GSig},\mathtt{Open},\mathtt{USK},\mathtt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk})$;
  $\sigma \leftarrow \mathsf{GSig}(\mathsf{gpk}, i, M)$;
  $\tau \leftarrow \mathcal{A}^{\mathtt{SndToU},\mathtt{WReg},\mathtt{GSig},\mathtt{Open},\mathtt{USK},\mathtt{CrptU}}(\mathsf{gpk}, \mathsf{mik}, \mathsf{mlk}, i, M, \sigma)$;
  If all of the following conditions hold, then return 1 and 0 otherwise
    $i \in \mathbf{HU}$ and $\mathbf{gsk}[i] \neq \emptyset$;
    $\mathtt{Open}(M, \sigma)$ has not been queried;
    $\mathsf{Judge}(\mathsf{gpk}, \mathbf{upk}[i], M, \sigma, \tau) = \mathtt{true}$
```

Figure 3.2: Experiment for JP-unforgeability.

Definition 19 (E-Linkability)**.** *A group signature scheme $\mathcal{GS}$ with controllable linkability is said to provide enforced linkability (E-linkability) if for any adversary $\mathcal{A}$ and any $\kappa \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{E-link}}_{\mathcal{GS},\mathcal{A}}(\kappa) = 1] \leq \epsilon(\kappa)$, where the experiment is defined in Figure 3.3.*

The definitions provided above have been slightly adapted. More specifically, we corrected some minor flaws in the definitions provided by Hwang et al. [HLC⁺11, HLC⁺13]. In particular, in the LO-unlinkability experiment the adversary $\mathcal{A}$ is additionally required to have access to the $\mathtt{AddU}(\cdot)$ oracle as otherwise there cannot be any honest user in the experiment and the winning condition can never be satisfied. Furthermore, in the JP-unforgeability experiment, we find it more natural to write that $\mathcal{A}$ is not allowed to query $\mathtt{GSig}(i_0, \cdot)$ and $\mathtt{GSig}(i_1, \cdot)$ in the second phase, which seems to be implicitly equivalent to their condition of not being allowed to query $\mathtt{GSig}(i_b, \cdot)$. Moreover, in their E-linkability experiment $\mathsf{upk}[i]$ should be $\mathsf{upk}[i_0]$ and $\mathsf{upk}[j]$ should be $\mathsf{upk}[i_1]$.

```
**Experiment** $\mathsf{Exp}^{\mathsf{E-link}}_{\mathcal{GS},\mathcal{A}}(\kappa)$:
  $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}) \leftarrow \mathsf{GkGen}(1^\kappa)$, $\mathbf{CU} \leftarrow \emptyset$, $\mathbf{HU} \leftarrow \emptyset$, $\mathbf{GSet} \leftarrow \emptyset$;
  $(M_0, \sigma_0, M_1, \sigma_1) \leftarrow \mathcal{A}^{\mathtt{SndToU},\mathtt{AddU},\mathtt{RReg},\mathtt{GSig},\mathtt{USK},\mathtt{CrptU}}(\mathsf{gpk}, \mathsf{mok}, \mathsf{mlk})$;
  If $\mathsf{GVf}(\mathsf{gpk}, M_0, \sigma_0) = \mathtt{false}$ or $\mathsf{GVf}(\mathsf{gpk}, M_1, \sigma_1) = \mathtt{false}$ then
    return 0;
  $(i_0, \tau_{i_0}) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M_0, \sigma_0, \mathsf{mok})$;
  $(i_1, \tau_{i_1}) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M_1, \sigma_1, \mathsf{mok})$;
  If $\mathsf{Judge}(\mathsf{gpk}, M_0, \sigma_0, i_0, \mathbf{upk}[i_0], \tau_{i_0}) = \mathtt{false}$ or
    $\mathsf{Judge}(\mathsf{gpk}, M_1, \sigma_1, i_0, \mathbf{upk}[i_1], \tau_{i_1}) = \mathtt{false}$ then
      return 0;
  If $i_0 \neq i_1$ and $\mathsf{Link}(\mathsf{gpk}, M_0, \sigma_0, M_1, \sigma_1, \mathsf{mlk}) = \mathtt{true}$ then
    return 1;
  else if $i_0 = i_1$ and $\mathsf{Link}(\mathsf{gpk}, M_0, \sigma_0, M_1, \sigma_1, \mathsf{mlk}) = \mathtt{false}$ then
    return 1;
  else return 0;
```

Figure 3.3: Experiment for E-linkability.

Our construction does not change the security arguments from [HLC$^+$11, HLC$^+$13]. The argumentation in the proofs is nearly identical whereas we require a more abstract level of argumentation for our generic transformation. Subsequently, we assume that AoN-PKEET* is based on the encryption scheme used in the respective PB-GSS.

Lemma 4. *If AoN-PKEET* is secure, PB-GSS is secure and Assumption 1 holds, the PB-GSS with controllable linkability obtained from the PB-GSS provides LO-linkability.*

PB-GSSs following the SEP paradigm that are secure in the BSZ model require an IND-CCA2 secure encryption scheme to simulate the Open oracle. In case of our AoN-PKEET* scheme that allows for efficient zero-knowledge proofs of knowledge of encrypted messages, we assume an IND-CCA2 secure encryption scheme resulting from the application of the twin encryption paradigm [NY90, RS91, FP01] on an IND-CPA secure encryption scheme. This assumption clearly holds for all existing GSSs following the SEP paradigm that are secure in the BSZ model as these schemes employ the twin encryption paradigm. Recall that the twin encryption paradigm uses two independent public keys of the same encryption scheme to encrypt a message under both keys, and uses a simulation sound non-interactive zero-knowledge proof of knowledge system (cf. [RS91]) Π to prove plaintext knowledge and equality. Now we prove LO-Linkability under the IND-CPA/IND-CCA2 security of the used AoN-PKEET* scheme. The idea is to obtain one public key from a challenger of an AoN-PKEET* instance, and to generate another independent public key which will be used to generate the linking trapdoor during the experiment. To properly answer oracle queries, we use the simulator of Π to produce false proofs which allow us to embed the challenge ciphertext into the challenge signature.

Proof (Lemma 4). An adversary $\mathcal{A}$ that is able to determine whether a challenge signature has been issued by i_0 or i_1 has done so without calling Open for the challenge signature, not knowing gsk_0 and gsk_1 as well as not having issued any signature query for i_0 or i_1 to the `GSig` oracle, can be turned into an adversary against the ciphertext indistinguishability of the underlying public key encryption scheme. We set up the environment for $\mathcal{A}$ by obtaining a public key pk_0 from an AoN-PKEET* challenger. In case of an $\mathsf{IND\text{-}CCA2}$ secure twin encryption scheme we simply take the first key. We choose sk_1 in the simulation, compute $\mathsf{mlk} \leftarrow \mathsf{Aut}(\mathsf{sk}_1)$, and give mlk to the adversary $\mathcal{A}$. During the first phase we can simulate all oracle queries for $\mathcal{A}$ as in the real game. Eventually $\mathcal{A}$ outputs (i_0, i_1, M) and the environment queries the AoN-PKEET* challenger with the two certificates corresponding to i_0 and i_1 in order to obtain a challenge ciphertext. Under the $\mathsf{IND\text{-}CPA}$ security (real-or-random) of the encryption scheme, the simulator encrypts some random message with the second public key pk_1. Note that the adversary $\mathcal{A}$ does not have access to mlk in the second phase anymore. The environment uses the simulator of Π to produce a simulated proof and generates the final challenge signature σ. If $\mathcal{A}$ is able to win the experiment, $\mathcal{A}$ can be turned into an adversary against the ciphertext indistinguishability of the AoN-PKEET* scheme, which contradicts the assumption that AoN-PKEET* is secure. ☐

Lemma 5. *If AoN-PKEET* is secure and PB-GSS is secure, the PB-GSS with controllable linkability obtained from the PB-GSS provides JP-Unforgeability.*

Proof (Lemma 5). In PB-GSSs following the SEP paradigm, the proof τ output by the Open algorithm represents a non-interactive zero-knowledge proof of knowledge of equality of the returned certificate and the certificate within the ciphertext (which is usually a simple proof of equality of discrete logarithms). An efficient adversary $\mathcal{A}$ that wins the JP-unforgeability experiment can be used to extract the private key corresponding to the public key encryption key in gpk and mlk. The reduction can embed the problem instance into the public key in gpk and mlk and simulating the proofs required by the Open queries (by programming the random oracle). The private key can then be extracted from $\mathcal{A}$ by using the extractor (e.g., standard rewinding techniques when in the random oracle model) for the proof of knowledge in the reduction. ☐

Lemma 6. *If AoN-PKEET* is secure and PB-GSS is secure, the PB-GSS with controllable linkability obtained from the PB-GSS provides E-linkability.*

Proof (Lemma 6). If an adversary $\mathcal{A}$ outputs two pairs of messages and signatures (M_0, σ_0) and (M_1, σ_1) such that $(i_b, \tau_b) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathbf{reg}, M_b, \sigma_b, \mathsf{mok})$ and $\mathtt{true} \leftarrow \mathsf{Judge}(\mathsf{gpk}, M_b, \sigma_b, i_b, \mathsf{upk}_{i_b}, \tau_b)$ for $b = 0, 1$, then we have two cases:

Case 1: In the first case we have that $i_0 = i_1$ but the two signatures do not link, *i.e.*, $\mathtt{false} \leftarrow \mathsf{Link}(\mathsf{gpk}, \mathsf{M}_0, \sigma_0, \mathsf{M}_1, \sigma_1, \mathsf{mlk})$. This means that for $\sigma_0 = (T_0, \pi_0)$ and $\sigma_1 = (T_1, \pi_1)$ both ciphertexts T_0 and T_1 are encryptions of the membership certificate of the same user ($i_0 = i_1$ follows from Open), but $\texttt{false} \leftarrow \mathsf{Com}(T_0, T_1, \mathsf{mlk})$. This, however, contradicts the soundness of the AoN-PKEET* scheme.

Case 2: In the second case we have that $i_0 \neq i_1$ but the two signatures link, *i.e.*, $\texttt{true} \leftarrow \mathsf{Link}(\mathsf{gpk}, \mathsf{M}_0, \sigma_0, \mathsf{M}_1, \sigma_1, \mathsf{mlk})$. This means that for $\sigma_0 = (T_0, \pi_0)$ and $\sigma_1 = (T_1, \pi_1)$ the ciphertexts T_0 and T_1 are encryptions of distinct certificates of the users $i_0 \neq i_1$. However, as $\mathsf{true} \leftarrow \mathsf{Com}(T_0, T_1, \mathsf{mlk})$, this, again, contradicts the soundness of the AoN-PKEET* scheme.

□

Since all other properties remain unaffected by the feature of controllable linkability, we obtain the following theorem from the above lemmas.

Theorem 1. *If AoN-PKEET* is secure, the PB-GSS is secure, and Assumption 1 holds, then the generic transformation yields a secure PB-GSS with controllable linkability.*

3.5 Comparison with Other Approaches

In this section, we discuss enhanced anonymity management mechanisms for GSs with respect to their capability of realizing controllable linkability.

3.5.1 Linkable Group Signatures

Tracing-by-linking (TbL) group signatures are group signatures where, in contrast to the SEP paradigm (tracing-by-escrowing), the signer's anonymity cannot be revoked by any combination of authorities. Only if a group member signs twice (or more general $k \geq 2$ times), then her identity can be traced publicly without any trapdoor (cf. [TFS04]). Another direction are link-but-not-trace GSs [NFW99], where signatures contain a tag such that double signing can be detected but without the help of each member (disavowing stage) the signer cannot be identified. In the disavowing stage each user needs to prove that a specific signature has not been produced by her, where the actual signer cannot produce a valid proof. These approaches cannot be compared to controllable linkability, since in these approaches linking is a public operation to detect double signing by the same entity (or more general k-time signing) while controllable linkability requires that only a dedicated party can perform the linking operation.

3.5.2 Traceable Signatures

Traceable signatures [KTY04, CPY06, LY09] are GSs extending the opening feature by (1) *user tracing*, meaning that the group manager can publish a tracing trapdoor which allows to trace all signatures of the respective user, and (2) *signature claiming*, meaning that the signer of a signature can provably claim that the signature has been produced by her. Hence, traceable signatures allow selective traceability, where an authority can compute a tracing trapdoor for every user such that only signatures produced by this user can be linked using this tracing trapdoor, but signatures from other users remain unlinkable. Consequently, this functionality could be used to implement controllable linkability by giving a tracing trapdoor for every user to a linking authority. Given two signatures, the linking authority can take every user trapdoor and check this relation for both signatures. If the relation is satisfied for the same trapdoor, the signatures are from the same anonymous group member and from different anonymous group members otherwise. Obviously, linking requires computational costs linear in the number of group members, which can soon become impractical for larger groups and additionally the group manager needs to communicate tracing trapdoors of newly joined users frequently (and in time) to the linking authority.

Chow [Cho09] employs the idea of "join once, spend many" from an e-cash scheme to propose real traceable signatures. He argues that all previous traceable signatures are not optimal since checking whether a signature has been issued by the user corresponding to a given tracing trapdoor requires additional computations. The idea behind his alternative construction is that a tracing trapdoor allows to deterministically recompute every tag of a user's signature (the tracing trapdoor is the seed for a verifiable random function [DY05]) and thus to trace. Consequently, the tracing authority can compute a list of tags and send them to every verifier who can then check if the tag of the signature can be found in the given list. However, the drawback of this approach is that only ℓ unlinkable signatures can be issued by every user and every signature requires an additional zero-knowledge range proof, *i.e.*, that the used value for the tag is less than ℓ, where ℓ is a fixed parameter in the system. Consequently, signatures get more expensive the larger ℓ is and so does the size of the list of tags per user. When used to realize controllable linkability, lists of tags for every member of the group may be precomputed and given to the linking authority and the linking of two signatures would then require a check whether the tags of the two signatures in question can be found on the same list. For large groups and large ℓ the size of the collection of lists for all users can be quite significant and, as in case of traceable signatures, for every joining user such a list for the new user has to be communicated to the linking authority. Moreover, we consider the application of this approach not suitable for controllable linkability, as the respective group signature scheme only supports ℓ unlinkable group signatures and consequently does not constitute a standard GSS.

3.5.3 Public Key Anonymous Tag Systems

Abe et al. [ACHO11] further generalized the approach of Chow [Cho09] to a primitive denoted as *public key anonymous tag system* which can be used to construct traceable signatures. Essentially, Abe et al. use so-called *link tokens*, where an authority with a dedicated link key can compute these link tokens for specific users that should be traceable. Given such a token for a specific member, signatures produced by this member can be publicly traced by anyone. Furthermore, their scheme allows all members to compute their link tokens without knowledge of the link key. Thus, signers can always link their own signatures which might not be desirable in some scenarios. An additional zero-knowledge proof based on this link token and the user's private key allows signers to claim and non-signers to deny being the signer of a message. While this property might be useful in some applications, it also leads to the unavoidable attack that $N-1$ colluding members can reveal the signer of a message by proving that they did not sign a specific message. Again, controllable linkability can be implemented with public key anonymous tag systems, but the linking requires costs linear in the size of the number of group members and cannot be precomputed (as it is the case with traceable signatures). In addition, the link tokens required to perform this linking must be distributed to the linking authority as soon as new members join the group.

3.5.4 Verifier Local Revocation

Verifier local revocation (VLR) for group signatures has been proposed by Boneh and Shacham [BS04]. In such schemes verifiers are given a revocation list in order to determine whether a signature stems from an already revoked member. More precisely, the verifier needs to check signatures against all entries on this revocation list, which means that the computational effort for the verifier grows linearly in the number of revoked users. Similar to how traceable signatures could be used for controllable linkability, the group manager could give a list with tokens for *all* group members to the linking authority. For two given signatures, the linking authority can test each of the signatures against the list and if the match happens on the same position in the two runs then the signature has been produced by the same anonymous signer. Consequently, this approach also requires costs linear in the number users. Furthermore, as it is also the case for traceable signatures, it requires to deliver a user token to the linking authority after every joining operation of a user.

3.5.5 Comparison

Table 3.1 provides a comparison of the above outlined concepts. Although there exist concepts that sound quite similar, *i.e.*, tracing-by-linking as well as link-but-not-trace GSs, the intention of these mechanisms is different, namely to prevent signers from signing more than a predefined number of messages. Thus, these two concepts cannot be employed for the implementation of controllable linkability. The other concepts can be employed to achieve controllable linkability, but are rather inefficient in terms of communication overhead (communication of trapdoors to the linking authority) every time a new user joins the group, in terms of additional costs for the actual linking of signatures as well as in terms of memory requirements to achieve the desired functionality of linking.

To conclude, when requiring controllable linkability in GSSs, the proposed generic construction to convert PB-GSSs based on the SEP paradigm into GSSs with controllable linkability is superior to "emulating" controllable linkability by means of features provided by other constructions. As the linking authority holds a single trapdoor in the form of a linking key, there is no necessity to

Table 3.1: Applicability of related concepts to achieve controllable linkability (CL). N denotes the number of group members. l denotes the number of signatures that can be issued by a signer before signatures become publicly linkable.

Mechanism	**Suitable for CL**	**Overhead for LA**		
		Update on Join	**Link**	**Memory**
Tracing-by-linking group signatures [TFS04]	✘	-	-	-
Link-but-not-trace group signatures [NFW99]	✘	-	-	-
Traceable signatures [KTY04]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Real traceable signatures [Cho09]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(\ell \cdot N)$
Public key anonymous tag systems [ACHO11]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Verifier local revocation [BS04]	✓	✓	$\mathcal{O}(N)$	$\mathcal{O}(N)$
Controllable linkability	✓	✘	$\mathcal{O}(1)$	$\mathcal{O}(1)$

communicate user specific trapdoors from the group manager to the linking authority on every joining of a new user. Thus, compared to other approaches that have a different goal but still may be used to achieve controllable linkability, our generic approach is more efficient and simple when only requiring controllable linkability and no other traceability feature.

4

Linking-Based Revocation

Quickness is the essence of the war.
— Sun Tzu, The Art of War

The applicability of GSs is still restricted due to inefficiencies of existing membership revocation mechanisms that place a computational burden and communication overhead on signers and verifiers. In particular, it seems that the general belief (or unwritten law) of avoiding online authorities by all means artificially and unnecessarily restricts the efficiency and practicality of revocation mechanisms in GSSs. While a mindset of preventing online authorities might have been appropriate more than 10 years ago, today the availability of highly reliable cloud infrastructures could be employed to solve open challenges.

In order to overcome the inefficiencies of existing revocation mechanisms, we propose an alternative approach denoted as linking-based revocation (LBR). The novelty of LBR is its transparency for signers and verifiers that spares additional computations as well as updates. We introduce dedicated revocation authorities (RAs) that can be contacted for efficient (constant time) revocation checks. In order to protect these RAs and to reduce the trust in these authorities, we also introduce distributed controllable linkability such that RAs need to cooperate with multiple authorities to compute the required linking/revocation tokens. Besides efficiency, an appealing benefit of LBR is its generic applicability to PB-GSSs secure in the BSZ model and GSSs with controllable linkability.

We provide a motivation for LBR in Section 4.1 and discuss state-of-the-art revocation mechanisms in Section 4.2. We recall the basic building blocks in Section 4.3 and introduce LBR in Section 4.4. We demonstrate the ease of applicability in Section 4.5. Parts of this chapter are taken verbatim from [SSU16].

Publication Data and Contribution

Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group Signatures with Linking-Based Revocation: A Pragmatic Approach for Efficient Revocation Checks. In *Conference on Cryptology & Malicious Security – Mycrypt 2016*, 2016. In press
Contribution: Main author; Initial idea together with Thomas Unterluggauer; Idea for token indistinguishability developed by Daniel Slamanig.

4.1 Introduction

Membership revocation in GSSs is a non-trivial task as (1) users are anonymous and their privacy should be protected even in case of revocation, and (2) the revocation of one user should not affect the signing capabilities of other users. Although membership revocation has gained increasing attention in the last decade [BS04, NF05, FHM11, AEHS14, EMO14, KLP$^+$15], existing revocation mechanisms place a computational burden and communication overhead on signers and verifiers. While existing concepts seem to prevent online authorities by all means, we show that a transition towards online authorities—which we will protect by means of threshold cryptography—allows for the most efficient (constant time), and most generic revocation mechanism for existing GSSs. Given a signature in question and a revocation list, a revocation authority determines the revocation status of a signer by using a dedicated trapdoor to (anonymously) link the signature against a list of revoked members. Hence, this authority still preserves the signers' anonymity. Our somewhat unconventional proposal of using an online revocation authority overcomes many issues of existing revocation mechanisms and, thus, we believe that our revocation mechanism represents a valuable addendum to the portfolio of revocation mechanisms.

Online Requirement. Although our approach relies on an online authority for revocation checks, we argue that today many devices are already connected to the Internet permanently and rely on the availability of cloud computing infrastructures. Especially the establishment of the Internet of Things (IoT) requires devices being connected to the Internet, either via WiFi or even embedded SIM cards, for various (sometimes dubious) reasons. Nevertheless, irrespective of whether existing IoT devices provide any useful features, the point is that many devices are already interconnected among each other and also extensively use Internet services based on cloud computing infrastructures and, thus, we consider an online RA as absolutely reasonable. In cases where network connectivity and availability are not an issue, our proposed revocation mechanism provides dedicated advantages, e.g., immediate revocation without the need to distribute revocation information to signers or verifiers. Further, since signature verification is decoupled from the online revocation checks, these revocation checks can also be postponed in case the revocation authority might not be available. Besides, as we will discuss later, we protect the trapdoor key by means of threshold cryptography in order to reduce the risk of attacks.

4.2 State-of-the-Art in Revocation

Below we discuss efficiency considerations of existing revocation mechanisms using the metric of additional computations and updates required for signers as well as verifiers. Subsequently, R denotes the number of revoked members and N the number of group members.

Basic Approaches. The most basic approach is *reissuance-based revocation* [AST02] that requires all non-revoked members to receive new credentials. Similarly, *credential-update revocation* (CUR) [BBS04] requires non-revoked members to update their credentials on every revocation. Both mechanisms suffer from additional communication and computation overhead in case of frequent revocations. In particular, $\mathcal{O}(R)$ (multi-)exponentiations for signers.

Blacklist Revocation. *Certificate-based blacklist revocation* (BR-C) [BS01] requires signers to provide a zero-knowledge proof that they are not listed on a revocation list (RL), which means that signers/verifiers need to perform $\mathcal{O}(R)$ computations for every sign/verify operation and the signature size also increases linearly in R. *Accumulator-based blacklist revocation* (BR-A) [CL02, ATSM09, FHM11] applies (universal) cryptographic accumulators to allow for a compact and constant-size representation of RL as well as constant-size proofs to prove (non-)membership. *Blacklists of ordered credential-identifier pairs* (BR-ID) [NFHF09] also lead to constant costs for signers and verifiers, but signers need to fetch an updated RL in the size of $\mathcal{O}(R)$ (and $\mathcal{O}(N)$ in predecessor schemes [NS04a, NKHF05]) on each revocation, and the public key size is $\mathcal{O}(N)$ (or $\mathcal{O}(\sqrt{N})$ with higher signing costs).

Verifier-Local Revocation. In *verifier-local revocation* (VLR) [BS04, NF05, NF06, ZL06], verifiers test for a given signature whether a certain relation holds for each entry on RL, which indicates that the signer has been revoked. Consequently, verifiers need to update RL on every revocation and a check costs $\mathcal{O}(R)$ group operations or even pairing evaluations. Although Boneh and Shacham [BS04] propose a constant-time revocation check, it only works if the same message and the same randomness is used for all signatures. This, however, is only reasonable for specific applications like the DAA setting described in [BS04]. Other disadvantages of VLR are that signatures of revoked members become linkable for all verifiers, *i.e.*, it lacks backward unlinkability, and that anyone in possession of RL can link signatures of revoked users. Although this can be fixed and backward unlinkability can be added, e.g., by introducing time intervals [NF05], this still adds additional non-trivial overhead.

Besides, Emura and Hayashi [EH15] proposed time-token dependent linking which can also be applied for VLR. However, signatures become publicly linkable (without any trapdoor information) if users sign more than once per time period, a separate RL must be maintained for each time period, and RLs need to be recomputed entirely for each time period since the revocation tokens of users change in each time period. While one could encrypt these revocation tokens to be decryptable by revocation authorities only, *i.e.*, to prevent public linkability, this would increase the signature size in part due to additional zero-knowledge proofs. In contrast, our approach preserves the privacy of signers and does not increase the signature size as it relies on the information already available in standard GSSs. Chow et al. [CSY06] proposed a similar concept for membership revocation in ID-based ring signatures, a related but different concept.

Group signatures with probabilistic revocation (GSPR) [KLP$^+$15] allow for constant-time revocation checks at the expense of probabilistic revocation guarantees. However, in contrast to VLR the signer has to perform $\mathcal{O}(m)$ expensive operations, where m is a fixed value representing the number of signatures that can be issued by a signer before signatures become publicly linkable. Consequently, there is a trade-off between the storage/computational requirements for signers and the requirement for performing the group setup phase again. Moreover, the size of the group public key is $\mathcal{O}(m)$ and the GM needs to process $\mathcal{O}(m)$ user-specific tokens in order to update RL.

Revocation Mechanisms for Standard Model GS. Revocation mechanisms have also been proposed for GSSs with security in the standard model. These mechanisms are designed to be compatible with the Groth-Sahai proof system [GS08], instead of relying on Σ-protocols and the random oracle model. State-of-the-art mechanisms have been proposed by Libert, Peters, and Yung (LPY) [LPY12b], which rely on the ciphertext of a broadcast encryption scheme as a RL. Later, LPY [LPY12a] has been improved to achieve constant size group signing keys. Attrapadung et al. (AEHS) [AEHS14] further reduced the revocation list to a constant size. However, signature sizes for LPY are about 100 and 144 group elements respectively, and AEHS produces even larger signatures. Thus, we exclude these revocation mechanisms from our comparison below, since we put a focus on GSSs that allow signers to be executed in resource-constrained environments as will be discussed in our motivating example later in this section.

Our Proposal (GS-LBR). Existing revocation mechanisms are often too inefficient to be implemented in practical scenarios and especially in resource-constrained environments, which is why revocation has been identified as the major bottleneck of state-of-the-art GSSs [MFG$^+$12]. We address this problem by introducing *linking-based revocation* (LBR). LBR can be generically applied to existing GSSs, and in particular PB-GSSs [NS04b, DP06, HLC$^+$11, HLC$^+$13, Int13, HCCN15] following the SEP paradigm. Essentially, we rely on the feature of controllable linkability as introduced in Chapter 3. The idea of LBR is that, in analogy to the online certificate status protocol (OCSP) [SMA$^+$13] which is used for certificate revocation checks in the PKIX setting[1], an online party can be contacted for the revocation check. To prevent attackers from compromising online RAs, we introduce the feature of *distributed controllable linkability* which

---

[1] OCSP is the most popular approach for revocation checks in the PKIX setting [Pon15].

may be of independent interest. When applying distributed controllable linkability to revocation, it allows RAs to anonymously link a signature—with the cooperation of at least two linking authorities—against anonymous revocation tokens on RL. An additional optimization allows for constant-time revocation checks.

In contrast to existing revocation mechanisms, our mechanism is transparent for signers and verifiers. Most importantly, LBR is efficient in the sense that (1) no key updates or additional computations are required for signers, (2) no expensive local revocation checks are required for verifiers, and (3) neither the signature size nor the key size increases. This allows us to support membership revocation even in applications that require resource-constrained environments for signature generation. While all existing revocation approaches require signers and/or verifiers to fetch (possibly large) RLs from time to time, our mechanism relies on an always-online authority that is available for revocation checks. Although an online authority for such tasks might be considered as being unconventional or impractical at first, we believe that such an always-online requirement for specific authorities is absolutely reasonable and a similar approach has also been discussed in the context of anonymous credential systems (cf. [Ver16]). In order to protect these RAs against attacks, we distribute the linking trapdoor required for the revocation checks to multiple entities. Consequently, an attacker would have to corrupt multiple entities to recover the linking trapdoor.

Comparison. Table 4.1 compares existing revocation mechanisms for practical group signature schemes secure in the random oracle model regarding their efficiency and practicality. For each mechanism, we compare the memory overhead for the group public key and the signature as well as the computational overhead for updating keys/credentials, signature generation, and signature verification. Furthermore, we indicate the amount of information that needs to be fetched by signers and verifiers in case of revocation. As some schemes require both signers and verifiers to fetch updates, we also indicate whether these updates must be synchronized, *i.e.*, whether both parties need to have the same update-version as otherwise valid signatures cannot be computed and verified. Last but not least, we indicate whether signers and verifiers must be online for the revocation mechanism to work, where $\Diamond$ means semi-online, *i.e.*, signers and verifiers can decide when to go online to fetch the necessary updates.

Although VLR seems to provide similar advantages and features as LBR, our approach of LBR only allows RAs to link signatures, which is not the case within VLR where users can link signatures themselves. Thus, LBR overcomes the delicate issue of revoked members losing their anonymity. In addition, our proposed revocation mechanism can be generically applied to many PB-GSSs following the SEP paradigm, which covers a large class of state-of-the-art and practically efficient GSSs. As we will see below, our approach of LBR provides dedicated advantages and superior features for specific scenarios.

Table 4.1: Comparison of revocation mechanisms. N denotes the number of group members, R denotes the number of revoked members, and m denotes the number of signatures that can be issued by a signer before signatures become publicly linkable. ◊ means semi-online, *i.e.*, signers and verifiers can decide when to go online to fetch the necessary updates.

Type	**Overhead memory**		**Overhead time**			**Updates**		**Synchronized**	**Online**	
	GPK	Signature	Update (signer)	Sign	Verify	Signers	Verifiers		Signers	Verifiers
CUR [BBS04, HLC+11]	–	–	$\mathcal{O}(R)$	–	–	$\mathcal{O}(R)$	$\mathcal{O}(1)$	✓	◊	◊
BR-C [BS01]	–	$\mathcal{O}(R)$	–	$\mathcal{O}(R)$	$\mathcal{O}(R)$	$\mathcal{O}(R)$	$\mathcal{O}(R)$	✓	◊	◊
BR-ID [NFHF09]	$\mathcal{O}(\sqrt{N})$	$\mathcal{O}(1)$	–	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(R)$	$\mathcal{O}(1)$	✓	◊	◊
BR-A [FHM11]	–	$\mathcal{O}(1)$	–	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	✓	◊	◊
VLR [BS04]	–	–	–	–	$\mathcal{O}(R)$	–	$\mathcal{O}(R)$	–	✘	◊
GSPR [KLP+15]	$\mathcal{O}(m)$	$\mathcal{O}(1)$	–	$O(m)$	$\mathcal{O}(1)$	–	$\mathcal{O}(1)$	–	✘	◊
LBR	–	–	–	–	$\mathcal{O}(1)$	–	–	–	✘	✓

Motivating Example. As pairing-based cryptography has been optimized for resource-constrained devices [CCdMP10, CDDT12, GAL$^+$12, PWH$^+$13, UW14, IiRPP15], PB-GSSs have become entirely practical, at least when considering the performance of signature generation only. For example, GSSs have been proposed as a privacy-preserving mechanism for public transport systems [IVP$^+$13]. Their application allows passengers to anonymously prove possession of a valid ticket, but the service provider cannot identify passengers. Still, revocation of misbehaving passengers by invalidating tickets must be possible and these revocations should not affect other tickets in any way. Clearly, frequent updates through authenticated channels between tickets and service providers are impractical, as they would affect the valid tickets. Besides, performance is a crucial issue and, hence, the invalidation of one ticket should not lead to additional computations for the remaining (valid) tickets.

While VLR might be a possible solution to overcome these problems, public transport systems usually support tickets with a limited validity, *i.e.*, 1-hour tickets, daily tickets, monthly tickets, and yearly tickets. Such tickets must be immediately revoked as soon as their validity ends and, thus, immediate revocation of tickets must be efficiently possible. Hence, VLR still faces the following problems. (1) RLs lead to $\mathcal{O}(R)$ computational effort for verifiers which is especially daunting in case of large RLs, and (2) RLs change frequently and must be distributed in a timely manner, *i.e.*, immediately after the revocation of one ticket, to many verifiers. Clearly, LBR overcomes these issues as it gets rid of the computational overhead for signers as well as verifiers and the need to communicate any revocation updates to signers and verifiers. Applying LBR allows the service provider to implement an existing PB-GSS (e.g., [NS04b, DP06, HLC$^+$11, HLC$^+$13, Int13, HCCN15]) as a means to prove possession of a valid ticket. Turnstiles and gates that check the validity of a ticket are connected to the revocation authority. For each ticket to be verified, turnstiles request the revocation check via specifically deployed RAs and depending on the returned decision, access is either granted or denied.

Considering some of the biggest metro systems around the world with several hundred millions of served passengers per year, e.g., Beijing, Moscow, and NY City, the efficiency of the used revocation mechanism is of utmost importance. Besides, also the European Union demands for privacy protection of individuals and the principle of data minimization in transportation systems within the EU Directive 2010/40. Hence, GSSs will likely play an important role in the future and efficient revocation mechanisms will be required.

4.3 Building Blocks for GSs with LBR

Subsequently, we briefly outline the high-level idea of our proposed revocation mechanism. Afterwards, we introduce the necessary building blocks and modifications, *i.e.*, we show how to achieve constant-time revocation checks and we also introduce the feature of distributed controllable linkability.

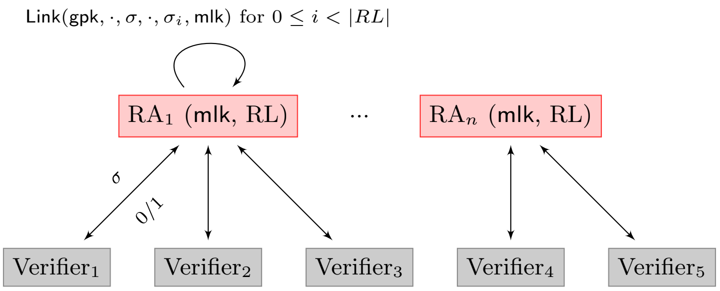

Figure 4.1: Naive instantiation of linking-based revocation.

4.3.1 High-Level Idea of GSs with LBR

Recall that our generic compiler introduced in Chapter 3 allows to add controllable linkability to PB-GSS following the SEP paradigm that are secure in the BSZ model. The dedicated approaches to construct group signatures with controllable linkability in [HLC$^+$11, HLC$^+$13, HCCN15] implicitly use the same idea and thus can also be used in combination with our revocation approach.

Based on the concept of controllable linkability, the idea of linking-based revocation is as follows. A dedicated RA is given the master linking key, a revocation list, e.g., a list of signatures of revoked members, and a signature for which the revocation status should be determined. For the revocation check, the RA links the given signature against all entries on RL. If the given signature can be linked to any of these signatures, the corresponding signer has been revoked, otherwise the signer has not been revoked. Figure 4.1 illustrates this basic approach, which is, however, rather naive for the following reasons.

1. The revocation check is linear in the size of RL, *i.e.*, the check requires $\mathcal{O}(R)$ computations.

2. If an attacker compromises the RA and steals the linking key, she would be able to link any two signatures which is clearly not desired.

Subsequently, we deal with these issues and gradually introduce the necessary modifications to achieve (1) constant-time revocation checks, and (2) to remove the single point of attack by distributing the linking key among multiple entities.

4.3.2 Constant-Time Revocation Checks

To obtain constant-time revocation checks, we modify the $\mathsf{Com}(T, T', \mathsf{tk})$ algorithm of the AoN-PKEET* scheme such that it no longer decides whether two ciphertexts encrypt the same message, but instead returns a value—the *revocation token*—computed from a given ciphertext T and the trapdoor tk. For example, for ElGamal encryption we have a ciphertext $T = (T_1, T_2) = (g^\alpha, mh^\alpha)$ and a trapdoor $\mathsf{tk} = (\hat{r}, \hat{r}^\xi)$, which yields revocation tokens of the form $e(T_2, \hat{r}) \cdot e(T_1, \hat{s})^{-1} = e(m, \hat{r})$. We denote such an invocation as $\mathsf{t} \leftarrow \mathsf{Com}(T, \perp, \mathsf{tk})$.

Security Definition. In order to reason about the security of such a mechanism when applied to revocation, we introduce the notion of token indistinguishability. Token indistinguishability considers an adversary that does not know the trapdoor tk but for a ciphertext $T = (T_1, T_2)$ on any message m observes tokens t of the form $e(T_2, \hat{r}) \cdot e(T_1, \hat{t})^{-1} = e(m, \hat{r})$. This information, however, does not allow the adversary to reason about any tokens seen in the future.

Definition 20 (Token Indistinguishability)**.** *An* AoN-PKEET* *scheme is* $\mathsf{T\text{-}IND}$*, if for all PPT adversaries* $\mathcal{A}$ *and security parameters* κ *there is a negligible function* ϵ *such that:*

$$\mathsf{Pr}\left[\begin{array}{l} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(1^\kappa), \mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{sk}), \\ s \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RMsg}}}(\mathsf{pk}), m \overset{R}{\leftarrow} \mathcal{M}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, m), \\ b \overset{R}{\leftarrow} \{0,1\}, \mathsf{t}_0 \leftarrow \mathsf{Com}(c, \bot, \mathsf{tk}), \mathsf{t}_1 \overset{R}{\leftarrow} \mathcal{T} \\ b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RMsg}}}(\mathsf{pk}, m, \mathsf{t}_b, s) \end{array} \quad : \quad b^* = b \right] \leq 1/2 + \epsilon(\kappa)$$

where $\mathcal{M}$ *represents the message space,* $\mathcal{T}$ *represents the token space, and* $\mathcal{O}_{\mathsf{RMsg}}$ *represents the oracle to generate random messages and corresponding tokens* (m_i, t_i)*, such that* $m_i \overset{R}{\leftarrow} \mathcal{M}$ *and* $\mathsf{t}_i \leftarrow \mathsf{Com}(\mathsf{Enc}(\mathsf{pk}, m_i), \bot, \mathsf{tk})$*.*

Lemma 7. *Under the DDH assumption, AoN-PKEET* based on ElGamal in* $\mathbb{G}_1$ *in an XDH setting is* $\mathsf{T\text{-}IND}$*.*

Proof (Lemma 7). Given an adversary $\mathcal{A}$ that breaks the $\mathsf{T\text{-}IND}$ of AoN-PKEET*, we show how to construct an adversary $\mathcal{B}$ against DDH. Let (g, g^a, g^b, g^c) be a DDH instance given to $\mathcal{B}$. $\mathcal{B}$ randomly generates a private key sk and corresponding public key pk, and sets $\mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{sk})$, *i.e.*, $\mathcal{B}$ implicitly sets $\hat{r} = \hat{g}^b$. $\mathcal{A}$ is now allowed to query $\mathcal{O}_{\mathsf{RMsg}}$, which $\mathcal{B}$ answers as $(g^{m_i}, e((g^b)^{m_i}, \hat{g}))$ for a random $m_i \in \mathbb{Z}_p$. Eventually, $\mathcal{A}$ receives the challenge $(g^a, e(g^c, \hat{g}))$ and outputs its guess. It is clear that if the DDH instance is valid, then the challenge represents a valid message-token tuple and is an independent and random element otherwise. Thus, we perfectly simulate the $\mathsf{T\text{-}IND}$ game for $\mathcal{A}$ and it is clear that whenever $\mathcal{A}$ breaks $\mathsf{T\text{-}IND}$ we can break DDH with the same probability. □

4.3.3 Distributed Controllable Linkability

Due to the fact that our proposed revocation mechanism relies on an always-online revocation authority, the attack potential is significantly higher than in case of an offline authority. Essentially, we want to ensure that an attacker cannot steal the master linking key mlk by compromising such an authority. In order to prevent such a single point of failure, we introduce threshold AoN-PKEET*. This in turn allows us to realize distributed controllable linkability. Thereby, we reduce the trust in the linking authority and also obtain more robustness. In a similar manner Ghadafi [Gha14] introduced distributed tracing, such that multiple opening authorities must cooperate in order to open a signature.

The idea is to distribute the trapdoor $\mathsf{tk} \leftarrow \mathsf{Aut}(\mathsf{sk})$ of an AoN-PKEET* primitive among n entities using a (t, n)-secret sharing scheme. Then, the cooperation of at least t authorities is required to recover the trapdoor tk or to employ the trapdoor to perform equality tests on encrypted data.

Formal Model. We define threshold AoN-PKEET* as a tuple of algorithms $\mathcal{T}\text{-}\mathcal{PKEQ}^* = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Aut}, \mathsf{DKAut}, \mathsf{TShare}, \mathsf{TSCom})$. DKAut computes the shares for the trapdoor key, TShare computes the trapdoor shares for given ciphertexts, and TSCom performs the equality test based on a given set of shares.

$\mathsf{DKAut}(\mathsf{tk}, t, n)$**:** Takes a trapdoor key tk, a threshold t, and a number of total shares n, and returns trapdoor shares $(\mathsf{tk}_i)_{i=1}^n$, such that a subset of at least t entities is required to perform equality tests.

$\mathsf{TShare}(T, T', \mathsf{tk}_i)$**:** Takes two ciphertexts $(T,\ T')$ and a trapdoor share tk_i, and returns corresponding shares C_i and C'_i for the equality test.

$\mathsf{TSCom}(\{C_i, C'_i\}_{i\in\mathcal{I}})$**:** Takes a set of shares $\{C_i, C'_i\}_{i\in\mathcal{I}}$ with $|\mathcal{I}| \geq t$, and combines the shares to perform the plaintext equality test. It returns `true` if T and T' encrypt the same (unknown) message and `false` otherwise.

We adapt TShare and TSCom to return appropriate shares and the corresponding revocation token, respectively. We denote an invocation that returns the corresponding shares as $\{C_i, \perp\} \leftarrow \mathsf{TShare}(T, \perp, \mathsf{tk}_i)$ and an invocation that combines the shares to compute the revocation token as $\mathsf{t} \leftarrow \mathsf{TSCom}(\{C_i, \perp\})$.

Instantiation based on ElGamal and Shamir's secret sharing. Again, we assume a conventional ElGamal-based AoN-PKEET* scheme in a bilinear group setting, where the private key is $\mathsf{sk} = \xi \in \mathbb{Z}_p$ and the trapdoor for plaintext equality tests is $\mathsf{tk} = (\hat{r}, \hat{s} = \hat{r}^{\xi}) \in \mathbb{G}_2^2$. We omit the KeyGen, Enc, Dec algorithms for the sake of brevity and only present the relevant algorithms below.

$\mathsf{DKAut}(\mathsf{tk}, t, n)$**:** Given a trapdoor key $\mathsf{tk} = (\hat{r}, \hat{s} = \hat{r}^{\xi})$, a threshold t, and a total number of shares n, it computes the shares $(\mathsf{tk}_i)_{i=1}^n$. Therefore, it computes a polynomial $F(x) = \hat{r} \cdot \prod_{\ell=1}^{t-1} \hat{f}_\ell^{x^\ell}$ for random $\hat{f}_\ell \in \mathbb{G}_2$. Similarly, it computes a polynomial $G(x) = \hat{s} \cdot \prod_{\ell=1}^{t-1} \hat{g}_\ell^{x^\ell}$ for random $\hat{g}_\ell \in \mathbb{G}_2$. Finally, it returns the shares $(\mathsf{tk}_i = (i, \hat{r}_i \leftarrow F(i), \hat{s}_i \leftarrow G(i)))_{i=1}^n$.

$\mathsf{TShare}(T, T', \mathsf{tk}_i)$**:** Given two ciphertexts $(T, T') = ((T_1, T_2), (T'_1, T'_2))$ and a trapdoor share $\mathsf{tk}_i = (\hat{r}_i, \hat{s}_i)$, it returns the comparison shares $C_i = e(T_2, \hat{r}_i)$ and $D_i = e(T_1, \hat{s}_i)$ as well as $C'_i = e(T'_2, \hat{r}_i)$ and $D'_i = e(T'_1, \hat{s}_i)$.

$\mathsf{TSCom}(\{C_i, D_i, C'_i, D'_i\}_{i\in\mathcal{I}})$**:** Given a set of comparison shares $\{C_i, D_i, C'_i, D'_i\}_{i\in\mathcal{I}}$ with $|\mathcal{I}| \geq t$, the algorithm combines the shares to perform the plaintext equality test. Therefore, it computes S and S' as follows:

$$S = \prod_{i\in\mathcal{I}} C_i^{L_i} \cdot \left(\prod_{i\in\mathcal{I}} D_i^{L_i}\right)^{-1} \qquad S' = \prod_{i\in\mathcal{I}} C_i'^{L_i} \cdot \left(\prod_{i\in\mathcal{I}} D_i'^{L_i}\right)^{-1}$$

where $L_i = \prod_{j\in\mathcal{I}} \frac{j}{j-i}$ for $j \neq i$ are the Lagrange coefficients. Finally, it returns `true` if $S = S'$ and `false` otherwise.

The correctness of the above construction can be seen by inspection. Furthermore, the notion of token indistinguishability ($\mathsf{T\text{-}IND}$) as defined in Section 4.3.2 also holds for the threshold variant of AoN-PKEET*.

4.4 GSs with Linking-Based Revocation

We now specify a GSS with linking-based revocation as a tuple $\mathcal{GS}\text{-}\mathcal{LBR}$ = (GkGen, UkGen, Join, Issue, GSig, GVf, Open, Judge, CheckStatus, Revoke). The algorithms that change due to our modifications as well as the additional algorithms CheckStatus and Revoke are as follows.

$\mathsf{GkGen}(1^\kappa, t, n)$: On input a security parameter κ, a threshold t, and a total number of shares n, the algorithm outputs a tuple $(\mathsf{gpk}, \mathsf{mok}, \mathsf{mik}, \mathsf{mlk}, (\mathsf{mlk}_i)_{i=1}^n)$. It runs $(\mathsf{pk}_e, \mathsf{sk}_e) \leftarrow \mathsf{KeyGen}(1^\kappa)$ of the (t, n)-threshold AoN-PKEET* scheme, sets $\mathsf{mok} = \mathsf{sk}_e$, and adds pk_e to gpk. Then, it runs $(\mathsf{tk}_{\mathsf{pub}}, \mathsf{tk}_{\mathsf{priv}}) \leftarrow \mathsf{Aut}(\mathsf{mok})$, sets the master linking key $\mathsf{mlk} = (\mathsf{tk}_{\mathsf{pub}}, \mathsf{tk}_{\mathsf{priv}})$, and adds $\mathsf{tk}_{\mathsf{pub}}$ to mik. Furthermore, it generates the shares $\mathsf{mlk}_i \leftarrow \mathsf{DKAut}(\mathsf{mlk}, t, n)$ for the n distributed linking authorities. The rest remains unchanged.

$\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$: On input the group public key gpk, a message M, and a signature σ, the algorithm determines whether the signature σ is valid with respect to the message M and the group public key gpk. It returns `true` if the signature is valid and `false` otherwise.

$\mathsf{CheckStatus}(\mathsf{RL}, \mathcal{L}, \sigma)$: On input a revocation list RL containing anonymous revocation tokens, a set of linking authorities $\mathcal{L}$, and a signature σ, this algorithm determines the revocation status of the signer corresponding to signature σ. Therefore, it interacts with t linking authorities $\mathcal{L}_i \in \mathcal{L}$ via the TShare algorithm and retrieves the corresponding shares for the computation of the revocation token. Afterwards, it uses the TSCom algorithm to combine these shares and to retrieve the final revocation token. If the revocation token exists on the RL, the signer has been revoked and it returns `true`. Otherwise, it has not been revoked and it returns `false`.

$\mathsf{Revoke}(\mathsf{gpk}, \mathsf{mik}, \mathbf{reg}, \mathsf{RL}, i)$: On input of the group public key gpk, the master issuing key mik, the registration table **reg**, the current revocation list RL, and a user i to be revoked, the algorithm computes the anonymous revocation token t_i corresponding to user i and adds it to the revocation list. It returns the updated revocation list $\mathsf{RL} = \mathsf{RL} \cup \{\mathsf{t_i}\}$.[2]

4.4.1 Discussion and Security

Computation of Revocation Tokens. Considering our conventional ElGamal example with revocation tokens of the form $e(T_2, \hat{r}) \cdot e(T_1, \hat{s})^{-1} = e(m, \hat{r})$, we observe that these tokens can be computed in two different ways.

Given m and $\hat{r}$: Given a message m, e.g., a user's certificate, and $\hat{r}$ allows to compute revocation tokens $\mathsf{t} = e(m, \hat{r})$. Thus, if the issuer is given access to $\hat{r}$, the revocation token t can be computed with the information available during the Issue algorithm and added to the registration table **reg**.

---

[2] Note that revocation can also be done based on a user's signature by means of mlk in which case the user's identity will not be required.

Given $\sigma = (T, \pi)$ and $\mathsf{mlk} = (\hat{r}, \hat{r}^{\xi})$: Given a signature $\sigma = (T, \pi)$ and the master linking key mlk, such a revocation token t can also be computed on the fly, *i.e.*, $\mathsf{t} \leftarrow \mathsf{Com}(T, \perp, \mathsf{mlk})$. Thus, such a token can be computed directly from a given signature which allows for anonymous revocation of users as signatures need not be opened before revocation.

Note that if the revocation tokens are precomputed and stored in the registration table **reg**, then an attacker who manages to get in possession of the registration table **reg** and RL (but not necessarily the mlk) can conceptually identify (open) all signers on RL as the same tokens can be found in **reg**. This is not possible in case the revocation tokens are computed on the fly as in this case the attacker cannot link entries on RL to entries in the registration table **reg** since the revocation tokens do not yield any useful information (cf. T-IND).

Revocation and Revocation Check. Eventually, in case of a revocation, the token—that can either be computed (1) by opening a signature first or (2) from the signature directly—is added to RL. The revocation check for a signature $\sigma = (T, \pi)$ then requires the computation of the token $\mathsf{t} \leftarrow \mathsf{Com}(T, \perp, \mathsf{mlk})$ and a simple look-up operation, *i.e.*, checking whether or not $\mathsf{t} \in \mathsf{RL}$, and consequently can be performed in time $\mathcal{O}(1)$.[3] Also note that, except for the party in possession of mlk, the anonymous revocation token does not yield any useful information and also does not endanger the privacy of signers.

Revocation Check with Threshold AoN-PKEET*. In case of threshold AoN-PKEET*, the RA does not hold the linking key mlk but always needs to contact at least t linking authorities (over authenticated and confidential channels) to compute the required tokens. Thereby, we assume that no t linking authorities can be compromised. Consequently, in contrast to the naive approach, breaking into the (always-online) RA does not reveal mlk. The only information that an attacker gains by compromising RAs is a list of revocation tokens t_i (and possibly the corresponding messages m_i and ciphertexts c_i). However, as argued in Section 4.3.2, this does not allow the attacker to compromise the overall anonymity of the scheme as this information does not allow her to distinguish other tokens t from random. Although we only cover passive attacks, this is a reasonable model because in case a RA gets compromised, the corresponding authentication key will be revoked and replaced and the adversary can only behave passive.

Figure 4.2 illustrates the idea of our secure instantiation of linking-based revocation. A verifier first verifies a given signature $\sigma = (T, \pi)$ via the GVf algorithm and afterwards it interacts with a RA by means of CheckStatus. The RA interacts with t LAs to retrieve the corresponding shares by means of $\{C_i, \perp\} \leftarrow \mathsf{TShare}(T, \perp, \mathsf{mlk}_i)$ and uses the $\mathsf{t} \leftarrow \mathsf{TSCom}(\{C_i, \perp\}_{i \in \mathcal{I}})$ algorithm to compute the revocation token t. After checking whether or not t exists on RL, the RA returns the corresponding decision via CheckStatus.

---

[3] Hash tables allow to check whether or not $\mathsf{t} \in \mathsf{RL}$ in constant time. For instance, employing cuckoo hashing [PR04] allows for a worst-case complexity of $\mathcal{O}(1)$.

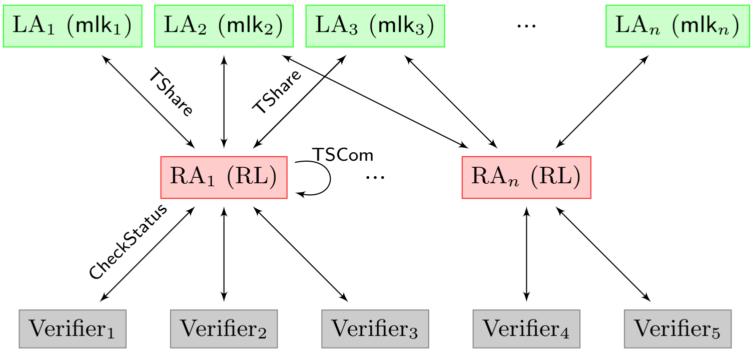

Figure 4.2: Schematic of our secure instantiation of linking-based revocation.

Since the RL as well as the tokens t do not endanger the privacy of group members (cf. $\mathsf{T\text{-}IND}$), the role of RAs can be distributed over multiple cloud services. Besides, we assume that no t LAs can be compromised at once and since the trapdoor shares mlk_i do not endanger the privacy of group members, these trapdoor shares mlk_i can also be safely distributed over multiple cloud services. Similar to traditional OCSP responses, the response from RAs must be signed. Although one could use the feature of verifiable controllable linkability [BDSS16] to prove that a given signature has or has not been revoked, this would add additional overhead for the verifier. If the signer has been revoked, the RA needs to perform a proof that the given signature can be linked to one specific entry on RL. If the signer has not been revoked, the RA needs to prove that the given signature has not been produced by any entity on RL and, hence, a naive instantiation of verifiable controllable linkability means that the proof increases linearly with the size of RL. In addition, a naive instantiation of verifiable controllable linkability would break backward unlinkability as verifiers would receive a proof that a specific signature links to a specific entry on RL (that must be publicly available in this case), which means that verifiers could link all signatures of revoked members. Although backward unlinkability can be achieved by using disjunctive zero-knowledge proofs (OR proofs), such proofs also introduce non-trivial overhead for the involved entities. Thus, we suggest that RAs sign the response. In order to reduce the communication and computational costs one could employ distributed OCSP (D-OCSP) [KS04], such that all RAs have a different signing key although they share the same public key.

Security Model. In contrast to other revocation mechanisms, the RA in our setting only returns a boolean decision, *i.e.*, whether or not the signer has been revoked. Thus, unlike other revocation mechanisms for group signatures, the RA does not reveal additional revocation-related information which would require to integrate the revocation feature into the formal security model of the GSS. Basically, LBR is an application of the feature of controllable linkability and, thus, we do not need to formally model our revocation mechanism in the security model. In fact, the role of the RA is already covered by the security model of group signatures with controllable linkability. Consequently, we also do not modify the security properties listed in Section 3.2. In addition, correctness of revocation follows from correctness of the GSS with controllable linkability.

4.5 Applying Linking-Based Revocation

For illustration purposes, we apply linking-based revocation to the *eXtremely Short Group Signature* scheme (XSGS) [DP06]. Using our generic compiler (cf. Chapter 3), XSGS can be turned into a GSS with controllable linkability for free (without any overhead) as the underlying encryption scheme is $\mathsf{IND\text{-}CCA2}$ secure. Since LBR is transparent for signers, we only need to modify the GkGen algorithm and we add $\mathsf{CheckStatus}$ and Revoke according to the model for GSs with controllable linkability. The scheme is as follows (cf. [DP06]):

$\mathsf{GkGen}(1^\kappa, t, n)$: The master opening key is $\mathsf{mok} = (\xi_1, \xi_2)$ for randomly chosen elements $\xi_1, \xi_2 \in \mathbb{Z}_p$. The master linking key is $\mathsf{mlk} = (\hat{r}, \hat{s} = \hat{r}^{\xi_1})$ for a randomly chosen element $\hat{r} \in \mathbb{G}_2$. The master issuing key consists of $\mathsf{mik} = (\gamma, \hat{r})$ for a randomly chosen element $\gamma \in \mathbb{Z}_p$. The master linking key mlk is distributed among n linking authorities via $\mathsf{mlk}_i = (\hat{r}_i, \hat{s}_i) \leftarrow \mathsf{DKAut}(\mathsf{mlk}, t, n)$. The group public key is $\mathsf{gpk} = (g_1, k, h = k^{\xi_1}, g = k^{\xi_2}, \hat{g}_2, \hat{w} = \hat{g}_2^\gamma) \in \mathbb{G}_1^4 \times \mathbb{G}_2^2$.

$\mathsf{Issue}(\mathsf{gpk}, \mathsf{mik}, \mathbf{reg})$: The Issue algorithm interacts with Join to add a user to the group. A user receives a membership certificate (A_i, x, y), such that $A_i^{x+\gamma} = g_1 h^y$, where $y \in \mathbb{Z}_p$ is known to the user only. The issuing authority adds the relevant information to the registration table **reg**.

$\mathsf{GSig}(\mathsf{gpk}, M, \mathsf{gsk_i})$: Given the group public key gpk, a message M, and a user's signing key $\mathsf{gsk_i} = (A, x, y) \in \mathbb{G}_1 \times \mathbb{Z}_p^2$, a signature is computed as follows. It randomly selects $\alpha, \beta \in \mathbb{Z}_p$ and encrypts the membership certificate:

$$T_1 = k^\alpha \qquad T_2 = Ah^\alpha \qquad T_3 = k^\beta \qquad T_4 = Ag^\beta$$

Then it sets $z = x\alpha + y$ and computes the non-interactive zero-knowledge proof of knowledge (α, β, x, z). It picks blinding values $r_\alpha, r_\beta, r_x, r_z \in \mathbb{Z}_p$ and computes the following values:

$$\begin{aligned}
&R_1 = k^{r_\alpha} \qquad\qquad R_3 = k^{r_\beta} \qquad\qquad R_4 = h^{r_\alpha}/g^{r_\beta} \\
&R_2 = e(T_2, g_2)^{r_x} \cdot e(h, w)^{-r_\alpha} \cdot e(h, g_2)^{-r_z} \\
&c = H(M, T_1, T_2, T_3, T_4, R_1, R_2, R_3, R_4) \\
&s_\alpha = r_\alpha + c\alpha \qquad s_\beta = r_\beta + c\beta \\
&s_x = r_x + cx \qquad s_z = r_z + cz
\end{aligned}$$

Finally, output the signature $\sigma = (T_1, T_2, T_3, T_4, c, s_\alpha, s_\beta, s_x, s_z)$.

$\mathsf{GVf}(\mathsf{gpk}, M, \sigma)$: Given the group public key gpk, a message M and a corresponding signature σ, verification is performed by checking the following relations:

$$k^{s_\alpha} = R_1 \cdot T_1^c \qquad k^{s_\beta} = R_3 \cdot T_3^c \qquad h^{s_\alpha}/g^{s_\beta} = R_4 \cdot \left(\tfrac{T_2}{T_4}\right)^c$$
$$e(T_2, g_2)^{s_x} \cdot e(h, w)^{-s_\alpha} \cdot e(h, g_2)^{-s_z} = R_2 \cdot \left(\tfrac{e(g_1, g_2)}{e(T_2, w)}\right)^c$$

If all of the above relations hold, the algorithm returns `true` and `false` otherwise.

$\mathsf{CheckStatus}(\mathsf{RL}, \mathcal{L}, \sigma)$: Given a revocation list RL consisting of revocation tokens of revoked members, a list of available linking authorities $\mathcal{L}$, and a signature σ, it determines the revocation status of the signatory corresponding to the signature σ in question. Therefore, it interacts with a subset of at least t linking authorities $\mathcal{I} \subseteq \mathcal{L}$, *i.e.*, $|\mathcal{I}| \geq t$, to retrieve t comparison shares $\{C_i, D_i, \perp, \perp\} \leftarrow \mathsf{TShare}((T_1, T_2), \perp, \mathsf{mlk}_i)$ as follows:

$$C_i = e(T_2, \hat{r}_i) \qquad D_i = e(T_1, \hat{s}_i).$$

Based on these comparison shares, it computes the revocation token t via $\mathsf{t} = \mathsf{TSCom}(\{C_i, D_i, \perp, \perp\}_{i \in \mathcal{I}})$ as follows:

$$L_i = \prod_{j \in \mathcal{I}, j \neq i} \frac{j}{j - i} \qquad \mathsf{t} = \prod_{i \in \mathcal{I}} C_i^{L_i} \cdot \left(\prod_{i \in \mathcal{I}} D_i^{L_i}\right)^{-1}$$

Return `true` (revoked) if $\mathsf{t} \in \mathsf{RL}$ and `false` (not revoked) otherwise.

$\mathsf{Revoke}(\mathsf{gpk}, \mathsf{mik}, \mathbf{reg}, \mathsf{RL}, i)$: Given the group public key gpk, the master issuing key mik, the registration table $\mathbf{reg}$, the current revocation list RL, and a user i to be revoked, the algorithm computes the revocation token $\mathsf{t}_i = e(A_i, \hat{r})$ for user i and adds it to the revocation list, *i.e.*, $\mathsf{RL} = \mathsf{RL} \cup \{\mathsf{t}_i\}$.[4]

Security Analysis. In order to apply linking-based revocation to the XSGS scheme we replaced the initially employed twin ElGamal encryption scheme with its AoN-PKEET* version according to our generic compiler (cf. Chapter 3). This transformation of the XSGS scheme into a GSS with controllable linkability is secure in the corresponding model described in Section 3.2 (cf. Theorem 1 in Section 3.4.1). Furthermore, token indistinguishability, as defined and shown for standard ElGamal in Section 4.3.2, also holds for twin ElGamal and consequently an adversary cannot learn anything from the anonymous revocation tokens on RL. Thus, the application of controllable linkability for revocation yields a secure revocation mechanism.

---

[4] Again, revocation can also be done based on a user's signature $\sigma = (T, \pi)$ by means of mlk in which case the user's identity will not be required.

4.6 Conclusion

Revocation mechanisms represent the major bottleneck [MFG^{+}12] in group signature schemes. However, the general belief that any online authority must be prevented unnecessarily restricts the efficiency and practicality of revocation in group signature schemes. In this chapter, we showed that the major drawbacks of existing revocation mechanisms, e.g., additional computations/updates for signers and verifiers, can be overcome by relying on an online revocation authority. Since many applications and services already rely on always-connected devices that permanently interact with cloud computing infrastructures, the introduction of such an online revocation authority is absolutely reasonable. Considering (1) the significant performance gain (constant-time revocation checks), (2) the transparency for signers as well as verifiers, and (3) the general applicability to well-established PB-GSSs, we claim that our approach advances the open issue of efficient membership revocation in the context of GSSs. Hence, linking-based revocation represents a valuable contribution to the existing portfolio of revocation mechanisms.

Part II

Side-Channel Attacks on Mobile Devices

5

Taxonomy of Side-Channel Attacks on Mobile Devices

For, you see, so many out-of-the-way things had happened lately, that Alice had begun to think that very few things indeed were really impossible.
— Lewis Carrol, Alice in Wonderland

Side-channel attacks on mobile devices have gained increasing attention since their introduction in 2007. While traditional side-channel attacks, such as power analysis attacks and electromagnetic analysis attacks, required physical presence of the attacker as well as expensive equipment, an (unprivileged) application is all it takes to exploit the leaking information on modern mobile devices. Given the vast amount of sensitive information that are stored on smartphones, the ramifications of side-channel attacks affect both the security and privacy of users.

In this chapter, we propose a new categorization system for side-channel attacks on mobile devices, which is necessary since side-channel attacks have evolved significantly since their extensive investigation during the smart card era in the 1990s. Besides this new categorization system, the extensive overview of existing attacks and attack strategies provides valuable insights on the evolving field of side-channel attacks on mobile devices.

We start with an introduction and motivation in Section 5.1. We discuss background information on side-channel attacks in Section 5.2 and we propose our new classification system in Section 5.3. We survey existing attacks in Sections 5.4–5.6, and we discuss observed trends in Section 5.7 and countermeasures in Section 5.8. Parts of this chapter are taken verbatim from [SMKM16].

Publication Data and Contribution

Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. *arXiv ePrint Archive, Report 1611.03748*, 2016. In submission

Contribution: Main author; Veelasha Moonsamy contributed to survey on wearables, IoT, and code analysis tools. Thomas Korak contributed to survey on side-channel attacks against smart cards.

5.1 Introduction

Side-channel attacks exploit (unintended) information leakage of computing devices or implementations to infer sensitive information. Starting with the seminal works of Kocher [Koc96], Kocher et al. [KJJ99], Quisquater and Samyde [QS01], and Mangard et al. [MOP07], many papers considered attacks against cryptographic implementations to exfiltrate key material from smart cards by means of timing information, power consumption, or electromagnetic (EM) emanation. These attacks have been classified along two orthogonal axes:

1. *Active* vs *passive*: Depending on whether the attacker actively influences the behavior of the device or only passively observes leaking information.
2. *Invasive* vs *semi-invasive* vs *non-invasive*: Depending on whether or not the attacker removes the passivation layer of the chip, depackages the chip, or does not manipulate the packaging at all.

With the era of cloud computing, the scope and the scale of side-channel attacks have changed significantly in the early 2000s. While early attacks required attackers to be in physical possession of the device, newer side-channel attacks are conducted remotely by executing software in the targeted cloud environment. With the advent of mobile devices, even more sophisticated side-channel attacks have been proposed since around the year 2010. Especially the following *key enablers* allow for more devastating attacks on mobile devices.

1. *Always-on and portability*: Mobile devices are always turned on and due to their mobility they can be easily carried around at all times. Thus, they are tightly integrated into our everyday lives.
2. *Bring your own device (BYOD):* To decrease the number of devices carried around, employees use private devices to process corporate data and to access corporate infrastructure, which clearly indicates the importance of secure computing platforms.
3. *Ease of software installation*: Due to the appification [ABB$^+$16] of mobile devices, *i.e.*, where there is an app for almost everything, additional software can be installed easily by means of established app markets. Hence, malware can also be spread easily and at a fast pace.

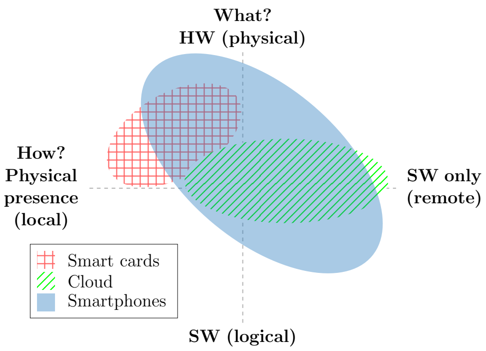

Figure 5.1: Scope of attacks for smart cards, cloud infrastructures, and smartphones.

4. *OS based on Linux kernel*: Modern mobile operating systems (OS), for example, Android, are based on the Linux kernel. The Linux kernel, however, has initially been designed for desktop machines and information or features that are considered harmless on these platforms turn out to be an immense security and/or privacy threat on mobile devices (cf. [ZYN$^+$15]).

5. *Features and sensors*: Last but not least, mobile devices include many features and sensors, e.g., accelerometer and GPS sensor, which are not present on traditional platforms. Due to the inherent nature of mobile devices (always-on and carried around, inherent input methods, etc.), such features often allow for devastating side-channel attacks [CC11, ZDH$^+$13].

High-Level Categorization. Considering these developments, we observe that the existing classification system does not meet these new attack settings anymore. Figure 5.1 illustrates a high-level view of our proposed categorization system and how *existing* side-channel attacks against smart cards, cloud computing infrastructures, and smartphones relate to it. We indicate the exploited information (What?) and how the adversary learns the information (How?) on the y-axis and x-axis, respectively. For instance, attackers exploit hardware-based information leakage (physical properties) [MOP07] of smart cards by measuring, for example, the power consumption with an oscilloscope. In contrast, side-channel attacks against cloud infrastructures do not (necessarily) require the attacker to be physically present as the attacker is able to remotely execute software. Usually, these attacks exploit microarchitectural behavior (like cache attacks [GYCH16]) or software features to infer secret information from co-located processes. Even more manifold and diverse side-channel attacks have been proposed for smartphones, as indicated by the larger area in Figure 5.1.

5.2 Background and Taxonomy

In this section, we provide an introduction to mobile security, define the general notion of side-channel attacks and we establish the boundaries between side-channel attacks and other attacks on mobile devices. Furthermore, we also discuss how smartphones allow for so-called *software-only attacks*.

5.2.1 A Primer on Smartphone Security

Modern mobile operating systems rely on two fundamental security concepts, *i.e.*, the concept of application sandboxing and the concept of permission systems. On Android the Linux kernel ensures the concept of sandboxed applications. Each application is assigned a unique user ID (UID), which allows the kernel to prevent applications from accessing resources of other applications. The permission system on the other hand allows applications to request access to specific resources outside of its sandbox, which typically includes resources that are considered as being sensitive. Besides these basic security concepts on the OS level, applications themselves rely on cryptographic primitives, cryptographic protocols, and dedicated security mechanisms to protect sensitive resources.

5.2.2 Concept of Side-Channel Attacks

Passive Side-Channel Attacks. The general notion of passive side-channel attacks can be described by means of three components, *i.e.*, *target*, *side-channel vector*, and *attacker*. A *target* represents anything of interest to possible attackers. During the computation or operation of a target, it influences a *side-channel vector* and thereby emits potential sensitive information. An *attacker* who is able to observe these side-channel vectors potentially learns useful information related to the computations or operations performed by the target.

Active Side-Channel Attacks. An active attacker tries to tamper with the device or to modify/influence the targeted device via a side-channel vector, e.g., via an external interface or via environmental conditions. Thereby, the attacker aims to influence computations/operations in a way that leads to malfunctioning, which in turn allows for attacks either indirectly via the leaking side-channel information or directly via the (erroneous) output of the targeted device.

Figure 5.2 illustrates the general notion of side-channel attacks. A target emits specific side-channel information as it influences specific side-channel vectors. For example, physically operating a smartphone via the touchscreen, *i.e.*, the touchscreen input represents the target, causes the smartphone to undergo specific movements and accelerations in all three dimensions. In this case, one possible side-channel vector is the acceleration of the device, which can be observed via the embedded accelerometer sensor and accessed by an app via the official Sensor API. The relations defined via the solid arrows, *i.e.*, *target* $\longrightarrow$ *side-channel vector* $\longrightarrow$ *attacker*, represent a passive side-channel attack. The relations defined via the dashed arrows, *i.e.*, *target* $\dashleftarrow$ *side-channel vector* $\dashleftarrow$

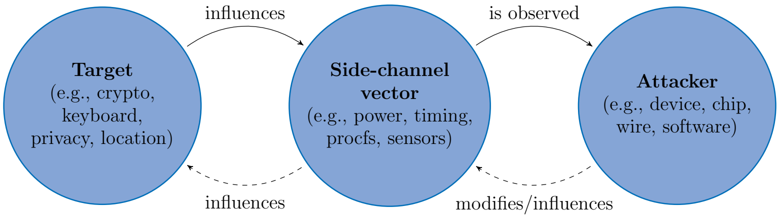
influences
is observed
Target
(e.g., crypto, keyboard, privacy, location)
Side-channel vector
(e.g., power, timing, procfs, sensors)
Attacker
(e.g., device, chip, wire, software)
influences
modifies/influences

Figure 5.2: General notion of passive (⟶) and active (←--) side-channel attacks.

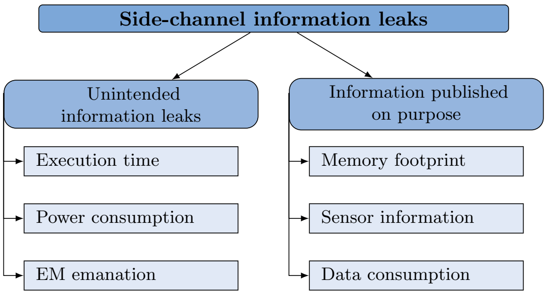

Figure 5.3: Categorization of side-channel information leaks.

attacker, represent an active side-channel attack where the attacker tries to actively influence or manipulate the target via a side-channel vector. Thereby, the attacker either tries (1) to enforce behavior that allows to bypass security mechanisms directly, or (2) to observe leaking side-channel information or the (sometimes erroneous) output of the targeted device.

5.2.3 Types of Side-Channel Information Leaks

Considering modern side-channel attacks, we identify two categories of information leaks, *i.e.*, *unintended information leaks* and *information published on purpose*, which are depicted in Figure 5.3. Side-channel attacks exploiting unintended information leaks are considered as "traditional" side-channel attacks. This type of information leak is considered as unintended because designers did not plan to leak this information on purpose. The second category of information leaks (*information published on purpose*) results from the ever-increasing number of features. For instance, specific features require the device to share (seemingly harmless) information and resources with all applications. Although this information is used by many legitimate applications for benign purposes, it sometimes turns out to leak sensitive information, e.g., the memory footprint [JS12] and the data-usage statistics (cf. Chapter 8).

5.2.4 Software-only Side-Channel Attacks

Irrespective of whether a physical property or a software feature are exploited, smartphones allow many of these information leaks to be exploited by means of *software-only attacks*, *i.e.*, without additional equipment like an oscilloscope. Besides, an attack scenario that requires the user to install an (unprivileged) application is entirely reasonable in an appified ecosystem.

SW-only side-channel attacks can be considered as runtime-information gathering (RIG) attacks [ZYN$^+$15], which refer to attacks that require a malicious app to run side-by-side with a victim app in order to collect runtime information of the victim. However, RIG attacks also include non side-channel attacks, where apps request a permission that is exploited for more obvious attacks, e.g., requesting access to the microphone in order to eavesdrop on conversations.

Similarly, we do not consider buffer overflow attacks as a means to launch active side-channel attacks because buffer overflow attacks represent a software vulnerability. Side-channel attacks, however, attack targets that are secure from a software perspective and still leak information unintentionally. Furthermore, software vulnerabilities, e.g., buffer overflow attacks, can be fixed easily, whereas side-channel attacks usually cannot be fixed or prevented that easily.

5.3 A New Categorization System

Our new categorization system classifies side-channel attacks along three axes:

1. *Passive* vs *active*: This category distinguishes between attackers that passively observe leaking side-channel information and attackers that also actively influence the target via any side-channel vector.

2. *Physical properties* vs *logical properties*: This category classifies side-channel attacks according to the exploited information, *i.e.*, whether the attack exploits physical properties or logical properties (software features).

3. *Local attackers* vs *vicinity attackers* vs *remote attackers*: Side-channel attacks are classified depending on whether or not attackers must be in physical proximity/vicinity of the target. *Local attackers* clearly must be in (temporary) possession of the device or at least in close proximity. *Vicinity attackers* are able to wiretap or eavesdrop the network communication of the target or to be somewhere in the vicinity of the target. *Remote attackers* only rely on software execution on the targeted device.

5.4 Local Side-Channel Attacks

In this section, we survey side-channel attacks that rely on local adversaries. This includes attacks that break cryptographic implementations, but also attacks that target the user's interaction with the device, *i.e.*, attacks resulting from the inherent nature of mobile devices. The surveyed attacks will show that the transition between local and vicinity attacks is seamless as the distance between victim and attacker can be increased, especially in the case of passive attacks.

5.4.1 Passive Attacks

Power Analysis Attacks. The power consumption of a computing device or implementation depends on the processed data and the executed instructions. Power analysis attacks exploit this information leak to infer sensitive information. Depending on whether a single measurement trace or multiple traces are required, we distinguish between simple power analysis (SPA) attacks and differential power analysis (DPA) attacks as defined by Kocher et al. [KJJ99].

Attacks. Traditional side-channel attacks that measure the power consumption via the voltage drop across a resistor inserted in the supply line are not that popular on smartphones. Nevertheless, a coarse-grained power-consumption monitoring allows to identify running applications, as shown in [YGCM15].

Electromagnetic Analysis Attacks. Another possibility to attack the leaking power consumption of devices is to exploit electromagnetic emanations, which are easier to obtain as there is no need to access the power line.

Attacks. Side-channel attacks exploiting the electromagnetic emanations of smart cards have also been applied on mobile devices. Gebotys et al. [GHT05] demonstrated attacks on software implementations of the Advanced Encryption Standard (AES) and Elliptic Curve Cryptography (ECC) on Java-based PDAs. Later on, Nakano et al. [NSN$^+$14a] attacked ECC and RSA implementations of the default crypto provider (JCE) on Android smartphones, and Belgarric et al. [BFMT16] attacked the ECDSA implementation of Android's Bouncy Castle.

Differential Computation Analysis. The idea of white-box crypto implementations is to embed the secret key into the software implementation in a way that prevents an attacker from extracting the key, even if the adversary has access to the source code itself. Therefore, the key and the algorithm itself are merged such that the key is hidden inside the code and cannot be separated easily. The white-box attack model assumes that the adversary has full control over the device and the execution environment.

Attacks. Bos et al. [BHMT16] showed that binary instrumentation can be used to observe and control the intermediate state of white-box crypto implementations. The instrumentation allows to precisely monitor the execution of the program and the observation of, e.g., the intermediate state and read/write accesses to memory, allow to profile program behavior. Based on the similarity to DPA attacks, these attacks are denoted as differential computation analysis (DCA) attacks. Although such attacks have not been applied on mobile devices so far, such an attack scenario works for these devices as well.

Smudge Attacks. The most common input method on mobile devices is the touchscreen, *i.e.*, users tap and swipe on the screen with their fingers. Due to the inherent nature of touchscreens, users always leave residues in the form of fingerprints and smudges on the screen.

Attacks. Aviv et al. [AGM$^+$10] pointed out that side-channel attacks can be launched due to interactions with the smartphone or touchscreen-based devices in general. More specifically, forensic investigations of smudges (oily residues from the user's fingers) on the touchscreen allow to infer unlock patterns. Even after cleaning the phone or placing the phone into the pocket, smudges seem to remain most of the time. Follow-up work considering an attacker who employs fingerprint powder to infer keypad inputs has been presented by Zhang et al. [ZXL$^+$12] and also an investigation of the heat traces—left on the screen due to finger touches—by means of thermal cameras has been performed [ATOY13].

Shoulder Surfing and Reflections. Touchscreens of mobile devices as well as displays in general optically/visually emanate the displayed content. Often these visual emanations are reflected by objects in the users' environment [BDU08].

Attacks. Maggi et al. [MGB11] observed that touchscreen input can be recovered by observing the visual feedback (pop-up characters) on soft keyboards during the user input. Therefore, they rely on cameras that are pointed directly on the targeted screen. Raguram et al. [RWG$^+$11, RWX$^+$13] observed that reflections, e.g., on the user's sunglasses, can also be used to recover input typed on touchscreens. However, the attacker needs to point the camera, used to capture the reflections, directly on the targeted user. Subsequently, they rely on computer vision techniques and machine learning techniques to infer the user input from the captured video stream. Xu et al. [XHW$^+$13] extended the range of reflection-based attacks by considering reflections of reflections. Although, they do not rely on the visual feedback of the soft keyboard but instead track the user's fingers on the smartphone while interacting with the device.

By increasing the distance between the attacker and the victim, e.g., by relying on more expensive and sophisticated cameras, some of these attacks might as well be considered as vicinity attacks.

Hand/Device Movements. Many input methods on various devices rely on the user operating the device with her hands and fingers. For instance, users tend to hold the device in their hands while operating it with their fingers.

Attacks. Similar to reflections, Shukla et al. [SKSP14] proposed to monitor hand movements as well as finger movements—without directly pointing the camera at the targeted screen—in order to infer entered PIN inputs. Similarly, Sun et al. [SJC$^+$16] monitored the backside of tablets during user input and detected subtle motions that allow to infer keystrokes. Yue et al. [YLF$^+$14] proposed an attack where the input on touch-enabled devices can be estimated from a video of a victim tapping on a touch screen.

Again, by increasing the distance between the attacker and the victim these attacks might also be considered as vicinity attacks, which demonstrates the seamless transition from local attacks to vicinity attacks.

5.4.2 Active Attacks

Active attacks against cryptographic implementations date back to the works of Boneh et al. [BDL97] (a.k.a. Bellcore attack) who attacked RSA crypto systems, especially implementations based on the Chinese Remainder Theorem (CRT), by relying on random hardware faults that result in the output of an erroneous signature. Later, Biham and Shamir [BS97] coined the term differential fault analysis (DFA) attacks and demonstrated that the introduction of faults and the observation of differences in the output ciphertext allow to recover the secret key of symmetric primitives. The principle of these attacks is to solve algebraic equations based on erroneous outputs (and valid outputs), which allows to break the crypto implementation.

Clock/Power Glitching. Varying the clock signal, e.g., overclocking, represents an effective method for fault injection attacks on embedded devices. One prerequisite for this attack is an external clock source. However, processors applied in smartphones typically have an internal clock generator making clock tampering impossible. Besides clock tampering, intended variations of the power supply represent an additional method for fault injection. With minor hardware modifications, power-supply tampering can be applied on most platforms.

Attacks. In [New16] it is shown how to disturb the program execution of an ARM CPU on a Raspberry PI by underpowering, *i.e.*, the supply voltage is set to GND for a short time. O'Flynn [O'F16] demonstrated that by shorting the power supply of an off-the-shelf Android smartphone a fault can be introduced that leads to an incorrect loop count.

Electromagnetic Fault Injection (EMFI). Transistors placed on microchips can be influenced by electromagnetic emanation. EMFI attacks take advantage of this fact. These attacks use short (in the range of nanoseconds), high-energy EM pulses to, e.g., change the state of memory cells resulting in erroneous calculations. In contrast to voltage glitching, where the injected fault is typically global, EMFI allows to target specific regions of a microchip by precisely placing the EM probe, e.g., on the instruction cache, the data memory, or CPU registers. Compared to optical fault injection, EMFI attacks do not necessarily require a decapsulation of the chip, which makes them more practical.

Attacks. Ordas et al. [OGSM16] demonstrate EMFI attacks targeting the AES hardware module of a 32 bit ARM processor. Rivière et al. [RNR+15] use EMFI attacks to force instruction skips and instruction replacements on modern ARM microcontollers. Considering the fact that ARM processors are applied in modern smartphones, EMFI attacks represent a serious threat for such devices.

Laser/Optical Faults. Optical fault attacks are another effective fault-injection technique. These attacks exploit the fact that a focused laser beam can change the state of a transistor resulting in, e.g., bit flips in memory cells. Compared to other fault-injection techniques (voltage glitching, EMFI), the effort for optical fault injection is high. First, decapsulation of the chip is a prerequisite to access the silicon with the laser beam. Second, finding the correct location for the laser beam to induce exploitable faults is also not a trivial task.

Attacks. First optical fault-injection attacks targeting an 8 bit microcontroller have been published by Skorobogatov and Anderson [SA02]. Inspired by their work, several optical fault-injection attacks have been published in the following years, most of them targeting smart cards or low-resource embedded devices (e.g. [vWWM11], [RSDT13]). The high decapsulation effort makes optical fault injection difficult to apply on modern microprocessors used in smartphones.

Temperature Variation. Operating a device outside of its specified temperature range allows to cause faulty behavior. Heating up a device can cause faults in memory cells. Cooling down the device has an effect on the speed RAM content fades away after power off (*remanence effect* of RAM).

Attacks. Hutter and Schmidt [HS13] present heating fault attacks targeting RSA implementations on AVR microcontrollers. FROST [MS13], on the other hand, is a tool for recovering disc encryption keys from RAM on Android devices by means of cold-boot attacks. Here the authors take advantage of the increased time data in RAM remains valid after power off due to low temperature.

Differential Computation Analysis. As an attacker in the white-box model has full control over the execution environment, she can produce erroneous or faulty outputs by manipulating intermediate values during the computation.

Attacks. Sanfelix et al. [SMdH15] showed that attackers with full control over the execution environment and the executed binary can also manipulate data during the program execution or manipulate the control flow. Similar to other fault attacks, the idea is to observe differences between normal outputs and erroneous outputs in order to break the cryptographic implementations.

NAND Mirroring. Data mirroring refers to the replication of data storage between different locations. Such techniques are used to recover critical data after disasters but also allow to restore a previous system state.

Attacks. The Apple iPhone uses a passcode and a hardware-based key to derive encryption keys. As a dedicated hardware-based key is used to derive these keys, brute-force attempts must be done on the attacked device. Further, brute-force attempts are discouraged by gradually increasing the waiting time between wrongly entered passcodes up to the point where the phone is wiped. NAND mirroring [Sko16] can be used to reset the phone state and, thus, can be used to brute-force the passcode. As the attacker influences (resets) the state of the device, this represents an active attack.

5.5 Vicinity Side-Channel Attacks

In this section, we briefly survey attacks that require the attacker to be in the vicinity of the targeted user/device. Thereby, the attacker, for example, compromises any infrastructure facility within the user's environment.

5.5.1 Passive Attacks

Network Traffic Analysis. Although the encryption of messages transmitted between two parties hides the actual content, meta data like the overall amount of data is not protected. This information can be used to infer information about the transmitted content and about the communicating parties.

Attacks. Network traffic analysis has been extensively studied in the context of website fingerprinting attacks. These attacks [PNZE11, CZJJ12, WG13, JAA$^+$14, PLP$^+$16] wiretap network connections to observe traffic signatures, e.g., unique packet lengths, and inter-packet timings, in order to infer visited websites. While these attacks target the network communication in general, attacks explicitly targeting mobile devices also exist. Stöber et al. [SFSM13] showed that eavesdropping the UMTS transmission allows to fingerprint smartphones due to the background traffic generated by installed apps. Conti et al. [CMSV16] consider an adversary who controls WiFi access points and, thereby, infer specific app actions like sending mails. In a similar setting, the feasibility to fingerprint apps and actions performed in apps based on traffic analysis techniques has been demonstrated (cf. [WYKH15, MLLB15, TSCM16, AK16]).

While the above presented attacks exploit logical properties, *i.e.*, the fact that encrypted packets do not hide meta data, a recent work by Schulz et al. [SKH$^+$16] showed that also hardware properties can be exploited. They exploit the electromagnetic emanation of Ethernet cables to eavesdrop on transmitted packets on the wire, which allow them to observe parts of the transmitted Ethernet frames.

USB Power Analysis. Similar to power analysis attacks, researchers have shown that modified charging stations can be used to collect power traces that allow to infer sensitive information about users and mobile devices.

Attacks. Conti et al. [CNRS16] demonstrated that wall-socket smart meters that capture the power consumption of plugged devices can be used to identify specific users/notebooks. Although Conti et al. demonstrated the power of their attack by using notebooks, it is likely that the same attack works for smartphones as well. In a similar setting, Yang et al. [YGZ$^+$16] successfully demonstrated that power traces collected via USB charging stations can be used to infer visited websites. Such attacks even work in case dedicated protection mechanisms, e.g., adapters that block data pins on the USB cable, are in place.

WiFi Signal Monitoring. WiFi devices continuously monitor the wireless channel (channel state information (CSI)) to efficiently transmit data. This is necessary as environmental changes cause the CSI values to change.

Attacks. Ali et al. [ALWS15] observed that even finger motions impact wireless signals and cause unique patterns in the time-series of CSI values. In a setting with a sender (notebook) and a receiver (WiFi router), they showed that keystrokes on an external keyboard cause distortions in the WiFi signal. By monitoring how the CSI values change, they are able to infer the entered keys. Later on, Zhang et al. [ZZT$^+$16] inferred unlock patterns on smartphones via a notebook that is connected to the wireless hotspot provided by the smartphone. Li et al. [LML$^+$16] further improved these attacks by considering an attacker controlling only a WiFi access point. They infer the PIN input on smartphones and also analyze network packets to determine when the sensitive input starts.

5.5.2 Active Attacks

Network Traffic Analysis. Network traffic analysis has already been discussed in the context of passive side-channel attacks. However, active attackers might learn additional information by actively influencing the transmitted packets, e.g., by delaying packets.

Attacks. He et al. [HYG$^+$14] demonstrated that an active attacker, e.g., represented by an Internet Service Provider (ISP), could delay HTTP requests from Tor users in order to increase the performance of website fingerprinting attacks. The idea is that instead of observing the generated traffic for all resources on a webpage in parallel, *i.e.*, the response packets from multiple requests in parallel overlap, an attacker delays the packet requesting a resource until the response from the previous request has been fully retrieved.

5.6 Remote Side-Channel Attacks

The attacks surveyed in this section can be executed remotely and target devices at a much larger scale.

5.6.1 Passive Attacks

Linux-inherited procfs Leaks. The Linux kernel releases information that is considered as being harmless via the procfs. For example, the memory footprint (total virtual/physical memory size) of each application via `/proc/[pid]/statm`, CPU utilization times via `/proc/[pid]/stat`, number of context switches via `/proc/[pid]/status`, and also system-wide information like interrupt counters via `/proc/interrupts` and context switches via `/proc/stat`.

Attacks. Jana and Shmatikov [JS12] observed that the memory footprint of the browser correlates with the rendered website and, thus, allows to infer a user's browsing behavior. Chen et al. [CQM14] exploited this information to detect *Activity* transitions within Android apps. They observed that the shared memory size increases by the size of the graphics buffer in both processes, *i.e.*, the app process and in the window compositor process, due to the IPC communication between the app and the window manager. Besides, they consider CPU utilization and network activity in order to infer the exact activity later on.

Simon et al. [SXA16] exploit the number of interrupts and context switches to infer text entered via swipe input methods. They observed that the number of interrupts and context switches correlates with the user's finger movements across the keyboard when transitioning from letter to letter. Diao et al. [DLLZ16] inferred unlock patterns and apps running in the foreground by exploiting interrupt time series of the touchscreen controller. Besides, also the power consumption is released via the procfs, which allows to infer the number of entered characters on the soft keyboard [YGCM15].

Data-Usage Statistics. Android keeps track of the amount of incoming and outgoing network traffic on a per-application basis. These statistics allow users to keep an eye on the data consumption of any app.

Attacks. Data-usage statistics are captured with a fine-grained granularity, *i.e.*, packet lengths of single TCP packets can be observed. Zhou et al. [ZDH$^+$13] demonstrated that by monitoring the data-usage statistics an adversary can infer disease conditions accessed via *WebMD*, the financial portfolio via *Yahoo! Finance*, and also a user's identity by observing the data-usage statistics of the *Twitter* app and exploiting the public Twitter API. Later, we showed [SGKM16] (cf. Chapter 8) that the data-usage statistics can also be exploited to fingerprint websites even though the traffic is routed through the anonymity network Tor.

Page Deduplication. To reduce the overall memory footprint of a system, (some) operating systems[1] search for identical pages within the physical memory and merge them—even across different processes—which is called page deduplication. As soon as one process intends to write onto such a deduplicated page, a copy-on-write fault occurs and the process gets its own copy of the previously shared memory region again.

Attacks. Copy-on-write faults have been exploited by Suzaki et al. [SIYA11] and recently Gruss et al. [GBM15] demonstrated the possibility to measure the timing differences between normal write accesses and copy-on-write faults from JavaScript code. Based on these timings they suggest to fingerprint visited websites by allocating memory that stores images found on websites. If the user browses the website with the corresponding image, then at some point the OS detects the identical content in the pages and deduplicates these pages. By continuously writing to the allocated memory, the attacker might observe a copy-on-write fault indicating that the user browses the corresponding website.

Microarchitectural Attacks. Modern computer architectures include many components that aim to improve the overall effectiveness and performance. For instance, CPU caches bridge the gap between the latency of main memory accesses and the fast CPU clock frequencies. Microarchitectural attacks exploit specific effects like the timing behavior of these components, e.g., branch prediction units and CPU caches, in order to learn sensitive information about executed instructions, code paths, etc. As CPU caches have been shown to represent a powerful source of information leaks, we focus on cache attacks.

Attacks. Cache-timing attacks against the AES have already been investigated on Android-based mobile devices. For instance, Bernstein's cache-timing attack [Ber05] has been launched on development boards [WHS12, WWAS14, ZMHS16] and we [SP13b, SG14] (cf. Chapter 6) also investigated these attacks

---

[1] For instance, the Android-based CyanogenMod OS allows to enable page deduplication.

on Android smartphones. In addition, we also demonstrated [SP13a] more fine-grained attacks like, e.g., Evict+Time [TOS10], against the AES on smartphones. These attacks relied on privileged access to precise timing measurements, but as stated by Oren et al. [OKSK15] cache attacks can also be exploited via JavaScript and, thus, do not require native code execution anymore. They even demonstrated the possibility to track user behavior including mouse movements as well as browsed websites via JavaScript-based cache attacks. In a recently co-authored paper by Lipp et al. [LGS$^+$16] we even show that all existing cache attacks, including the effective Flush+Reload attack [YF14], can be applied on modern Android-based smartphones without any privileges. While early attacks on smartphones exclusively targeted cryptographic implementations, more recent work also shows that user interactions (touch actions and swipe actions) can be inferred through this side channel. Similar investigations of Flush+Reload on ARM have also been conducted by Zhang et al. [ZXZ16].

As some of these attacks actively influence the behavior of the victim, e.g., the execution time, some of these microarchitectural attacks can also be considered as active attacks. For a more detailed survey about microarchitectural attacks we refer to the recent survey paper by Ge et al. [GYCH16].

Sensor-based Keyloggers. Cai et al. [CMC09] and Raij et al. [RGKS11] were one of the first to discuss possible privacy implications resulting from mobile devices equipped with cameras, microphones, GPS sensors, and motion sensors in general. Nevertheless, a category of attacks that received the most attention are sensor-based keyloggers. These attacks are based on two observations. First, smartphones are equipped with lots of sensors—both, motion sensors as well as ambient sensors—that can be accessed without any permission, and second, these devices are operated with fingers while being held in the users' hands. Hence, the following attacks are all based on the observation that users tap/touch/swipe the touchscreen and that the device is slightly tilt and turned during the operation.

Attacks. In 2011, Cai and Chen [CC11] observed a correlation between entered digits on touchscreens and accelerometer sensor readings that can be exploited for motion-based keylogging attacks. Following this work, Owusu et al. [OHD$^+$12] extended the attack to infer single characters and Aviv [Avi12, ASBS12] investigated the accelerometer to attack PIN and pattern inputs. Subsequent publications [CC12, MVBC12, XBZ12] also considered the combination of the accelerometer and the gyroscope in order to improve the performance as well as to infer even longer text inputs [PSM15]. Since these sensors can also be accessed from JavaScript, motion-based keylogging attacks have even been successfully demonstrated via websites [MTSH16a, MTSH16b]. Some browsers even continue to execute JavaScript code, although the user closed the browser.

While the above surveyed attacks exploit different types of motion sensors, e.g., accelerometer and gyroscope, keylogging attacks can also be launched by exploiting ambient sensors. To the best of our knowledge, we [Spr14] (cf. Chapter 7) currently presented the only attack that exploits an ambient sensor, namely the ambient-light sensor, in order to infer a user's touch actions.

As demonstrated by Simon and Anderson [SA13], PIN inputs on smartphones can also be inferred by continuously taking pictures via the front camera and investigating the relative changes of objects in subsequent pictures that correlate with the entered digits. Fiebig et al. [FKH14] demonstrated that the front camera can also be used to capture the screen reflections in the user's eyeballs, which also allows to infer user input. In a similar manner, Narain et al. [NSN14b] and Gupta et al. [GSAV16] showed that tap sounds (inaudible to the human ear) recorded via smartphone stereo-microphones can be used to infer typed text on the touchscreen. However, these attacks require dedicated permissions to access the camera and the microphone, which might raise the user's suspicion during the installation. In contrast, the above presented motion and ambient sensors can be accessed without any permission.

For a more complete overview of sensor-based keylogging attacks we refer to the survey papers by Hussain et al. [HAZ$^+$16] and Nahapetian [Nah16]. Considering the significant number of papers that have been published in this context, user awareness about such attacks should be raised. Especially since a recent study by Mehrnezhad et al. [MTSH16a] found that the perceived risk of motion sensors, and especially ambient sensors, among users is very low.

Fingerprinting Devices/Users. The identification of smartphones (and users) without a user's awareness is considered a privacy risk. While obvious identification mechanisms like device IDs and web cookies can be thwarted, hardware imperfections of different components as well as software features can also be employed to stealthily fingerprint and identify devices and users, respectively.

Attacks. Bojinov et al. [BMNB14] and Dey et al. [DRX$^+$14] observed that unique variations of sensor readings (e.g., of the accelerometer) can be used to fingerprint devices. These variations are a result of the manufacturing process and are persistent throughout the life of the sensor/device. As these sensors can also be accessed via JavaScript, it is possible to fingerprint devices via websites [DBC16]. Similarly, such imperfections also affect the microphones and speakers [DBC14, ZDLZ14], which also allow to fingerprint devices. In addition, by combining multiple sensors even higher accuracies can be achieved [HHH16].

Kurtz et al. [KGB$^+$16] demonstrated that users can also be identified by fingerprinting mobile device configurations, e.g., device names, language settings, installed apps, etc. Hence, their fingerprinting approach exploits software properties (*i.e.*, configurations) only. Hupperich et al. [HMK$^+$15] proposed to combine hardware as well as software features to fingerprint mobile devices.

Location Inference. As smartphones are always carried around, information about a phone's location inevitably reveals the user's location/position. Hence, resources that obviously can be used to determine a user's location, e.g., the GPS sensor, require a dedicated permission. Yet, even without permissions, side-channel attacks allow to infer precise location information about users.

Attacks. More specifically, Han et al. [HON$^+$12], Nawaz et al. [NM14], and Narain et al. [NVBN16] demonstrated that the accelerometer and the gyroscope can be used to infer car driving routes. Similarly, Hemminki et al. [HNT13] showed that the transportation mode, e.g., train, bus, metro, etc., can be inferred via the accelerometer readings of smartphones. Besides the accelerometer and the gyroscope also ambient sensors can be used to infer driving routes. Ho et al. [HMSS15] exploit the correlation between sensor readings of the barometer sensor and the geographic elevation to infer driving routes.

Even less obvious side-channels that allow to infer driving routes and locations are the speaker status (e.g., speaker on/off) and the power consumption (available via the procfs). Zhou et al. [ZDH$^+$13] observed that the Android API allows to query whether or not the speaker is currently active, *i.e.*, boolean information that indicates whether or not any app is playing sound on the speakers. They exploit this information to attack the turn-by-turn voice guidance of navigation systems. By continuously querying this API, they determine how long the speaker is active, which allows to infer the speech length of voice direction elements, e.g., the length of “Turn right onto East Main Street”. As driving routes consist of many such turn-by-turn voice guidances, they use this information to fingerprint driving routes. Michalevsky et al. [MSV$^+$15] showed that the power consumption is related to the strength of the cellular signal, which depends on the distance to the base station and allows to infer a user’s location.

Speech Recognition. Eavesdropping conversations represents a severe privacy threat. Thus, a dedicated permission protects the access to the microphone. However, acoustic signals in the vicinity of a mobile device also influence the gyroscope measurements of this device, which also holds for human speech.

Attacks. Michalevsky et al. [MBN14] exploited the gyroscope sensor to measure acoustic signals in the vicinity of the phone and to recover speech information. Although they only consider a small set of vocabulary, *i.e.*, digits only, their work demonstrates the immense power of gyroscope sensors.

Soundcomber. Customer service departments often rely on automated menu services to interact with customers over the phone. A well-known example are interactive voice response systems supported by telephone services that use dual-tone multi-frequency (DTMF) signaling to transmit entered numbers, *i.e.*, an audio signal is transmitted for each key.

Attacks. As DTMF tones are also played locally, they can be recored and used to infer private information, e.g., credit card numbers entered while interacting with interactive voice response systems of credit card companies [SZZ$^+$11].

5.6.2 Active Attacks

An area of research that gains increasing attention among the scientific community are active side-channel attacks that can be exploited via SW-only attacks.

Rowhammer. The increasing density of memory cells within the DRAM requires the size of these cells to decrease, which in turn decreases the charging of single cells but also causes electromagnetic coupling effects between cells.

Attacks. Kim et al. [KDK+14] demonstrated that these observations can be used to induce hardware faults, *i.e.*, bit flips in neighboring cells, via frequent memory accesses to the main memory. Seaborn and Dullien [SD15] demonstrated how to possibly exploit these bit flips from native code and Gruss et al. [GMM16] showed that such bit flips can even be induced via JavaScript code. A recent paper [vdVFL+16] successfully demonstrates the exploitation of the Rowhammer bug to gain root privileges on Android smartphones.

5.7 Trend Analysis

Figure 5.4 classifies the attacks surveyed in Section 5.4–5.6 according to our new classification system. We indicate passive attacks with blue dots and active attacks with red triangles. The red (grid pattern) and green (diagonal lines) areas illustrate the scope of smart card attacks and cloud attacks, respectively. Based on this classification system we observe specific trends in modern side-channel attacks that will be discussed within the following paragraphs.

From Local to Remote Attacks. In contrast to the smart card era, the smartphone era faces a shift towards remote side-channel attacks that focus on both hardware properties and software features. The shift from local attacks towards remote attacks can be addressed to the fact that the attack scenario as well as the attacker have changed significantly. More specifically, side-channel attacks against smart cards have been conducted to reveal sensitive information that should be protected from being accessed by benign users. For example, in case of pay-TV cards the secret keys must be protected against benign users, *i.e.*, users who bought these pay-TV cards in the first place. In contrast, smartphones are used to store and process sensitive information and attackers interested in this information are usually not the users themselves but rather criminals, imposters, and other malicious entities that aim to steal this sensitive information from users. Especially the appification of the mobile ecosystem provides tremendous opportunities for attackers to exploit identified side-channel leaks via software-only attacks. Hence, this shift also significantly increases the scale at which attacks are conducted. While local attacks only target a few devices, remote attacks can be conducted on millions of devices at the same time.

From Active to Passive Attacks. While fault injection attacks have been quite popular on smart cards, (local) fault attacks are not that widely investigated on smartphones, at least at the moment. Consequently, we also observe that the variety of fault attacks conducted in the smart card era has decreased significantly in the smartphone era, which can be addressed to the following observations. First, the targeted device itself, e.g., a smartphone, is far more expensive than a smart card and, hence, fault attacks that potentially permanently break the device are only acceptable for very targeted attacks. Even in case of highly targeted attacks (cf. Apple vs FBI dispute) zero-day vulnerabilities might be chosen instead. Second, remote fault attacks seem to be harder to conduct as such faults are harder to induce via software execution. Currently, the only remote fault attack (software-induced fault attack) is the Rowhammer attack.

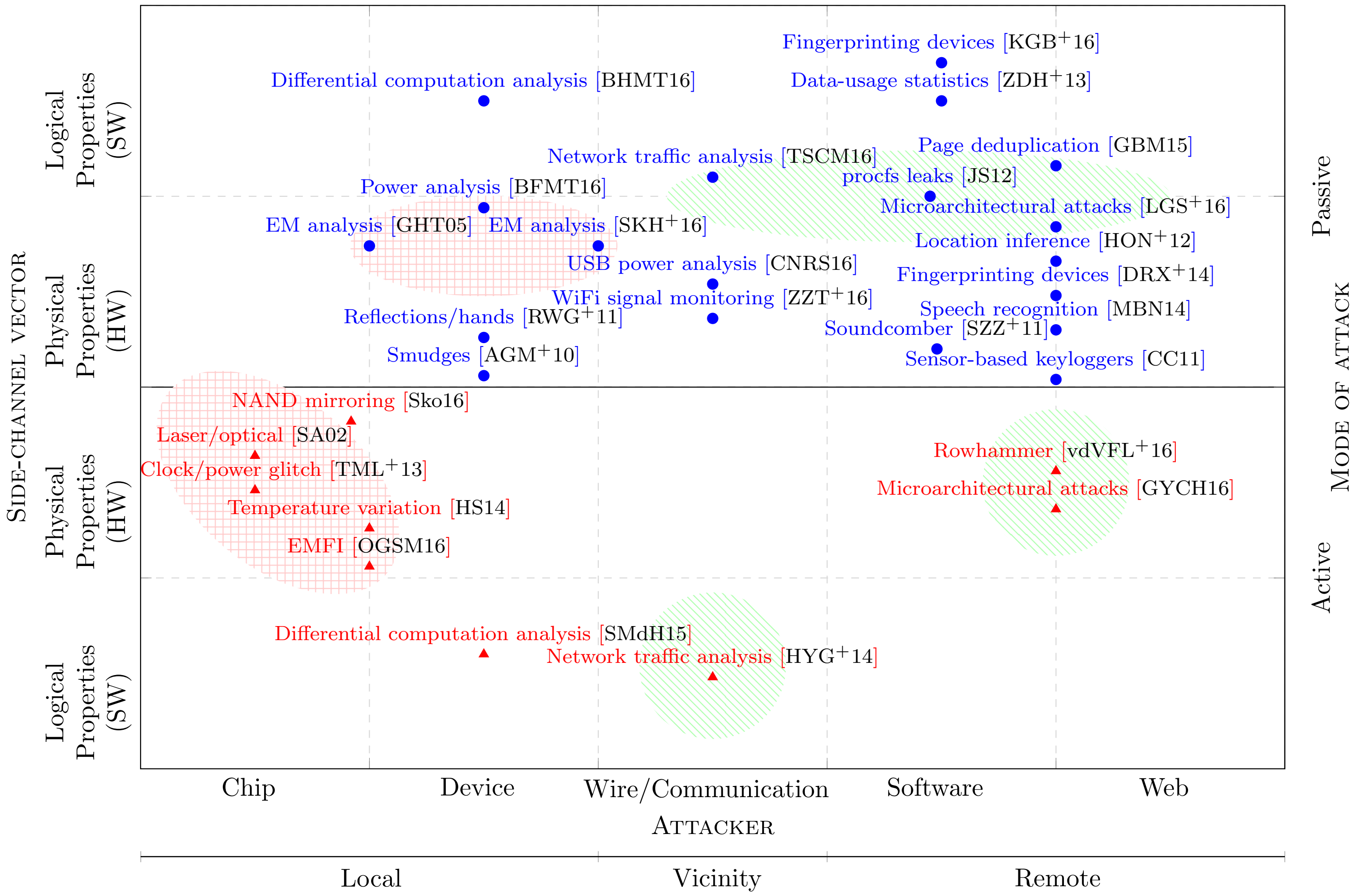

Figure 5.4: Classification of side-channel attacks: (1) active vs passive, (2) logical vs physical, (3) local vs vicinity vs remote.

Some microarchitectural attacks can also be considered as active attacks because the attacker influences the behavior of the victim. For example, cache attacks can be used to slow down the execution of the victim due to cache contention. However, this does not introduce a fault in the computation and, hence, Rowhammer currently represents the only software-induced fault attack.

Exploiting Physical and Logical Properties. In contrast to physical properties, logical properties do not result from any physical interaction with the device, but due to dedicated features provided via software. While traditional side-channel attacks mostly exploited physical properties and required dedicated equipment, more recent side-channel attacks exploit physical properties as well as logical properties. Although the majority of attacks on mobile devices still exploits physical properties, the exploitation of logical properties also receives increasing attention. Especially the procfs seems to provide an almost inexhaustible source for possible information leaks.

Empty Areas. As can be observed, a few areas in this categorization system are not (yet) covered. For instance, there is currently no active side-channel attack that exploits logical properties (software features) to induce faults. However, by considering existing passive attacks, one could come up with more advanced attacks by introducing an active attacker. Such an active attacker might, for example, block/influence a shared resource in order to cause malfunctioning of the target. For instance, considering the passive attack exploiting the speaker status (on/off) to infer a user's driving routes [ZDH$^+$13], one could easily induce faults by playing inaudible sounds in the right moment in order to prevent the turn-by-turn voice guidance from accessing the speaker. Thereby, the active attacker prevents the target (victim) from accessing the shared resource, *i.e.*, the speaker, and based on these induced "faults" an active attacker might gain an advantage compared to a passive attacker.

5.8 Discussion of Countermeasures

In this section, we discuss countermeasures against side-channel attacks. Overall we aim to shed light onto possible pitfalls of existing countermeasures and to stimulate future research for more generic countermeasures.

5.8.1 Local Side-Channel Attacks

Protecting Crypto Implementations. Masking of sensitive values, *i.e.*, the randomization of key-dependent values during cryptographic operations, or execution randomization are countermeasures for hardening the implementation against passive attacks like power analysis or EM analysis [MOP07]. Executing critical calculations twice allows to detect faults that are injected during an active side-channel attack [LRT12].

Protecting User Input. Mitigation techniques to prevent attackers from inferring user input on touchscreens are not that thoroughly investigated yet. Proposed countermeasures include, for example, randomly starting the vibrator to prevent attacks that monitor the backside of the device [SJC$^+$16], or to randomize the layout of the soft keyboard each time a user provides input to prevent smudge attacks [Avi12] as well as attacks that monitor the hand movement [SKSP14]. Aviv [Avi12] also proposes to align PIN digits in the middle of the screen and after each authentication the user needs to swipe down across all digits in order to hide smudges. Besides, Kwon and Na [KN14] introduce a new authentication mechanism denoted as *TinyLock* that should prevent smudge attacks against pattern unlock mechanisms. Krombholz et al. [KHH16] proposed an authentication mechanism for devices with pressure-sensitive screens that should prevent smudge attacks and shoulder surfing attacks. Raguram et al. [RWG$^+$11, RWX$^+$13] suggest to decrease the screen brightness, to disable visual feedback (e.g., pop-up characters) on soft keyboards, and to use anti-reflective coating in eyeglasses to prevent attackers from exploiting reflections.

5.8.2 Vicinity Side-Channel Attacks

Preventing Network Traffic Analysis. Countermeasures to prevent attackers from performing traffic analysis techniques on wiretapped network connections have been extensively considered in the context of website fingerprinting attacks. The main idea of these obfuscation techniques is to hide information that allows attackers to uniquely identify, for example, visited websites. Proposed countermeasures include [WCM09, LZC$^+$11, DCRS12, CNJ14, NCJ14] which, however, require the application as well as the remote server to cooperate. Furthermore, we already pointed out in [SGKM16] (cf. Chapter 8) that these countermeasures add overhead in terms of bandwidth and data consumption which might not be acceptable in case of mobile devices.

5.8.3 Remote Side-Channel Attacks

Permissions. A possible approach always discussed as a means to prevent specific side-channel attacks is to protect the exploited information or resource by means of dedicated permissions. However, studies [FHE$^+$12] have shown that permission-based approaches are not quite convincing. Some users do not understand the exact meaning of specific permissions, and others do not care about requested permissions. Acar et al. [ABB$^+$16] even attest that the Android permission system "has failed in practice". Despite these problems it seems to be nearly impossible to add dedicated permissions for every exploited information.

Keyboard Layout Randomization. In order to prevent sensor-based keylogging attacks that exploit the correlation between user input and the device movements observed via sensor readings, the keyboard layout of soft keyboards could be randomized [OHD$^+$12]. For instance, the Android-based CyanogenMod OS allows to enable such a feature for PIN inputs optionally. However, it remains an open question how this would affect usability in case of QWERTY keyboards and, intuitively, it might make keyboard input nearly impossible.

Limiting Access or Sampling Frequency. It has also been suggested to disable access to sensor readings during sensitive input or to reduce the sampling frequency of sensors. This, however, would prevent applications that heavily rely on sensor readings, e.g., pedometers. Specific resources, e.g., the ambient-light sensor, could be restricted to the OS exclusively (cf. Chapter 7).

Noise Injection. Shrestha et al. [SMS16] proposed a tool named *Slogger* that injects noise into sensor readings as soon as the soft keyboard is running. Therefore, Slogger relies on ADB capabilities in order to inject events into the files corresponding to the accelerometer and the gyroscope located in `/dev/input/`. In order to do so, Slogger relies on a tool that needs to be started via the ADB shell (in order to be executed with ADB capabilities). Das et al. [DBC16] also suggest to add noise to sensor readings in order to prevent device fingerprinting via hardware imperfections of sensors.

Preventing Microarchitectural Attacks. Although specific cryptographic implementations can be protected against cache attacks, e.g., bit-sliced implementations or dedicated hardware instructions to protect AES implementations, generic countermeasures against cache attacks represent a non-trivial challenge. However, we consider it of utmost importance to spur further research in the context of countermeasures, especially since cache attacks do not only pose a risk for cryptographic algorithms, but also for other sensitive information like keystroke logging [GSM15, LGS$^+$16].

App Guardian. While the above presented countermeasures aim to prevent specific attacks, App Guardian [ZYN$^+$15] represents a more general approach to defend against software-only attacks. App Guardian is a third party application that runs in user mode and employs side-channel information to detect RIG attacks (including software-only side-channel attacks). The principle of App Guardian is to stop the malicious application while the app to be protected is being executed and to resume the (potentially malicious) application later on. Although App Guardian still faces challenges it is a novel idea to cope with such side-channel attacks.

App Guardian seems to be a promising research project to cope with side-channel attacks on smartphones at a larger scale. However, an unsolved issue of App Guardian is the problem that it still struggles with the proper identification of applications to be protected. The effectiveness of App Guardian should be further evaluated against existing side-channel attacks and it might be interesting to extend it to cope with side-channel attacks conducted from within the browser, *i.e.*, to mitigate side-channel attacks via JavaScript.

5.9 Conclusion

In this chapter, we proposed a new categorization system for modern side-channel attacks on mobile devices. Although side-channel attacks are studied by the research community since the 1990s, the scope as well as the scale of such attacks has changed dramatically with the introduction of mobile devices in the last few years. One of the main key enablers of modern side-channel attacks is the immense number of sensors and features that can be exploited in order to conduct side-channel attacks remotely and without any additional equipment. Based on this observation, we aimed to categorize existing attacks according to our new classification system.

Considering the immense threat arising from side-channel attacks on mobile devices, a thorough understanding of information leaks and possible exploitation techniques is necessary. Based on this open issue, we surveyed existing side-channel attacks and identified commonalities of these attacks in order to systematically categorize existing attacks. With the presented classification system we aim to provide a thorough understanding of information leaks and hope to spur further research in the context of side-channel attacks as well as countermeasures and, thereby, to pave the way for secure computing platforms.

6

Time-Driven Cache Attacks

If I had an hour to solve a problem
I'd spend 55 minutes thinking about the problem
and 5 minutes thinking about solutions.
— Albert Einstein

Side-channel attacks usually rely on the divide-and-conquer approach. In the *divide step*, leaking information is collected for parts of the key. In the *conquer step*, this information is exploited to recover the full key. Focusing on both of these steps, we discuss potential enhancements of Bernstein's cache-timing attack against the Advanced Encryption Standard. Concerning the divide part, we analyze the impact of attacking different key-chunk sizes aiming at the extraction of more information from the overall encryption time under a secret key. Furthermore, we analyze an improvement of this attack presented by Aly and ElGayyar at AFRICACRYPT 2013—who suggest to exploit the minimum encryption time—according to its applicability on ARM Cortex-A platforms. For the conquer part, we use the optimal key-enumeration algorithm proposed by Veyrat-Charvillon et al. in order to reduce the exhaustive key-search phase compared to the current threshold-based approach. Our experimental results show that the optimal key-enumeration algorithm leads to more practical attacks. For instance, on a Samsung Galaxy SII, the remaining key bits to be searched can be reduced from 128 bits to 46 bits.

We motivate the investigation of cache-timing attacks on mobile devices in Section 6.1. We introduce the basics of Bernstein's timing attack in Section 6.2. We analyze possible improvements of the divide part in Section 6.3 and possible improvements of the conquer part in Section 6.4. We present experimental results in Section 6.5. Parts of this chapter are taken verbatim from [SG14].

Publication Data and Contribution

Raphael Spreitzer and Benoît Gérard. Towards More Practical Time-Driven Cache Attacks. In *Information Security Theory and Practice – WISTP 2014*, volume 8501 of *LNCS*, pages 24–39. Springer, 2014
Contribution: Main author; Implementation of attacks on smartphones, and analysis of divide part; Initial idea and analysis of conquer part contributed by Benoît Gérard.

6.1 Introduction

Kocher [Koc96] and Kelsey et al. [KSWH98] first mentioned the cache memory as a potential side channel, followed by Page [Pag02] who performed more detailed investigations of cache-based side channel attacks. Cache attacks against cryptographic primitives are separated into three categories: *time-driven attacks*, *access-driven attacks*, and *trace-driven attacks*. Time-driven cache attacks [TSS+03, Ber05] rely on the overall encryption time to recover the secret key via statistical methods. In contrast, access-driven[1] attacks [NS06b, TOS10, GBK11] and trace-driven attacks [AK06, GKT10, GK11] rely on more detailed knowledge about the implementation and the targeted hardware architecture. However, access-driven and trace-driven attacks require far less measurement samples than their time-driven counterpart. In short, there is a trade-off between the required knowledge and the required number of measurement samples.

An attack that has gained particular attention among the scientific community is Bernstein's [Ber05] cache-timing attack against T-table implementations of the Advanced Encryption Standard (AES). For instance, Neve [Nev06] and Neve et al. [NSW06] performed a detailed analysis of Bernstein's timing attack. Weiß et al. [WHS12] performed this attack against an AES implementation running in a trusted execution environment, and we [SP13b] investigated the applicability of Bernstein's attack on Android devices. Aly and ElGayyar [AE13] also investigated this timing attack and suggested to exploit the minimum encryption time, instead of the average encryption time. Saraswat et al. [SFKD14] investigated the applicability of Bernstein's timing attack against remote servers.

These investigations of Bernstein's cache-timing attack against AES T-table implementations emphasize the general applicability of this attack. However, especially on the ARM Cortex-A platform—the most commonly used architecture in modern mobile devices—investigations showed that timing information is leaking, but the complexity of the remaining key-search phase is usually very high. For instance, Weiß et al. [WHS12] presented a remaining key-search complexity of about 65 bits and we [SP13b] confirmed such an order of magnitude. While this might be within the range of sophisticated attackers, e.g.,

---

[1] Note that we did not consider the powerful Flush+Reload attack [YF14] back in 2014 as a dedicated flush instruction was not available on ARMv7 CPUs, e.g., on our targeted ARM Cortex-A8 and Cortex-A9 processors. Nevertheless, in a co-authored work [LGS+16] published in 2016 we demonstrated that Flush+Reload also works without a dedicated flush instruction.

state institutions—that have far more efficient techniques to recover information anyway—such attacks may not be possible for less sophisticated attackers.

Our motivation is to investigate potential improvements of time-driven cache attacks and to determine if such attacks could be performed on a broader scale. We propose and investigate multiple enhancements to the cache-based timing attack of Bernstein. Our practical experiments on ARM-based smartphones demonstrate that the number of key bits to be searched exhaustively can be reduced significantly, *i.e.*, from 128 bits to 46 bits, by applying the optimal key-enumeration algorithm. This confirms our initial proposition: timing attacks are within the range of less-powerful attackers.

6.2 Background

This section details the principle of the Advanced Encryption Standard, CPU caches, as well as the principle of Bernstein's cache attack.

6.2.1 Advanced Encryption Standard

The Advanced Encryption Standard (AES) [Nat01] is a block cipher that operates on 128-bit states—denoted as $\mathbf{S} = (\mathbf{s}_0, \ldots, \mathbf{s}_{15})$—and supports key lengths of 128, 192, and 256 bits.[2] The initial state is computed as $\mathbf{S}^0 = \mathbf{P} \oplus \mathbf{K}^0$, with $\mathbf{P} = (\mathbf{p}_0, \ldots, \mathbf{p}_{15})$ being the plaintext and $\mathbf{K}^0 = \left(\mathbf{k}_0^0, \ldots, \mathbf{k}_{15}^0\right)$ the initial round key. After the initial key addition, the round transformations (1) *SubBytes*, (2) *ShiftRows*, (3) *MixColumns*, and (4) *AddRoundKey* are applied multiple times, whereas the last round omits the *MixColumns* transformation. The number of rounds depends on the key length, e.g., 10 in case of a 128-bit key.

In the context of cache attacks the details of these round transformations are mostly irrelevant, since software implementations usually employ so-called T-tables, denoted as $\mathbf{T}_i$. These T-tables hold the precomputed round transformations and are composed of 256 4-byte elements. In every round the state bytes $\mathbf{s}_i$ are used to retrieve the precomputed 4-byte values which are then combined with a simple XOR operation to form the new state. The resulting state after the last round represents the ciphertext.

6.2.2 CPU Caches

The purpose of the CPU cache—a small and fast memory between the CPU and the main memory—is to buffer frequently used data and, thus, to enhance the performance of memory accesses. Today, most caches are divided into equally sized cache sets, each consisting of multiple cache lines. Contiguous bytes of the main memory are mapped to a specific cache set and can be placed in any cache line of this cache set. A dedicated replacement policy determines the cache line

---

[2] Although we consider a key length of 128 bits in this work, the outlined concepts can be applied to other key lengths analogously.

within a cache set where new data should be placed. ARM Cortex-A platforms employ a random-replacement policy.

6.2.3 Seminal Work: Bernstein's Timing Attack

Bernstein's [Ber05] timing attack against AES T-table implementations relies on the correlation of timing information of different plaintexts under a known key $\mathbf{K}$ and an unknown key $\widetilde{\mathbf{K}}$. The attack consists of the following four phases.

Study Phase. The attacker measures the encryption time of multiple plaintexts $\mathbf{P}$ under a known key $\mathbf{K}$. Without loss of generality, we assume that the zero-key is used. The information is stored in $\mathbf{t}[j][b]$ which holds the sum of all encryption times observed for plaintexts where $\mathbf{p}_j = b$, and $\mathbf{n}[j][b]$ which counts the number of encrypted plaintexts where $\mathbf{p}_j = b$.

Attack Phase. The attacker collects the exact same information as in the study phase, but this time under an unknown key $\widetilde{\mathbf{K}}$ that she wants to recover. The gathered information is stored in $\widetilde{\mathbf{t}}[j][b]$ and $\widetilde{\mathbf{n}}[j][b]$, respectively.

Correlation Phase. After gathering the required measurement samples, the attacker computes the so-called plaintext-byte signature (cf. [Nev06]) of the study phase as shown in Equation 6.1. The plaintext-byte signature of the attack phase is computed analogously, except that $\mathbf{v}[j][b]$, $\mathbf{t}[j][b]$, and $\mathbf{n}[j][b]$ are replaced with $\widetilde{\mathbf{v}}[j][b]$, $\widetilde{\mathbf{t}}[j][b]$, and $\widetilde{\mathbf{n}}[j][b]$, respectively.

$$\mathbf{v}[j][b] = \frac{\mathbf{t}[j][b]}{\mathbf{n}[j][b]} - \frac{\sum_j \sum_b \mathbf{t}[j][b]}{\sum_j \sum_b \mathbf{n}[j][b]} \tag{6.1}$$

Afterwards, the correlations of the plaintext-byte signature within the study phase and the attack phase are computed as outlined in Equation 6.2.

$$\mathbf{c}[j][b] = \sum_{i=0}^{255} \mathbf{v}[j][i] \cdot \widetilde{\mathbf{v}}[j][i \oplus b] \tag{6.2}$$

The correlations are sorted in a decreasing order and, based on a predefined threshold, a list of potential values for each key byte $\mathbf{k}_j$ is obtained.

Key-Search Phase. Usually, more than one value per byte is selected in the correlation phase (due to the selected threshold). Thus, the attacker performs an exhaustive search over all possible key candidates that can be formed from the selected values using a known plaintext-ciphertext pair.

6.3 Analysis of the Divide Part

In this section, we detail potential improvements regarding the gathering of the required timing information. Experimental results can be found in Section 6.5.

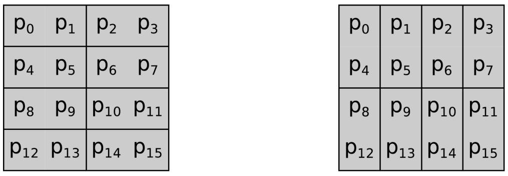

Figure 6.1: Possibilities to combine different key chunks: combination of bytes within a row (left) and combination of bytes within a column (right).

6.3.1 Attacking Different Key-Chunk Sizes

Bernstein [Ber05] considered the leaking timing information for exactly one key byte. We investigate the possibility to attack different key-chunk sizes. To this end, we briefly analyze the main concept of this idea as well as potential pitfalls.

Let us denote by n_{kc} the number of key chunks the whole key is comprised of, and by s_{kc} the size of each key chunk, *i.e.*, the number of possible values each key chunk might take. If we attack each key byte separately ($n_{kc} = 16$, $s_{kc} = 256$), then $\mathbf{t}[j][b]$ holds the total of all encryption times where plaintext byte $\mathbf{p}_j = b$ and $\mathbf{n}[j][b]$ counts the number of plaintexts where $\mathbf{p}_j = b$, for $0 \leq j < n_{kc}$ and $0 \leq b < s_{kc}$. Hence, attacking larger parts of the key at once leads to fewer key chunks n_{kc}, but a larger number of possible values per key chunk s_{kc}. In contrast, attacking smaller parts of the key leads to an increasing number of key chunks n_{kc} with fewer possible values s_{kc} for each of these key chunks.

Considering the plaintext as a 4×4 matrix, we observe that larger blocks can be formed in different ways. Figure 6.1 (left) illustrates the gathering of timing information for two consecutive plaintext bytes in one specific row. In contrast, Figure 6.1 (right) illustrates the combination of two plaintext bytes within one column. In case of the OpenSSL T-table implementation, the former approach collects timing information of two bytes accessing two different T-tables and the latter collects timing information of two bytes accessing the same T-table.

In our practical evaluation we gather timing information of two bytes accessing the same T-table. This case reduces the eventuality of corrupting key chunks with noise that might affect a specific T-table, e.g., in case many memory accesses of different processes compete for the same cache lines (cache thrashing). This can be illustrated as follows. Recall that look-up indices $\mathbf{s}_i$ access T-table $\mathbf{T}_j$, with $i \equiv j \mod 4$. If noise affects T-table $\mathbf{T}_0$, then in case of attacking two-byte key chunks that access the same T-table (e.g., $\mathbf{T}_0$), only two key chunks are affected by this noise. If we attack two-byte key chunks that access different T-tables (e.g., $\mathbf{T}_0$ and $\mathbf{T}_1$), four key chunks are affected by this noise.

Below we investigate potential pitfalls of attacking different key-chunk sizes.

Memory Requirements. The memory consumption of this timing attack depends on the number of key chunks n_{kc}, the size of each key chunk s_{kc}, and the size s_d of the used data type in bytes. Assuming the size of the data type $s_d = 8$, then for attacking each key byte separately ($n_{kc} = 16, s_{kc} = 256$) the size of one such data structure is 32 KB. Attacking two bytes at once ($n_{kc} = 8, s_{kc} = 256^2$) would result in 4 MB for each data structure and attacking four bytes at once ($n_{kc} = 4, s_{kc} = 256^4$) would result in 128 GB for each data structure. Thus, attacking more than two bytes at once is not applicable for devices with limited resources, e.g., mobile devices.

Number of Measurement Samples. The key-chunk size also has an impact on the noise reduction of the timing information. First, the larger the key chunks are, the smaller should be the algorithmic noise in the gathered encryption times. This positive effect of large chunks is counterbalanced by the fact that the larger the chunks are, the smaller is the number of samples obtained for a specific chunk value. For instance, let N be the number of encrypted plaintexts, then each possible value b of a specific block $\mathbf{p}_j$ is encrypted approximately $\frac{N}{s_{kc}}$ times. Thus, there is a trade-off between the algorithmic noise and the number of samples per value.

Indistinguishable Key Bits. The cache-line size determines the number of contiguous T-table elements to be loaded at once in the event of a cache miss. In case of a cache-line size of 32 (resp. 64) bytes, each cache line holds 8 (resp. 16) T-table elements. This in turn means that in general one cannot distinguish between accessed elements in one specific cache line and, therefore, the number of recoverable key bits is limited. Let us denote by s_c the cache-line size in bytes and by s_t the size of a T-table element in bytes. Then, according to Tromer et al. [TOS10] the number of non-recoverable key bits per key byte—at least for attacks considering only the first round of the AES—is given as $\log_2 \frac{s_c}{s_t}$. In case of a cache-line size of 32 (resp. 64) bytes, each cache line holds 8 (resp. 16) T-table elements, and the number of non-recoverable key bits per key byte are 3 (resp. 4). However, this is just a theoretical observation since in practice more advanced features of the architecture like *critical word first*[3] or *early restart*[4], as well as particular properties of the implementation, *i.e.*, disaligned T-tables [SP13a, TFAF13], usually lead to more information leakage.

6.3.2 Minimum Timing Information

Recently, Aly and ElGayyar [AE13] suggested to compute the correlation of the minimum encryption time within the study phase and the minimum encryption time within the attack phase. Therefore, $\mathbf{tmin}$ and $\widetilde{\mathbf{tmin}}$ hold the overall minimum encryption time of all encrypted plaintexts in the study phase and the attack phase, respectively. In addition, the data structure $\mathbf{umin}[j][b]$ holds the minimum encryption time of all plaintexts $\mathbf{p}$ where $\mathbf{p}_j = b$. The same holds for the attack phase, except that $\mathbf{umin}[j][b]$ and $\mathbf{p}$ are replaced by $\widetilde{\mathbf{umin}}[j][b]$ and $\widetilde{\mathbf{p}}$, respectively. The computation of the correlation is based on $\mathbf{umin}[j][b] - \mathbf{tmin}$

---

[3] *Critical word first* means that in case of a cache miss the missed word is loaded first and then the CPU continues its work while the remaining words are loaded into the cache.

[4] *Early restart* means that as soon as the critical word arrives, the CPU continues its work. In practice this would impose a serious side channel.

and $\widetilde{\mathbf{umin}}[j][b] - \widetilde{\mathbf{tmin}}$. Combining the timing information initially proposed by Bernstein and the minimum timing information, they claim to recover the whole secret key without a single exhaustive key-search computation.

We assume that the attacker computes the correlation with the timing information initially proposed by Bernstein and the correlation with the minimum timing information. So for each key byte $\mathbf{k}_i$ the attacker retrieves two sets of potential key candidates. Afterwards the attacker combines the sets of potential key candidates with the lowest cardinalities.

6.4 Analysis of the Conquer Part

In the conquer part, the information obtained during the divide step is exploited to recover the full key.

6.4.1 Combining Information from the Divide Part

We briefly describe two possible approaches to recover the full key. The first one is the one currently used in timing attacks, and the second one overcomes some of the mentioned shortcomings of the first one.

Threshold Approach. Currently, timing attacks rely on a threshold-based approach. This means that one fixes a threshold on the computed correlations and considers sub-key values as potential candidates if the corresponding correlation is larger than the threshold. Notice that one may use different thresholds for the different sub-keys, either because a profiling phase has shown different behaviors for different sub-keys or because they are dynamically computed.

The threshold approach is simple to implement but has two major drawbacks. The first one is that the actual key may not be found. If one of the sub-key values led to a small correlation, the key will never be tested in the search-phase and, thus, the attack will provide no advantage over exhaustive search. The second drawback is a loss of information since the order of sub-key values is not exploited in the search phase. Though Neve [Nev06, p 58] suggested that the key search "could start by the most probable key candidates", no indication is given how this should be accomplished.

Optimal Enumeration Approach. Veyrat-Charvillon et al. [VGRS12] recently proposed an optimal key-enumeration algorithm that solves the aforementioned problems at the cost of additional computations for generating the next full key to be tested. The algorithm requires a combination function that computes the score of the concatenation of two key chunks based on the scores of each chunk. Using such a combination function, a global score can be computed for each full key by combining the sub-key scores. The "optimal" notion comes from the fact that the algorithm ensures that keys will be generated in a decreasing order of global scores.

6.4.2 Evaluating the Key-Search Complexity

Threshold Approach. The lower bound on the key-search complexity is easy to obtain. Assuming that the attacker dynamically chooses a threshold for each targeted sub-key, it will, for a given sub-key, keep at least all values with a score larger than the correct one. The cardinality of the set of keys to be tested is then equal to the product of sub-key ranks. This lower bound, however, is very optimistic as such a magic threshold choice does not exist. Concerning the upper bound, it will depend on the allowed probability of missing the correct key. For given threshold(s), upper bounds on the key-search complexity and estimates of the missing-key probability can be obtained by simulating attacks. The upper bound being the size of the key-search space and the success probability being the probability that the actual key belongs to this space.

Optimal Enumeration Approach. Following the proposition of the key-enumeration algorithm in [VGRS12], Veyrat-Charvillon et al. [VGS13] proposed a key-rank estimation algorithm that estimates the key-search complexity of the optimal key-enumeration algorithm for a given combination function. More precisely, the algorithm requires the combination function, the scores obtained for one attack, and the correct key. When stopped, the program provides an interval $[2^x; 2^{x+\epsilon}]$ ensuring that the key rank lies in this range.

6.4.3 Choosing Thresholds and Combination Functions

Choosing Thresholds. After performing the correlation of the timing information from the study phase and the attack phase one retrieves a correlation vector $\mathbf{c}[j][b]$. The elements are sorted in a decreasing order and byte values b with a correlation value above a predefined threshold are considered as potential key candidates. Bernstein suggested a threshold based on the *standard error of the mean*. The idea is to take a key candidate b for a key byte $\mathbf{k}_j$ only into consideration if the difference of the byte value providing the highest correlation and the correlation of b is smaller than the established threshold.

Choosing Combination Functions. Ideally a combination function turns scores into sub-key probabilities that can be combined by multiplication. The so-called *Bayesian extension* in [VGRS12] uses Bayes' relation and a theoretical model of obtained scores to compute sub-key probabilities. In the context of timing attacks the scores obtained are similar to correlations. A minor modification of Bernstein's scoring function turns the scores into actual correlations without modifying the ordering of sub-key values. The modified formula for computing scores according to Pearson's correlation coefficient is outlined in Equation 6.3.

$$\mathbf{c}'[j][b] = \frac{\sum_{i=0}^{s_{kc}-1} \mathbf{v}[j][i] \cdot \widetilde{\mathbf{v}}[j][i \oplus b]}{\sqrt{\sum_{i=0}^{s_{kc}-1} \mathbf{v}[j][i]^2} \cdot \sqrt{\sum_{i=0}^{s_{kc}-1} \widetilde{\mathbf{v}}[j][i \oplus b]^2}} \tag{6.3}$$

Table 6.1: Device specifications of the test devices.

	Google Nexus S	**Samsung Galaxy SII**
Processor	Cortex-A8	Cortex-A9
L1 cache size/associativity	32 KB/4 way	32 KB/4 way
L1 cache-line size	64 byte	32 byte
L1 cache sets	128	256
Critical word first	yes	yes
Operating system	Android 2.3.6	Android 2.3.4

Then, using a Bayesian extension similar to the one in [GS13] (based on Fisher transform of correlation coefficients) we are able to estimate sub-key probabilities. The idea is that $\mathbf{f}[j][b] = \operatorname{arctanh}(\mathbf{c'}[j][b])$ follows a Gaussian distribution of variance $\sigma^2 = \frac{1}{s_{kc}-3}$ and with mean $\mu_c = 1$ if $\mathbf{k}_j = b$ and $\mu_w = 0$ otherwise. The estimated likelihood ratio between the probabilities of the j-th key chunk to be equal to b or not is then:

$$\begin{aligned}\mathbf{l}[j][b] &= \exp\left(\frac{(s_{kc}-3)(\mathbf{f}[j][b]-0)^2}{2} - \frac{(s_{kc}-3)(\mathbf{f}[j][b]-1)^2}{2}\right) \\ &= \exp\left((s_{kc}-3)\left(\mathbf{f}[j][b]-0.5\right)\right).\end{aligned}$$

The score of a full-key candidate $\mathbf{k}$ will be given by

$$S_{\text{Bayes}} = \prod_j \mathbf{l}[j][\mathbf{k}_j]. \tag{6.4}$$

To investigate the relevance of such Bayesian extension, Section 6.5 also contains data obtained with a different combination function that does not use Fisher transform. Natural combinations of correlation coefficients are operators $+$ and $\times$. The latter one is not relevant here since two values with correlation -1 will combine to a key with correlation 1, which is not desirable. We thus propose results obtained using $+$ as a combination function to complement our evaluation. In that case, the score of a full-key candidate $\mathbf{k}$ will be given by

$$S_{Add} = \sum_j \mathbf{c'}[j][\mathbf{k}_j]. \tag{6.5}$$

6.5 Experimental Results

In this section, we present the measurement setup and our practical evaluation.

6.5.1 Setup and Methodology

For the practical investigation of the suggested enhancements we employed two Android-based smartphones as shown in Table 6.1. Both devices are rooted to allow precise timing measurements via the ARM cycle counter register [ARM10].

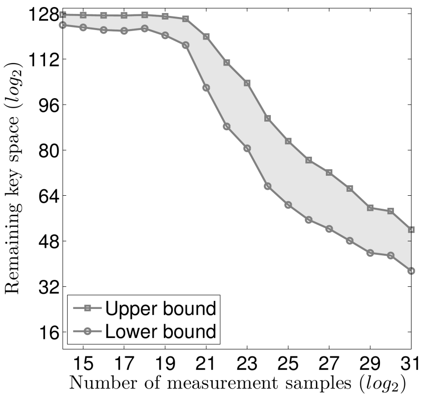

Figure 6.2: Rank evolution for one-byte key chunks.

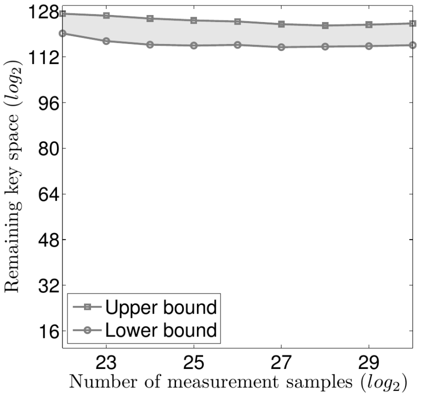

Figure 6.3: Rank evolution for two-byte key chunks.

Definitions. We define the gathering of the measurement samples under a known key and the gathering of the measurement samples under an unknown key as one *run*. Thus, one *run* of the attack application constitutes the gathering of the measurement samples for both of these phases. The *number of measurement samples* denotes the number of gathered samples in each of these two phases.

6.5.2 Attacking Different Key-Chunk Sizes

We launched Bernstein's attack multiple times on both devices, targeting either four bits, one byte, or two bytes of the key. Figures 6.2 and 6.3 illustrate the rank evolution for a specific number of measurement samples on the *Samsung Galaxy SII.* These plots show the average range (bounds) of key bits to be searched with the optimal key-enumeration algorithm after a specific number of measurement samples has been gathered. We clearly see that below 2^{21} measurement samples hardly any information leaks. Targeting four-bit chunks we observed a similar rank evolution as for one-byte chunks. Hence, we omitted this figure here.

According to Figure 6.3, the noise induced by the small number of samples per chunk value is significantly larger than the noise reduction obtained by considering larger chunks. The problem can be illustrated as follows. Figure 6.4 shows the plaintext-byte signature for one specific key byte for the study phase. The x-axis shows the possible chunk values of a plaintext byte and the y-axis shows the average encryption time for this specific byte subtracted by the overall average encryption time, after gathering 2^{30} samples. Figure 6.5 illustrates this information for the attack phase. We observe a visible pattern in both plots and the correlation yields a few possible key candidates. We also point out that most of the values lie in the range $[-0.5; 0.5]$ with peaks up to 2.5.

In contrast, Figure 6.6 and Figure 6.7 illustrate the chunk signatures for an attack targeting two-byte key chunks. Again, after gathering 2^{30} measurement

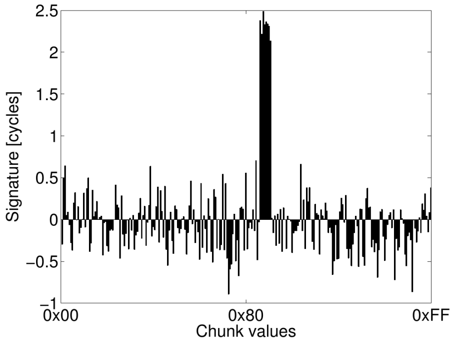

Figure 6.4: One-byte chunk signatures for the study phase of one run.

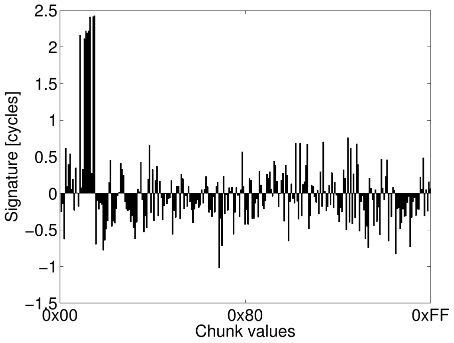

Figure 6.5: One-byte chunk signatures for the attack phase of one run.

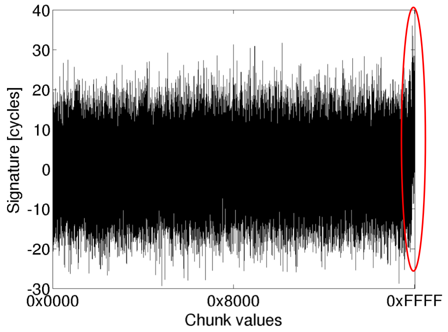

Figure 6.6: Two-byte chunk signatures for the study phase of one run.

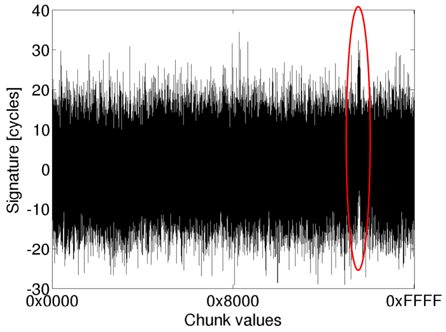

Figure 6.7: Two-byte chunk signatures for the attack phase of one run.

samples. Since the pattern is not that clearly visible we marked the similar peaks appropriately. Neve [Nev06] also performed an investigation of such signature plots for one-byte chunks. In accordance with his terminology, we note that both plots show rather *noisy* profiles with most values lying in the range $[-25; 25]$. Due to these *noisy* profiles the correlation does not reduce the key space significantly and the sub-key value for this specific key chunk cannot be determined. Though we also observed rather *noisy* profiles for attacks targeting one-byte chunks, most of the profiles established for one-byte chunks showed a clear pattern. In contrast, for two-byte key chunks we mostly observed plots where we could not find any specific pattern.

Our observations showed that attacking smaller key chunks works, while attacking larger key chunks seems to leak less information for the same (realistic) number of measurement samples. Targeting more than 2^{30} samples is not realistic anymore, at least for mobile devices, since a running time of more than eight hours does not represent a realistic scenario anymore.

6.5.3 Minimum Timing Information

Aly and ElGayyar [AE13] argue that noise usually increases the encryption time and, thus, the exploitation of the minimum timing information should significantly improve the timing attack. Their idea is to capture only one single measurement sample without noise, which is then stored and used for the correlation later on. They successfully launched their attack against a Pentium Dual-Core and a Pentium Core 2 Duo processor. However, contrary to their conclusion that this approach significantly improves the timing attack on Pentium processors, our results indicate that this approach does not work at all on ARM Cortex-A series processors. The reasons for this approach to fail are potentially manifold.

First, the OpenSSL implementation itself might be the reason. Aly and ElGayyar [AE13] attacked an implementation of OpenSSL that employs 4 T-tables. In contrast, we attacked an implementation that employs 5 T-tables.

Second, according to our understanding, gathering the minimum encryption time misses potential useful information. As Neve et al. [NSW06] put it, Bernstein's timing attack implicitly searches for cache evictions due to work done on the attacked device. Such cache evictions lead to cache misses within the encryption function and, thus, to slower encryptions. As a result, not only noise in terms of measurement noise increases the encryption time, but also cache misses increase the encryption time. While noisy encryption times do not carry useful information, encryption times where a cache miss occurred definitely do so. However, gathering the minimum timing information does not capture this information because the minimum timing information seeks for encryption times where a cache hit occurred. The problem is that once we observe a cache hit, *i.e.*, a fast encryption, we store this timing information. So this approach only searches for the cache hits and in the worst case, after a certain number of measurement samples, we observe a cache hit for all possible key bytes due to the random-replacement policy on ARM processors.

Furthermore, in the long run, the local minimum as well as the global minimum might become equal in which case these timings do not carry any information at all. Our practical evaluation showed that after a certain number of measurement samples on the Google Nexus S the minimum timing information $\mathbf{umin}[j][b] - \mathbf{tmin}$ equals 0 for most of the key bytes. Additionally, the random-replacement policy implemented on ARM Cortex-A series processors strengthens this reasoning. Though Aly and ElGayyar implemented this approach on Pentium processors that usually rely on a deterministic replacement policy, we consider the gathering of the minimum timing information also risky on such processors.

Concluding the investigation of the minimum timing information we point out that instead of using the minimum timing information, we stick to the exploitation of the timing information as proposed by Bernstein and only take

Table 6.2: Sample results on the Samsung Galaxy SII.

Run	**Key-Chunk Size**	**Samples**	**Bernstein**				**Minimum**
			Optimal Threshold	**Threshold**	**Key Enumeration**		**Optimal Threshold**
					(6.4)	(6.5)	
1	4 bits	2^{30}	50 bits	102 bits	79.3 - 104.4	86.5 - 112.3	84 bits
2	4 bits	2^{31}	32 bits	87 bits	58.9 - 82.4	62.1 - 92.6	88 bits
3	1 byte	2^{27}	42 bits	84 bits	59.3 - 80.3	58.4 - 81.8	107 bits
4	1 byte	2^{28}	41 bits	93 bits	55.7 - 77.7	56.2 - 79.2	104 bits
5	1 byte	2^{30}	23 bits	64 bits	36.6 - 44.9	36.4 - 46.5	100 bits
6	1 byte	2^{30}	32 bits	92 bits	49.1 - 70.1	49.1 - 70.3	100 bits
7	1 byte	2^{31}	24 bits	65 bits	37.4 - 52.5	37.6 - 53.6	99 bits
8	1 byte	2^{30}	20 bits	74 bits	36.5 - 45.6	36.0 - 46.4	105 bits
9	1 byte	2^{30}	32 bits	70 bits	43.9 - 64.1	44.0 - 66.6	106 bits
10	2 bytes	2^{30}	107 bits	123 bits	118.9 - 125.3	118.9 - 125.3	104 bits
11	2 bytes	2^{30}	96 bits	128 bits	115.5 - 122.2	115.0 - 123.2	114 bits
12	2 bytes	2^{30}	90 bits	124 bits	110.5 - 119.7	110.5 - 119.9	118 bits
13	2 bytes	2^{30}	110 bits	126 bits	120.2 - 126.7	120.3 - 126.7	115 bits

encryption times below a specific threshold into consideration. This approach also reduces the impact of noise if the threshold is selected properly.

6.5.4 Summary of Practical Results

Table 6.2 summarizes the results of our practical investigations on the Android-based Samsung Galaxy SII smartphone. For different runs, we provide the attacked key-chunk size as well as the number of samples acquired. The rest of the columns contain different $\log_2$ time complexities of the key-search phase depending on the exploited information, e.g., either Bernstein's timing information or the minimum timing information from [AE13], and depending on the conquer-phase technique. For the threshold-based conquer phase we provide the remaining key space for: (1) an optimal threshold choice, such that for each chunk the threshold is chosen in a way that only values with better scores than the correct one are selected (cf. Section 6.4.2), and (2) a threshold based on the *standard error of the mean* as suggested by Bernstein. The key-enumeration column contains bounds of the obtained key rank if the optimal key-enumeration algorithm from [VGRS12] is used. In Table 6.2 this column is separated into two, the first one being the result of the use of the Bayesian extension (cf. Equation 6.4) the second being obtained by addition of correlations (cf. Equation 6.5). We observe that using the optimal key-enumeration algorithm instead of the threshold-based approach has a positive impact on the key-search complexity. For instance, in case of run 5 and run 8—that require far more than 2^{60} keys to be tested in case of the threshold-based approach—the optimal key-enumeration algorithm recovers the key in less than 2^{46} tests. Considering the improvement of using the Bayesian extension, we observe that it is small when attacking one-byte chunks but becomes more significant when attacking four-bit chunks.

Furthermore, for ARM Cortex-A processors we cannot confirm that the minimum timing information improves the timing attack. The last column in Table 6.2 shows that the minimum timing information hardly leaks any information. Table 6.3 summarizes the exact same information for the Google Nexus

Table 6.3: Sample results on the Google Nexus S.

Run	Key-Chunk Size	Samples	Bernstein			Minimum
			Optimal Threshold	Threshold	Key Enumeration	Optimal Threshold
1	1 byte	2^{31}	64 bits	108 bits	83.5 - 109.1 bits	105 bits
2	1 byte	2^{30}	62 bits	119 bits	77.6 - 104.9 bits	95 bits
3	1 byte	2^{26}	66 bits	101 bits	79.0 - 101.3 bits	104 bits
4	1 byte	2^{30}	67 bits	96 bits	77.6 - 104.4 bits	108 bits
5	1 byte	2^{30}	58 bits	91 bits	69.6 - 90.5 bits	107 bits
6	1 byte	2^{28}	82 bits	95 bits	105.3 - 115.2 bits	110 bits
7	1 byte	2^{30}	61 bits	97 bits	84.0 - 99.0 bits	97 bits
8	2 bytes	2^{27}	121 bits	128 bits	127.8 - 128.0 bits	121 bits
9	2 bytes	2^{28}	116 bits	128 bits	125.0 - 127.8 bits	124 bits
10	2 bytes	2^{30}	118 bits	128 bits	126.1 - 127.2 bits	121 bits

S smartphone. Since we observed only minor differences between the usage of the Bayesian extension (Equation 6.4) and the usage of addition as a correlation function, we only provide the bounds based on the former approach.

6.6 Conclusion

In this chapter, we analyzed multiple improvements of Bernstein's timing attack. Based on the "divide-and-conquer" strategy we investigated different improvements for both of these two phases. Considering these improvements we also provide practical insights on two devices employing an ARM Cortex-A processor. We performed theoretical investigations of attacking smaller (resp. larger) chunks of the key and presented potential pitfalls. Our practical investigations on ARM-based smartphones showed that attacking one-byte chunks seems to be the best choice for resource-constrained devices. Furthermore, these investigations showed that the minimum timing information, as proposed by Aly and ElGayyar, does not improve the cache timing attack on the ARM Cortex-A devices at all.

The most important contribution of this work is the shift from the threshold-based approach for the selection of potential key candidates towards the application of the optimal key-enumeration algorithm. Instead of selecting potential key candidates on a threshold basis, we iterate over potential keys according to their probability for being the correct key. As our practical observation showed, this approach significantly reduces the complexity of the remaining key-search phase, which brings this attack to a complexity that can be considered as practical.

7

Exploiting the Ambient-Light Sensor

When you have a smartphone,
the things that it can do are kind of ridiculous and terrifying.
— Jonathan Nolan

In this chapter, we propose a new side-channel attack based on the ambient-light sensor. While recent advances in this area of research focused on the motion sensors and the camera as well as the sound, we investigate a less obvious source of information leakage, namely the ambient light. We successfully demonstrate that minor tilts and turns of mobile devices cause variations of the ambient-light sensor information. Furthermore, we show that these variations leak enough information to infer a user's personal identification number (PIN) input based on a set of known PINs. Our results show that we are able to determine the correct PIN—out of a set of 50 random PINs—within the first ten guesses about 80% of the time. In contrast, the chance of finding the right PIN by randomly guessing ten PINs would be 20%. Since the light-sensor information can be gathered without any specific permissions or privileges, the presented attack seriously jeopardizes the security and privacy of mobile-device owners.

We start with a short introduction in Section 7.1 and provide background information on the ambient-light sensor as well as the RGB(W) sensor in Sections 7.2 and 7.3, respectively. We discuss the attack scenario in Section 7.4 and our attack approach in Section 7.5. We provide an evaluation of our attack in Section 7.6 and discuss limitations in Section 7.7. We discuss possible mitigation techniques in Section 7.8 and compare our attack to similar attacks in Section 7.9. Parts of this chapter are taken verbatim from [Spr14].

Publication Data and Contribution

Raphael Spreitzer. PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices. In *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*, pages 51–62. ACM, 2014
Contribution: Main author.

7.1 Introduction

Besides providing useful information, sensors employed in mobile devices also represent a threat to the users' security and privacy. Although Android employs a permission system to prevent malicious access to specific device resources, many sensors can be accessed without any permission. This exacerbates the severity of information leaks since applications without any specific permission are able to exploit these side channels, which has already been demonstrated impressively. For instance, Aviv et al. [ASBS12] inferred the PIN input and the unlock pattern by exploiting the accelerometer sensor of smartphones. In addition, Miluzzo et al. [MVBC12] showed how to infer pressed keys from the accelerometer sensor readings in combination with the gyroscope sensor readings.

We investigate a less obvious source of information leakage, *i.e.*, the ambient-light sensor. Due to minor tilts and turns during the operation of the device, the sensor information allows an attacker to infer the input provided by the user.

7.2 Ambient-Light Sensor

Figure 7.1 illustrates the proximity and ambient-light sensor as well as the front camera on a Samsung Galaxy SIII. The ambient-light sensor measures the intensity of illumination of a surface and the information reported is given in SI *lux* units. This information is used to adapt the screen brightness. For instance, outside in direct sunlight the screen brightness must be increased, whereas in darker surroundings the screen is dimmed to reduce eye fatigue [Sam13].

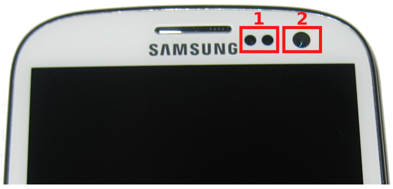

Figure 7.1: (1) Proximity and ambient-light sensor as well as (2) front camera.

The ambient-light sensor can be accessed via the Android Sensor API and a *rate* parameter determines how fast the events should be reported. Table 7.1 shows the sampling frequencies for the different rate parameters on a Samsung Galaxy SIII running Android 4.3. We also observed a sensor resolution of 1 *lux*.

Table 7.1: Sampling rates on the Samsung Galaxy SIII.

Rate parameter	Sample rate
SENSOR_DELAY_FASTEST (0)	~ 750 Hz
SENSOR_DELAY_GAME (1)	~ 49 Hz
SENSOR_DELAY_UI (2)	~ 15 Hz
SENSOR_DELAY_NORMAL (3)	~ 5 Hz

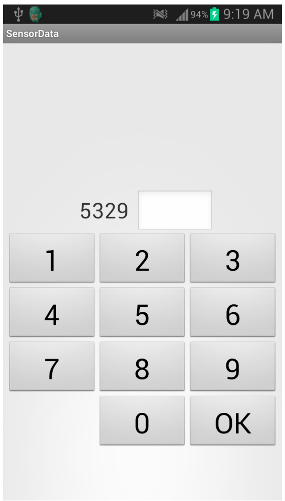

Figure 7.2: PIN input mask to gather test samples.

In order to determine the exploitability of the light-sensor information, we need to determine whether operating the smartphone leads to changes in the sensor readings. Therefore, we developed an Android application that prompts the user to enter a random 4-digit PIN. Figure 7.2 illustrates a screenshot of this application. While the user enters the PIN, the app collects the light sensor information. We collected this information for five consecutive PIN inputs and visualized the gathered information in Figure 7.3. For the sake of clarity we also plotted the single digits of the PIN and, as can be seen, a recurring pattern for the PIN (1-5-9-0) can be observed. These differences in the light intensity during the PIN input occur inevitably due to slight tilts and turns.

Figure 7.4 illustrates the tilts and turns leading to variations of the sensor values. For instance, assuming the light bulb being the main source of light, then tilting the device to the left causes a decrease of the reported *lux* value. Although we illustrate a point-like light source, the attack also works for environments that are uniformly lit via tube lights. Note that the PIN input mask can be aligned on the top or on the bottom of the screen. However, our observations showed that the alignment does not influence the information leakage.

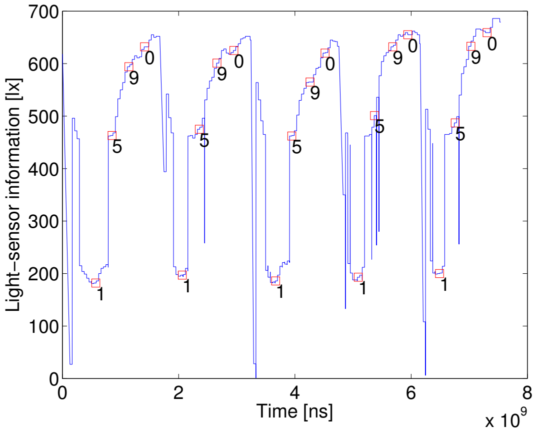

Figure 7.3: Light-sensor information for five consecutive PIN inputs (1-5-9-0).

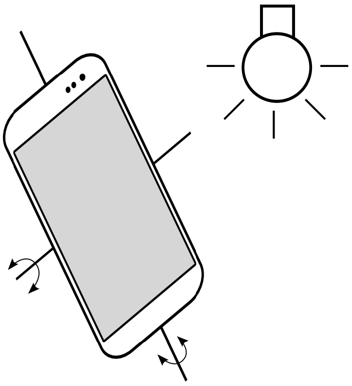

Figure 7.4: Schematic illustrating the tilts and turns of the smartphone.

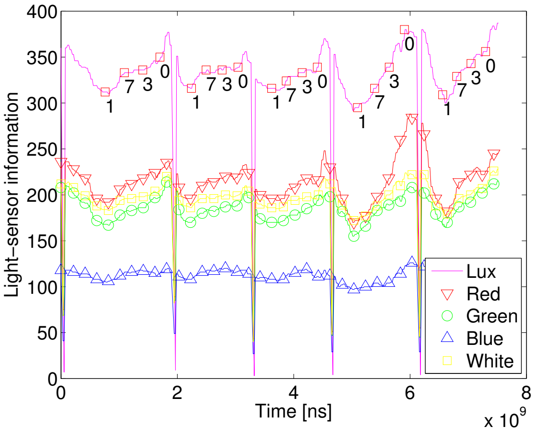

Figure 7.5: RGB(W)-sensor information for five consecutive PIN inputs (1-7-3-0).

7.3 RGB(W) Sensor

More recent smartphones, e.g., the Samsung Galaxy SIII [Samb] and the Samsung Galaxy S4 [Sama], also employ an RGB(W) sensor. This sensor reports the red, green, blue, and white (RGBW) intensities of the ambient light and is used to optimize the screen brightness and sharpness to prevent eye fatigue [Sam13].

The Android API, including Android 7.0, does not support RGB(W) sensors (yet), but sensor values can be read from the virtual file system directly. For instance, on the Samsung Galaxy SIII the RGBW values are publicly available via */sys/devices/virtual/sensors/light_sensor/raw_data*.

Figure 7.5 illustrates the data of the same experiment as mentioned before, but this time we also include the information provided by the RGB(W) sensor. Due to reasons of readability, we plotted only every 10-th value of the RGBW intensities. We observe that all values show a similar curve, but the intensity of the blue light seems to provide a smoother curve than the other four values. Our observations show that the RGBW information provides additional information that can be exploited, *i.e.*, additional features to be used for the machine-learning algorithm later on.

7.4 Attack Scenario

Since the light-sensor information reflects the ambient-light conditions, we consider a scenario where the training data for the classification algorithm is gathered in the same environmental setting as the data which is to be classified. A generic attack requires further investigations and is considered as future work.

7.4.1 Training Phase

A hand-crafted application is used to gather the light-sensor information during the user's interaction with the smartphone. The application tricks the user into operating the smartphone in a way that is similar to the input of multiple PINs. For example, a game where the user is supposed to solve mathematical puzzles could be used. In this case, the entered numbers can be interpreted as PINs.

We assume that users play such games in the living room while watching TV, or in a waiting room while waiting for an appointment. A study performed by the UK's Office of Communications [UK 13] coined the term *media stacking*, which refers to the fact that about half of UK's adults conduct their smartphone or tablet computer while watching TV. Thus, our assumption is reasonable.

As the game, as outlined above, might only be able to capture a limited number of training samples (PINs), the attacker might take the users' tendency to choose PINs into consideration. For instance, Bonneau et al. [BPA12] as well as Jakobsson and Liu [JL13] found that people tend to choose specific PINs like dates or PINs that represent common four-digit words, e.g., *"love"* (5683).

In case computing power for the machine-learning algorithm is required, the app transmits the data to a powerful server. Nevertheless, convincing the user that the application requires Internet access should be easy and since Android Marshmallow (6.0), the `INTERNET` permission is even granted automatically. Thus, the proposed attack can be performed without raising the user's suspicion.

7.4.2 Exploitation Phase

After gathering enough samples, the application tricks the user into restarting the device or starting the desired application, e.g., the banking application, to capture the light-sensor information during the input of the authentication PIN. If one considers to attack the smartphone's PIN, a service can be implemented to be started on boot time. Afterwards the sensor information captured during the game play is used to deduce the unknown PIN input by means of machine learning. The game might also trick the user into buying a specific add-on, a "new level", or a "new stage". When the user performs the in-app billing via *Google Wallet*, the game skims the corresponding authentication PIN.

Now one might question whether the revealed PIN is of any value for the attacker. In fact, if the attacker later gains physical access to the mobile device, she might gain access to the mobile device by unlocking the phone or might cause financial damage by authenticating herself to the corresponding application using the correct PIN. Furthermore, Aviv et al. [ASBS12] argued that the learned PIN might be valuable in case users reuse the PIN, for instance, as the ATM PIN. In addition, Simon and Anderson [SA13] predict that the number of smartphone applications requiring an authentication PIN will increase over time. Hence, users might be tempted to reuse one PIN across different applications, which exacerbates PIN-skimming vulnerabilities.

7.4.3 Security Implications

Simon and Anderson [SA13] state that sensor-based side-channel attacks are capable to overcome strong separation mechanisms like *Samsung KNOX* [Samc]. Samsung KNOX tries to separate the “private” world from the “business” world in order to provide better protection of corporate data on smartphones. However, based on the leaking sensor information, an unprivileged application running in the “private” world of the smartphone could learn sensitive information entered in the “business” world.

7.4.4 Observations and Assumptions

Considering a PIN-input mask as illustrated in Figure 7.2, and a user operating the smartphone with only one hand, using the thumb to enter the digits, we make the following observations. Left-handed persons tilt the smartphone slightly to the left side when entering PIN digits in the middle and right column of the key pad, *i.e.*, 2, 3, 5, 6, 8, 9. In contrast, right-handed persons tilt the smartphone slightly to the right side when entering PIN digits in the left and middle column, *i.e.*, 1, 2, 4, 5, 7, 8. This can be attributed to the fact that people slightly push the display towards the thumb. These tilts of the device cause the variations in the captured light-sensor information. Similarly, we expect fewer tilts and turns of the mobile device if it is held with one hand and operated with the index finger of the other hand or a stylus pen. Based on these observations we make the following assumptions in order for the outlined attack scenario to work.

Assumption 1: We assume that the targeted user is holding the mobile device in her hands rather than laying it onto a flat surface while operating it. If we would assume the mobile device is lying on a stable surface, *i.e.*, a table, the light-sensor information would not change during the operation of the device, unless the user’s hand causes the light-sensor changes.

Assumption 2: Furthermore, we assume that the PIN is entered on a key pad similar to the one illustrated in Figure 7.2 rather than a QWERTY keyboard with a single row of numbers. Examples of applications that are “protected” with an authentication PIN are, for instance, AppLock [DoM], Evernote [Eve], and KeepSafe [Kee], as well as mobile banking applications, e.g., Barclays [Bar], and NAB [NAB], just to name a few of them. Screenshots of these applications—provided by the corresponding developers—show that the authentication PIN is entered on a key pad as illustrated in Figure 7.2. Hence, this seems to be a rather fair assumption which does not have a negative impact on the attack scenario.

Assumption 3: We also assume that the user operates the mobile device in an environment where the light sensor faces a sufficiently large variance of the ambient light. This is not the case in completely dark environments. However, also rather gloomy environments, *i.e.*, a room in the late afternoon without any artificial light source (rather diffuse light), can be considered for potential attack scenarios, if the *lux* values vary slightly during the handling of the device.

7.5 Attack Approach

In this section, we detail the steps for the exploitation of the light-sensor information. As outlined in the scenario above, we perform a matching of sensor values captured during the input of an unknown PIN to the sensor values of known PINs. In terms of machine learning this represents a classification problem, where a so-called feature vector is mapped to a finite number of labels or categories, *i.e.*, PINs in our case. The required steps are as follows: (1) gathering the sensor values under known PINs as well as the sensor values under unknown PINs, and (2) employing machine-learning techniques to determine the unknown PINs based on the set of known PINs.

In order to perform the classification, sensor values and the corresponding PINs are used to train the machine-learning algorithm. This data is referred to as training data. The data that is to be classified is known as test data. We stick to the notation of Alpaydin [Alp10] and Bishop [Bis06], *i.e.*, bold letters denote vectors that are assumed to be column vectors by default, a superscript T denotes the transpose of a vector, and uppercase bold letters denote matrices.

7.5.1 Gathering the Required Data

The gathering of the required training data during game play can be formalized as follows. The malicious application captures a list of tuples (t, L, R, G, B, W), with t being the timestamp and L, R, G, B, and W representing scalars of the *lux* information, as well as the red, green, blue, and white intensities of the ambient light. Furthermore, we capture a list of tuples (t_p, d), with t_p being the timestamp of a pressed digit $d \in \{0, \ldots, 9\}$ of the p-th PIN, which will act as the ground truth. One PIN consists of four consecutive tuples in this list.

For each PIN we extract the sequence of sensor values (consisting of l tuples) within the time period defined by the timestamp of the first digit and the timestamp of the last digit of the PIN. In addition, one might consider a timeframe of a few milliseconds before and after the input of the first and the last digit of one PIN, which covers additional information. The resulting matrix $\mathbf{M}$ for one PIN (consisting of four digits) is as follows.

$$\mathbf{M} = \begin{bmatrix} (t, L, R, G, B, W)_1 \\ \vdots \\ (t, L, R, G, B, W)_l \end{bmatrix}$$

Each column in matrix $\mathbf{M}$ represents the sensor information during the input of one specific PIN, except the first one which represents the timestamp at which the information was captured. Before the gathered information can be exploited, we normalize the sensor values appropriately. This normalization of the data is done either by dividing each value (L, R, G, B, W) in one column by the norm of the corresponding column vector, or by rescaling the values via, e.g., $L_i = (L_i - \min(L)) / (\max(L) - \min(L))$. This data is then used to derive the feature vectors, which are briefly outlined within the following paragraphs.

Lux Values Only. The first set of feature vectors we consider are the exact *lux* values during the input of each specific digit of the PIN. We represent these values as a vector $\mathbf{x} = [L_1, L_2, L_3, L_4]^{\mathrm{T}}$, where the subscript refers to the corresponding digit of the entered PIN.

Lux Values Including RGBW Values. The second set of features are the exact *lux* values as well as the red, green, blue, and white (RGBW) intensities during the input of each digit for one specific PIN. We represented these values as a vector $\mathbf{x} = [L_1, R_1, G_1, B_1, W_1, \ldots, L_4, R_4, G_4, B_4, W_4]^{\mathrm{T}}$. In the following we refer to the feature vector comprised of these five features as $LRGBW$.

Polynomial of Degree 3. The third possibility we consider is fitting a polynomial of degree 3 through the second column of $\mathbf{M}$, *i.e.*, through all *lux* values during one PIN input. The coefficients of this polynomial $f(x) = ax^3 + bx^2 + cx + d$ are then considered as the features of a specific PIN, *i.e.*, $\mathbf{x} = [a, b, c, d]^{\mathrm{T}}$. The coefficients for the red, green, blue, and white intensities—the remaining columns of $\mathbf{M}$—are obtained in the same manner and appended to $\mathbf{x}$.

After gathering the data as outlined above, any set of the above outlined feature vectors is then combined into a matrix $\mathbf{F}_n$ with n rows, *i.e.*, one row for each PIN. Furthermore, a class vector $\mathbf{c}_n = [(d_1, d_2, d_3, d_4)_1, \ldots, (d_1, d_2, d_3, d_4)_n]$ of tuples corresponding to the PIN-digits (d_1, d_2, d_3, d_4) can be derived.

$$\mathbf{F}_n = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}, \ \mathbf{c}_n = \begin{bmatrix} (d_1, d_2, d_3, d_4)_1 \\ \vdots \\ (d_1, d_2, d_3, d_4)_n \end{bmatrix}$$

The matrix $\mathbf{F}_n$ containing the feature vectors as well as the label vector $\mathbf{c}_n$ are then used to train the classifier.

7.5.2 Detecting PIN Inputs

Capturing the training data allows to capture the timestamp of each digit input. However, during the input of the unknown PIN, the timestamp of single digit inputs is not known and hence we need a mechanism to determine the PIN input on a continuous sequence of sensor values. Therefore, we could use other sensors, e.g., the accelerometer, to reliably detect the input. Simon and Anderson [SA13] suggest to use the microphone to capture the vibrations of the haptic feedback to determine when a button has been pressed. Thus, we consider the detection of PIN inputs on a continuous sequence of sensor values as solved.

7.5.3 Determining the Unknown PIN

After we gathered the required light-sensor information for all the PINs ($\mathbf{F}_n$ and $\mathbf{c}_n$) to be used for the training phase of the machine-learning algorithm, we start the training phase. For this purpose, we use Matlab's Statistics Toolbox [Mat] with its extensive features and machine-learning algorithms. We perform the outlined attack by employing a *supervised learning* algorithm, which tries to learn a function and its parameters based on labeled training data. This function is later on used to determine the label of unseen data. More formally, given a set of tuples $(\mathbf{x}_i, c_i)$, with $\mathbf{x}_i \in \mathbb{R}^n$ being a feature vector and c_i the corresponding label of the observation, the algorithm tries to infer a function $f : X \to C$, where $X \in \mathbb{R}^n$ represents the feature vector of an observation and C the inferred label.

We investigate three classification algorithms—which have also been used in related work [ASBS12, MVBC12]—and compare their results afterwards. We briefly outline the chosen classification algorithms in the following paragraphs (cf. Alpaydin [Alp10] and Bishop [Bis06]).

Multiclass Logistic Regression. The classifier tries to learn parameters $\mathbf{w}_k$ for every class label $k \in C$, such that a vector $\mathbf{x}$ is assigned to label k in case $p(k|\mathbf{x}) > p(j|\mathbf{x})$ for all $j \neq k$, with p defined as below.

$$p(k|\mathbf{x}) = \frac{\exp\left(\mathbf{w}_k^{\mathrm{T}} \cdot \mathbf{x}\right)}{\sum_i \exp(\mathbf{w}_i{}^{\mathrm{T}} \cdot \mathbf{x})}$$

Discriminant Analysis. The classifier tries to learn parameters $\mathbf{w}_k$ for every class label $k \in C$, such that a vector $\mathbf{x}$ is assigned to label k in case $f_k(\mathbf{x}) > f_j(\mathbf{x})$ for all $j \neq k$, with f defined as below.

$$f_k(\mathbf{x}) = \mathbf{w}_k^{\mathrm{T}} \cdot \mathbf{x} + w_{k0}$$

When talking about *discriminant analysis*, we refer to *linear discriminant analysis* where classes are separated linearly.

K-Nearest Neighbor. The algorithm assigns the input vector $\mathbf{x}$ a label which is determined by the majority of the K nearest neighbors. We use $K = 5$ and the Euclidean distance to determine the distance between two vectors.

7.6 Evaluation and Results

We engaged ten users to evaluate this information leakage. Each user was asked to enter at least one set of random PINs with cardinality $k \in \{15, 30, 50\}$, each PIN for a specific number of times $N \in \{3, \ldots, 10\}$. If a PIN was entered incorrectly, we ignore the corresponding input and prompt the user to enter the PIN again. We follow the approach of Aviv et al. [ASBS12] who also measured the performance of their attack on a set of 50 PINs. In total we use the data of 29 test runs, *i.e.*, one test run consists of test data for a specific set of k PINs repeated N times by one user.

Evaluation Methodology. We apply k-fold cross validation to estimate the success rate of a classifier. Compared to the performance based solely on specific test data, cross validation provides more reliable estimations [Bis06].

Setup. We performed the experiments in office rooms that were uniformly illuminated via tube lights, in a living room with a standard ceiling lamp, and in rooms where the only light source was a window. We even considered different daytimes, e.g., during the day and in the late afternoon in order to test diffuse light conditions without a direct source of light rays. We asked the users not to walk around, which is compliant with our attack scenario. Furthermore, we did not insist on a specific input method, but only that the users hold the device during the PIN input. We observed the following input methods:

1. Holding the smartphone in one hand and entering the digits with the thumb of the same hand.
2. Holding the smartphone in one hand and entering the digits with the thumb of the other hand.
3. Holding the smartphone in one hand and entering the digits with the index finger of the other hand.

Subsequently, we analyze the gathered data with the above mentioned machine-learning algorithms and the proposed feature vectors to determine whether the secret PIN input can be recovered based on a set of known PINs.

7.6.1 Comparison of Classification Algorithms

First, we determine the overall classification rate based on the different classification mechanisms: (1) *multiclass logistic regression* (logistic regression), (2) *linear discriminant analysis* (discriminant analysis), and (3) *k-nearest neighbor classification* (KNN). These classifiers are fed with the feature vectors: (a) the *lux* values only (L), and (b) the *lux* values including the RGBW values (LRGBW).

We applied a 10-fold cross validation on all three classifiers and evaluated the performance of the suggested features for different numbers of samples (repetitions) per PIN. Figure 7.6 illustrates the average rate of correctly classified PINs out of a set of 15 known PINs for the different classifiers and the proposed feature vectors. The y-axis represents the average rate of correctly classified PINs, and the x-axis illustrates the number of gathered samples (repetitions) per PIN. We observe that the additional information leaked through the RGB(W) sensor leads to a better attack performance.

In addition, we also observe that the *linear discriminant analysis* provides better results than the other two classifiers, and that the average rate of correctly classified PINs increases with the number of samples per PIN. For instance, if we perform a *linear discriminant analysis* with a training set of 15 PINs, each repeated 8 times, then we are able to classify more than 80% of the PINs

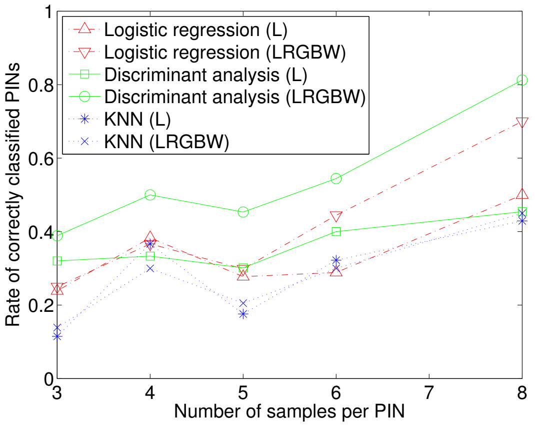

Figure 7.6: Average classification rate over multiple runs with a set of 15 PINs each.

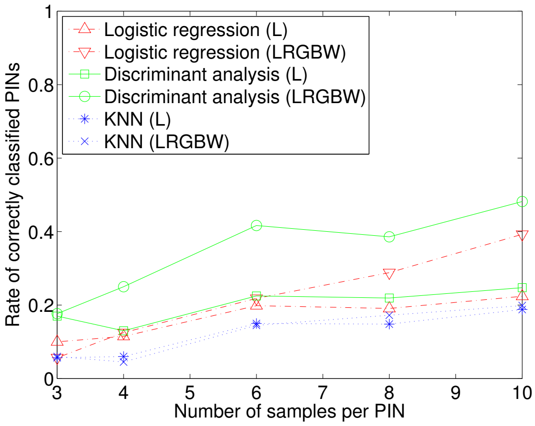

Figure 7.7: Average classification rate over multiple runs with a set of 50 PINs each.

correctly. In contrast, the chance of correctly guessing the right PIN from a set of 15 PINs randomly is $\frac{1}{15} = 6.7\%$.

Figure 7.7 illustrates the average rate of correctly classified PINs out of a set of 50 PINs. Again, the *linear discriminant analysis* outperforms the other two classifiers and the additional information from the RGB(W) sensor increases the performance compared to the *lux* value only. At first glance, an average rate of correctly classified PINs of 40 to 50% seems to be quite moderate. However, the chance of correctly guessing the right PIN from a set of 50 PINs is $\frac{1}{50} = 2\%$. Thus, our attack outperforms random guessing by a factor of 20 to 25.

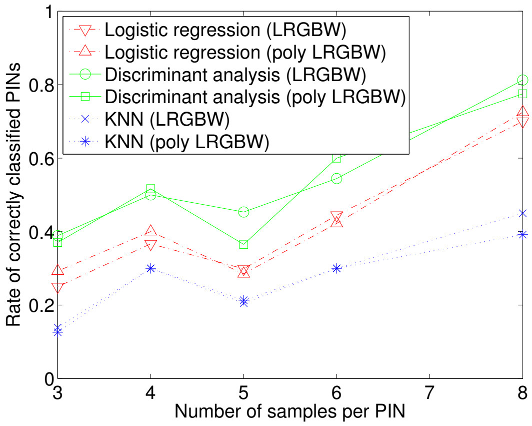

Figure 7.8: Average classification rate over multiple runs with a set of 15 PINs each.

7.6.2 Comparison of Feature Vectors

We now compare the feature vector comprised of the plain sensor values and the feature vector comprised of degree 3 polynomials fitted through all sensor values (e.g., lux and RGBW values) during one PIN input. Figure 7.8 illustrates the average rate of correctly classified PINs for these feature vectors based on a 10-fold cross validation over multiple runs with a set of 15 PINs. Both feature vectors, *i.e.*, the plain sensor values as well as the approximated sensor values, yield a similar performance for the different classification algorithms. Again, the *linear discriminant analysis* performs better than the other two classifiers. Based on this observation we only focus on the *linear discriminant analysis* in the following investigations.

7.6.3 Guessing PINs Based on Their Probability

An interesting approach is to consider the fact that an attacker can enter PINs for a specific number of times, *i.e.*, to guess possible PINs according to their probability for being the correct one. In this case the probability of finding the correct PIN increases with every tested PIN. Thus, we instruct the classifier to return a probabilistic ranking of the inferred PINs. Afterwards, we sort the potential PINs according to their probability for being the correct one and illustrate the rate of correctly classified PINs after a specific number of guesses.

Figure 7.9 shows the average rate of correctly guessing a PIN out of a set of 50 random PINs for a specific number of guesses. The additional information provided by the RGB(W) sensor yields better results and increases the success rate by about 10 percentage points compared to the *lux* value only feature vector. We also illustrate the success rate if one were trying to guess PINs randomly, which clearly shows the advantage of our attack compared to random guessing.

Comparing our results to the work of Aviv et al. [ASBS12], we observe that

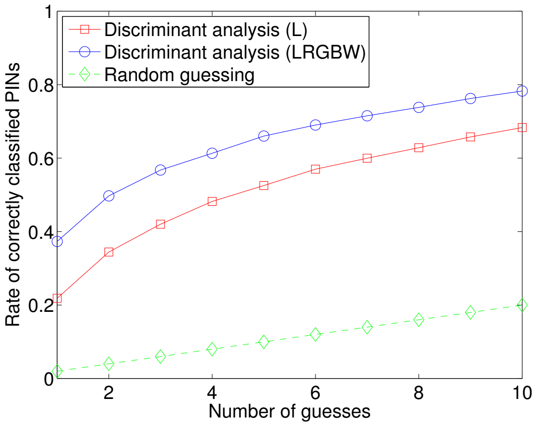

Figure 7.9: Average classification rate over multiple guesses on a set of 50 PINs.

the ambient-light sensor provides results at least as good as those achieved by exploiting the accelerometer sensor. Comparing our results to the work of Simon and Anderson [SA13], we observe that the ambient-light sensor provides even better results than the approach of exploiting the camera. For instance, Simon and Anderson claim to infer more than 30% of the PINs after two guesses and more than 50% of the PINs after five guesses. In contrast, the ambient-light sensor allows us to infer about 50% of the PINs after two guesses and about 65% of the PINs after five guesses.

Our results indicate that guessing PINs according to their probability for being the correct one provides an effective means of finding the correct one. On average we are able to infer the correct PIN with a probability of 80% when considering the ten most probable PINs. In contrast, guessing PINs randomly from a set of 50 PINs would result in a success rate of 20% after ten guesses.

7.6.4 Impact of Different Input Methods

We also investigate the impact of an input method on the classification rate. To this end, we compare the three input methods observed during our experiments:

1. Holding the device in one hand and using the thumb of the same hand to operate it.
2. Holding the device in one hand and operating it with the thumb of the other hand.
3. Holding the device in one hand and using the index finger of the other hand to operate it.

Figure 7.10 illustrates the average rate of correctly classified PINs for the three different input methods after guessing a specific number of the most probable PINs. The plot is based on a 10-fold cross validation considering a *linear*

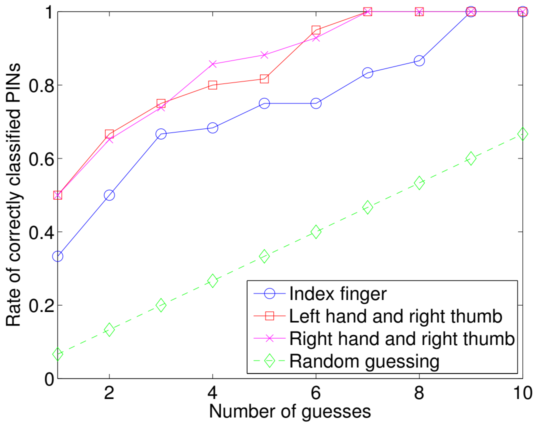

Figure 7.10: Average classification rate for different input methods on 15 PINs.

Table 7.2: Input methods of three users.

User	Input method
User 1	Left hand and index finger
User 2	Right hand and right thumb
User 3	Left hand and index finger

discriminant analysis on the LRGBW values. Each of the underlying sets of PINs had a cardinality of 15. According to this plot the two input methods involving the thumb, *i.e.*, left hand and right thumb as well as right hand and right thumb, seem to be more vulnerable to this attack than the one with the index finger. This is due to the fact that the mobile device usually undergoes only minor movements when the index finger is used, because one hand is solely used to hold the mobile device. However, this is not entirely correct because also for the input method involving the left hand and the right thumb one hand is solely used to hold the mobile device.

To gain further insight into factors affecting the rate of correctly classified PIN inputs, we compare the data of three different users. All three users entered 30 × 3 PINs, *i.e.*, each of the 30 PINs was entered 3 times, in the same room, and with the same environmental conditions regarding the ambient light. The corresponding input methods of these users are illustrated in Table 7.2.

Figure 7.11 illustrates the result of the 10-fold cross validation for the three data sets provided by *User 1*, *User 2*, and *User 3*, respectively. The input method of *User 2* seems to leak the most information. This appears to be due to the fact that she rested her upper arm against her upper body in a relaxed manner and operated the smartphone in a very comfortable way, which caused significant tilts and turns. *User 1* placed her elbows on her knees and also operated the device in a very relaxed way. In contrast, *User 3* tightly pressed

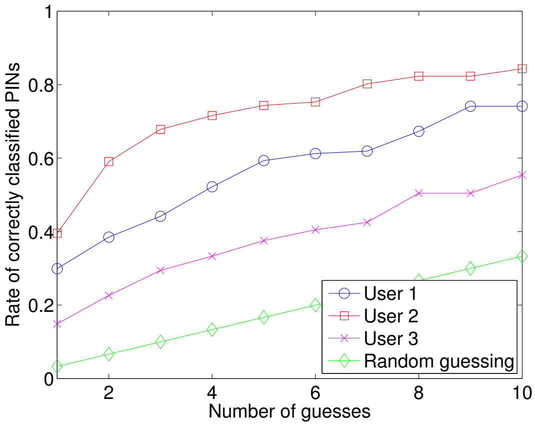

Figure 7.11: Average classification rate for three specific users on a set of 30 PINs.

her upper arm against the upper body and held the device very firmly in her hand while entering digits with the index finger.

Based on the investigations of these three users, we observe that though the input method seems to have an impact on the classification rate, a general statement regarding the security or insecurity of a specific input method is difficult to make. Nevertheless, the tighter and more firmly one holds the device, the less information is leaked. To put it more generally, the more movements the mobile device undergoes during the operation, the more information is leaked.

7.6.5 Impact of the Sampling Frequency

As outlined in Section 7.2, the ambient-light sensor can be configured to operate with different sampling frequencies. With a sampling frequency of 750 Hz, the Samsung Galaxy SIII provides an immense number of measurement samples per second, far more than what is necessary for a successful attack. Although we performed successful attacks with all possible sampling frequencies, most of our attacks were performed with sampling frequencies between 5 and 50 Hz.

As illustrates in Figure 7.12, the lowest sampling frequency supported by our device (5 Hz) is enough to perform the attack. In fact, the plot illustrates that the performance does not even decrease when sampling with the lowest frequency of 5 Hz. However, even lower sampling frequencies should prevent this attack as for sampling frequencies below 5 Hz too few measurement samples might be gathered if one enters the PIN too fast.

7.7 Limitations

The presented attack also has some limitations. First, our model does not consider mistyped PINs. If a user deletes an incorrect digit and enters the correct

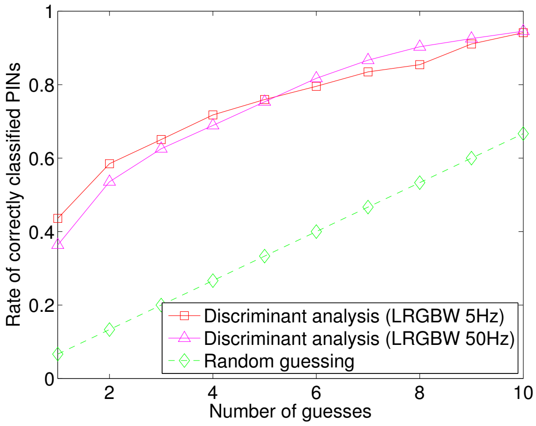

Figure 7.12: Average classification rate for different sampling rates on a set of 15 PINs.

digit afterwards, we are not able to infer the correct PIN anymore. Second, we did not evaluate our attack outside under sunny, foggy, or cloudy light conditions. However, we evaluated the attack in a room—where the only light source was a window—during different daytimes. Furthermore, the data used to train the classifier is currently gathered in the same environment as the data that should be classified. Future work, however, might investigate whether a more general model can be established in order to decouple the training phase from the attack phase. For instance, a "calibration" phase might be used to determine the overall light conditions in the user's environment. This might allow to reuse training data from different environments as in case of motion-based keyloggers and might reduce the overall effort of the training phase. Third, the presented attack is based on the ambient-light sensor and thus it does not work in case the user operates the mobile device in completely dark environments.

7.8 Countermeasures

In this section, we discuss potential mitigation techniques to prevent the exploitation of sensor information.

7.8.1 UI and API Modifications

Aviv et al. [ASBS12] argue that an effective security mechanism would be to prevent untrusted applications from accessing motion sensors, at least during the input of sensitive information. However, the crucial question is: *When is an input considered as being sensitive?* Clearly, the input of an authentication PIN or a password represents sensitive input, but also the input while writing an e-mail or the data entered in forms on websites can be considered as sensitive.

From this perspective, sensors should be disabled as soon as the virtual keyboard is displayed. This, however, renders applications that rely on these sensors useless as is also stated in [ASBS12]. Owusu et al. [OHD$^+$12] suggest to vary the keyboard layout by rearranging the buttons on the virtual keyboard prior to every sensitive input. The drawback of such countermeasures is the dramatic decrease of usability. While this might be applicable for a PIN pad, it would definitely undermine the usability of the QWERTY keyboard layout.

Reducing the sampling frequency to 1–2 Hz should suffice for the task of adapting the screen brightness and to mitigate the presented attack. Furthermore, since only the OS performs the task of adapting the screen brightness, access to this sensor might be restricted to the OS exclusively.

7.8.2 Rethinking the Permission Model

A quite sophisticated countermeasure might be a fine-grained permission system in mobile operating systems.[1] Felt et al. [FEF$^+$12] evaluated different permission-granting mechanisms, including automatic granting, trusted UIs (cf. [RKM$^+$12]), runtime consent dialogs, and install-time warnings. After considering their model we conclude that an effective countermeasure would be an install-time warning, *i.e.*, to pause the installation process and to explicitly inform or warn the user about the requested permission.

Specific risks that arise from permissions must be communicated to the user, especially since the manifold permissions confuse many users [FHE$^+$12, KCC$^+$12]. However, excessive warnings lose effectiveness and might cause users to ignore these warnings again. In order to overcome this problem, Peng et al. [PGS$^+$12] suggest to rank applications according to their risks. This ranking decision is based on the requested permissions of applications that are known to be malware. Based on such a ranking users might make more deliberate decisions regarding the installation of applications. However, such rankings are only applicable if the motion sensors as well as the ambient-light sensor are considered within the permission system of the OS.

7.8.3 Application Analysis

A similar approach might be achieved by extending *AppGuard* [BGH$^+$13]—an application to enforce security policies—to support the detection of possibly unwanted sensor accesses by malicious applications. *AppGuard* could scan applications during the installation and inform the user about sensor accesses that potentially leak sensitive information. Other malware-analysis applications such as static analyzers, e.g., *Stowaway* [FCH$^+$11] or *AndroidLeaks* [GCEC12], or applications like *VirusTotal* [Vir] could also be extended to check applications for malicious sensor accesses.

---

[1] Recently Google "simplified the permission system" [Goo14]. Now users are informed about rather coarse-grained permission groups during the installation of applications.

Table 7.3: Comparison of related work targeting a set of 50 PINs.

	Aviv et al. [ASBS12]	Simon and Anderson [SA13]	Ours
Sensor	Accelerometer	Camera	Ambient-light sensor
Permissions	Internet	Camera, Internet	Internet
Training	Independent of user/location	For each user individually	For each user/environment individually
Drawbacks	-	LED and shutter sound on non-rooted devices	Does not work in completely dark environments
Input method	No constraints	Thumb of holding hand	No constraints
Accuracy	43% within 5 guesses	50% within 5 guesses	65% within 5 guesses

7.8.4 User Behavior

Another countermeasure might be to enter sensitive data in environments without any light source and by using the index finger or a stylus pen. However, in this case other sensors, e.g., motion sensors, might still be exploitable. So, for really sensitive data, the user might cover the ambient-light sensor as well as the camera, e.g., with her finger, and place the mobile device on a flat surface.

Encouraging users to choose longer PINs and passwords might also increase the security [SA13]. However, some applications do not even allow PINs with more than 4–5 digits. Last but not least, awareness must be raised amongst users and users must be encouraged to be wary when installing applications.

7.9 Comparison With Related Work

Table 7.3 provides a comprehensive comparison of related work that is similar to ours, *i.e.*, attacks targeting a specific set of PINs. The main drawback of our attack is that users are not allowed to walk around while entering the PINs because currently the data is only exploitable for one specific environment. Hence, the data cannot be reused for multiple attacks as in case of the accelerometer sensor (cf. [ASBS12]). Nevertheless, their results indicate a rather low success rate of 20% within 5 guesses when inferring PINs that were entered while walking around. Furthermore, their attack also works in completely dark environments which does not hold for the ambient-light sensor. However, our results indicate a slightly better accuracy of 65% within 5 guesses when inferring unknown PINs.

The work of Simon and Anderson [SA13] requires the `CAMERA` permission which potentially gains the user's suspicion and their attack must deal with the audio-visual feedback, e.g., the shutter sound or the LED, while capturing the required data. Compared to their work we do not restrict our study to specific input methods as long as the user holds the device while operating it.

Furthermore, they need to transfer image data to the server, which cannot be represented as compact as sensor values. While this first investigation showed that the ambient-light sensor might not be superior to existing sensor-based attacks, we have shown that this sensor leaks sensitive information that can be exploited effectively. Besides, we are the first to show that an ambient sensor can also be exploited to infer user input.

7.10 Conclusion and Future Work

In this chapter, we investigated a new type of side channel which is based on the ambient-light sensor. We developed a proof-of-concept application that allows us to infer PIN inputs, which clearly demonstrates that the leaked information represents a viable side channel for compromising a user's privacy and security. Since no specific permission is required to access this sensor, an adversary is able collect sensitive inputs from mobile-device owners without raising any suspicion.

There are many different dimensions among the information leakage of sensors can be investigated. Examples of factors affecting the applicability and performance of sensor-based side-channel attacks are, for example, the sensor hardware itself, the screen dimension, the device orientation, the keyboard layout, the user's behavior and typing style, the different classification algorithms and the employed feature vectors, the environment (e.g., indoor and outdoor), etc. These examples demonstrate that further research is necessary to evaluate the performance of available sensors under different settings in order to determine the best sensor for a specific attack scenario. However, the intention of this work was to provide a first feasibility study. By comparing three classification algorithms, different feature vectors, different input methods, different environments, and the impact of different sampling frequencies, we showed that the ambient-light sensor provides a viable side channel.

Related work on the leakage of motion sensors claimed that access to these sensors must be limited with a fine-grained permission system. As shown in this work, access to the ambient-light sensor must also be protected through such a permission system and, thus, operating-system developers need to deal with this problem. Probably even more important is the fact that users need to be aware of such threats and to be wary when installing applications.

8

Exploiting the Data-Usage Statistics

The amount of control you have over somebody
if you can monitor Internet activity is amazing.
— Tim Berners-Lee

The browsing behavior of a user allows to infer personal details, such as political interests, sexual orientation, etc. In order to protect this sensitive information, defense mechanisms like SSH tunnels and anonymity networks (e.g., Tor) have been established. A known shortcoming of these defenses is that website fingerprinting attacks allow to infer a user's browsing behavior based on traffic analysis techniques. However, website fingerprinting typically assumes access to the client's network. We propose a client-side attack that overcomes several limitations and assumptions of network-based fingerprinting attacks, e.g., network conditions and traffic noise, expensive training phases, etc. Thereby, we present a practical attack that can be implemented easily and deployed on a large scale. In fact, we show that an unprivileged application can infer the browsing behavior by exploiting the unprotected access to the Android data-usage statistics. We are able to infer 97% of 2 500 page visits out of a set of 500 monitored pages correctly. Even if the traffic is routed through Tor, we can infer 95% of 500 page visits out of a set of 100 monitored pages correctly. Thus, our attack bypasses the `READ_HISTORY_BOOKMARKS` permission.

We start with an introduction in Section 8.1 and discuss the concept of website fingerprinting in Section 8.2. We investigate the data-usage statistics in Section 8.3, propose possible attack scenarios in Section 8.4, and describe our attack in Section 8.5. We evaluate our attack in Section 8.6 and discuss countermeasures in Section 8.7. Parts of this chapter are taken verbatim from [SGKM16].

Publication Data and Contribution

Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 49–60. ACM, 2016
Contribution: Main author; Initial prototype implemented by Simone Griesmayr.

8.1 Introduction

Android tracks the amount of incoming and outgoing network traffic on a per-application basis. This information is used by data-usage monitoring applications to inform users about the traffic consumption. However, while this feature might be helpful to stick to one's data plan and to identify excessive data consumptions of applications, we show that this seemingly innocuous information allows to infer a user's visited websites. We demonstrate that the `READ_HISTORY_BOOKMARKS` permission, which is intended to protect this sensitive information, is actually useless as any application without any permission is able to infer visited websites rather accurately.

The exploitation of observed traffic information to infer visited websites is known as website fingerprinting [Hin02, SSW$^+$02]. Thereby, an attacker aims to match observed traffic information to a previously established mapping of websites and their corresponding traffic information. Most of these investigations consider an attacker who sniffs the traffic information "on the wire". This means that the attacker needs to be located on the client's network or on the ISP's router near the client. However, as Android allows an attacker to capture the required data directly on the smartphone without any permission, we show that an attacker is not required to be located somewhere on the victim's network. Hence, the rather strong assumption of a network-based attacker is not required for website fingerprinting attacks. Furthermore, our attack is invariant to traffic noise of other applications—one of the major drawbacks of network-based attacks—as Android captures these statistics on a per-application basis. Compared to existing website fingerprinting attacks, we significantly reduce the computational complexity of classifying websites as we do not require a dedicated training phase, which sometimes requires several hundred CPU hours [WG13, WCN$^+$14].

Based on our observations, we developed a proof-of-concept application that captures the data-usage statistics of the browser application. With the acquired information in a closed-world setting of 500 monitored websites of interest, we are able to classify 97% of 2 500 visits to these pages correctly. The fact that not even Tor on Android (e.g., the *Orbot*[1] proxy in combination with the *Orweb*[2] browser) is able to protect a user's page visits clearly demonstrates the rather

---

[1] `https://guardianproject.info/apps/orbot`

[2] `https://guardianproject.info/apps/orweb`

delicate issue of this fundamental design flaw.

Further Security Implications. The presented attack can be combined with related studies for even more sophisticated attack scenarios. For instance, in combination with sensor-based keyloggers (cf. Chapter 7), our attack would allow an adversary to determine when a user visits a specific website and to gather login credentials for specific websites. Such an attack does not only endanger the users' privacy but also allows for large-scale identity theft attacks.

8.2 Background and Related Work

Website fingerprinting can be considered as a supervised machine-learning problem, namely a classification problem. The idea is to capture the "traffic signature" for specific websites—which are known to the attacker—during a training phase. The "traffic signature" consists of specifically chosen features like unique packet lengths, packet length frequencies, packet ordering, inter-packet timings, etc. In order to capture this information, the attacker loads different websites and observes the resulting "traffic signature", which is usually done somewhere on the network. During the attack phase, an observed "traffic signature" can be classified according to the previously trained classifier.

Most related work in the context of website fingerprinting attacks operate in the closed-world setting, which, in contrast to the open-world setting, assumes that the victim only visits a specific set of monitored websites. Furthermore, most fingerprinting attacks assume a passive attacker, where the attacker cannot influence the transmitted packets, e.g., by delaying specific packets [HYG+14].

Subsequently, we summarize related work according to the exploited information and how this information is gathered. We start with attacks that require access to the victim's network or to the ISP's router near the victim. Then, we continue with attacks that exploit shared resources on the victim's device, which is the category of attacks our work belongs to.

8.2.1 On the Wire

Hintz [Hin02] mentioned that encrypted traffic does not prevent an adversary from inferring a user's visited website. The simple observation of the amount of transferred data—which is not protected by means of SSL/TLS (cf. [MHJT14])—allows an adversary to infer visited websites. Similarly, Sun et al. [SSW+02] mentioned that the observation of the total number of objects and their corresponding lengths allows to identify websites, even if the content is encrypted.

While early studies [Hin02, SSW+02] exploited the actual size of web objects, a more recent study by Liberatore and Levine [LL06] focused on the exploitation of individual packet sizes. Such fingerprinting attacks have also been demonstrated to work against the anonymity network Tor [PNZE11, CZJJ12, WG13] and also against WEP/WPA encrypted communication as well as communication protected by means of IPsec and SSH tunnels [BLJL05, LCC10]. Instead of inferring visited websites, Chen et al. [CWWZ10] extracted illnesses and medications by observing the sequence of packet sizes.

Gong et al. [GKB10, GBKS12] even demonstrated that fingerprinting can be done remotely when given the victim's IP address. Therefore, the attacker sends ping requests to the user's router and computes the round-trip time, which correlates with the victim's HTTP traffic.

8.2.2 Shared Resources and Software Execution

Timing attacks on the browser cache [FS00, LYL+14] can be used to infer whether or not a user visited a specific website before. More specifically, by measuring the loading time of a specific resource, an attacker can determine whether it was served from the browser's cache or not. Similarly, CSS styles of visited URLs can be used to determine the browsing behavior [JO10].

Another timing attack has been demonstrated by Oren et al. [OKSK15], who showed that JavaScript-based cache attacks (cf. Section 5.6) allow to infer page visits to a set of 8 websites. Similarly, Gruss et al. [GBM15] exploited so-called page-deduplication attacks to infer page visits to a set of 10 websites.

Jana and Shmatikov [JS12] demonstrated the possibility to fingerprint websites based on the browser's memory footprint, which is available via the `/proc` file system. In addition, Zhou et al. [ZDH+13] demonstrated that the data-usage statistics of Android applications can be used to infer the user's activities within three Android applications, namely *Twitter*, *WebMD*, and *Yahoo! Finance*. Later, Zhang et al. [ZYN+15] exploited the data-usage statistics of an Android-based Wi-Fi camera to determine when a user's home is empty.

Even though Zhou et al. [ZDH+13] and Zhang et al. [ZYN+15] started investigations of the information leakage through the data-usage statistics on Android devices for specific applications, a detailed study regarding the applicability of this information leakage to infer websites has not been done yet. Compared to the work of Jana and Shmatikov [JS12] who exploit the memory footprint of the browser application for website fingerprinting attacks, we demonstrate a significantly more accurate attack by exploiting the data-usage statistics.

8.3 Android Data-Usage Statistics

Android keeps track of the data usage in order to allow users to stick to their data plan. This accounting information is available through the public API as well as through the `/proc` file system. More specifically, the `TrafficStats` API as well as `/proc/uid_stat/[UID]/{tcp_rcv|tcp_snd}` provide detailed information about the network traffic statistics on a per-UID basis. Since every Android application is assigned a unique UID, these traffic statistics are gathered on a per-application level. In order to observe the data-usage statistics of an application, e.g., the browser, the corresponding UID is required. The `ActivityManager` API can be used to retrieve the UID for all running processes.

Subsequently, we study the information leakage for browser applications in a standard setting and in case the traffic is routed through the Tor network. Furthermore, we also investigate the information leakage depending on the network connection. Finally, we discuss the API support for the data-usage statistics as well as a mechanism to circumvent the `REAL_GET_TASKS` permission, which is required on Android Lollipop to retrieve the list of running applications.

8.3.1 Usage Statistics in a Controlled Scenario

In a first experiment, we set up a server hosting a website and we launched `tcpdump` to dump the TCP packets on this server. In addition, we launched the browser application on one of our test devices (a Samsung Galaxy SIII), retrieved its UID, and navigated to the website hosted on our server and monitored `tcp_snd` and `tcp_rcv` for ten seconds with a sampling frequency of 50 Hz.

Figure 8.1 illustrates the accumulated TCP packet lengths (left) and the data-usage statistics on the Android smartphone (right). We indicate the outgoing traffic on our server as well as the incoming traffic on the smartphone above the x-axis. Similarly, we indicate the incoming traffic on our server as well as the outgoing traffic on the smartphone below the x-axis. For the sake of readability, we removed consecutive samples where the `tcp_rcv` and `tcp_snd` values did not change. Furthermore, we labeled each TCP packet with the corresponding packet length in both plots. The left plot shows the generated TCP packets according to our website. The first three outgoing packets (1448, 1448, 1417) correspond to the HTML page itself and the following packets (1448, 1448, 1448, 71) correspond to the embedded image. Interestingly, the data-usage statistics on the Android smartphone (right) corresponds to these TCP packet lengths, except for the last two packets (1448, 71), which are observed as one “large” packet (1519=1448+71) instead of two separate packets. The same observation also holds for the incoming traffic on the server and the outgoing traffic on the smartphone, which are indicated below the x-axis. The corresponding TCP packet lengths can be observed in the outgoing data-usage statistics (366, 364).

The plots in Figure 8.1 illustrate the TCP packet lengths when loading the website for the first time, *i.e.*, without any data being cached. When visiting the website for the second time, the traffic signature slightly changes. More specifically, the second part of the packet sequence (1448, 1448, 1448, 71) is missing as the embedded image is not requested anymore. However, some packet lengths remain the same, regardless of whether the website is cached or not.

Sampling Frequency. Zhou et al. [ZDH^{+}13] reported to be able to observe single TCP packet lengths with a sampling frequency of 10 Hz most of the time. We performed experiments with higher sampling frequencies but also observed the aggregation of multiple TCP packet lengths as one “larger” packet from time to time. A more detailed investigation of specific browser implementations—which is considered as future work—might reveal further insights and might allow to pick up every single TCP packet properly. Nevertheless, even with some TCP packet lengths being accumulated into one observation, we can successfully exploit this side channel with a sampling frequency between 20 Hz and 50 Hz.

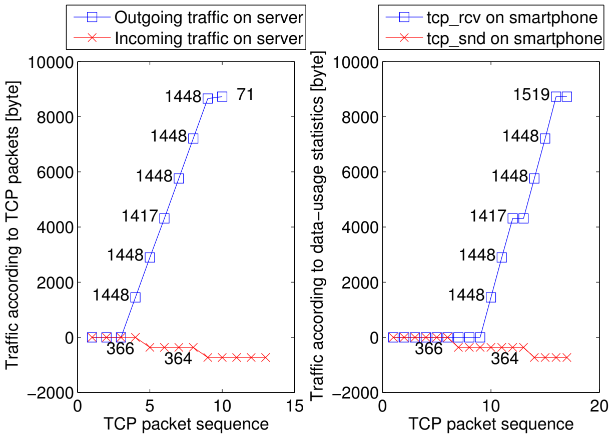

Figure 8.1: TCP packet lengths according to `tcpdump` on the server and data-usage statistics on the smartphone.

8.3.2 Usage Statistics for Real Websites

In order to investigate the information leakage for real websites, we developed an application that performs the following actions. First, we launch the browser and retrieve its UID. Then, we load three different websites (`google.com`, `facebook.com`, and `youtube.com`) and monitor `tcp_snd` and `tcp_rcv` for a period of ten seconds. The resulting plots can be seen in Figure 8.2. According to the notion of Jana and Shmatikov [JS12], these measurements are *stable*, meaning that these observations are similar across visits to the same page, and also *diverse*, meaning that these observations are dissimilar for visits to different pages. Hence, this plot confirms our previous observation that the data-usage statistics can be used to distinguish websites. A similar plot can be obtained from the `tcp_snd` file, *i.e.*, the outbound network traffic.

8.3.3 Usage Statistics in the Tor Setting

Background on Tor. Before we investigate the information leakage of the data-usage statistics in case the network traffic is routed through the Tor network, we start with some background information on Tor. The major design goal of Tor [DMS04] is "to frustrate attackers from linking communication partners" by considering an attacker who can, for instance, observe the network traffic. Therefore, a user runs a so-called onion proxy that is responsible for handling connections from user applications (e.g., the browser), fetching directories (e.g., known onion routers), and establishing circuits (paths) through the network. Such circuits consist of multiple onion routers—which are connected by means of a TLS connection—and are updated periodically. However, establishing such circuits is a costly action that takes some time, which is why multiple TCP streams share one circuit. The onion proxy accepts TCP streams from user ap-

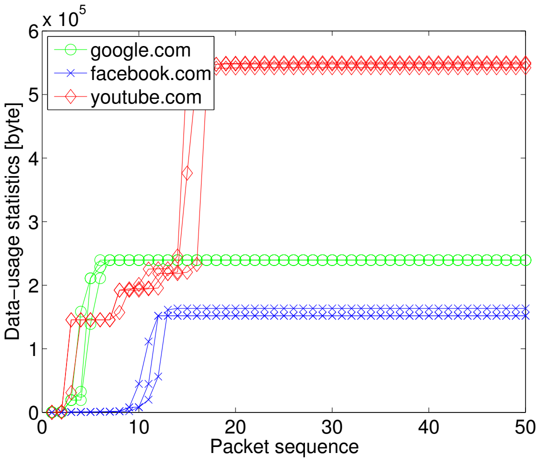

Figure 8.2: Data-usage statistics for the inbound traffic of three samples per website.

plications (browsers) and forwards the data in fixed-size cells (512 bytes) through the TLS connection to the Tor network.

Information Leakage. In order to investigate the information leakage for traffic routed through the Tor network, we installed the Orweb browser and the corresponding Orbot proxy. The Orweb browser represents the user application and the Orbot proxy represents the onion proxy that handles connections from the Orweb browser and forwards the data to the Tor network. Since websites take longer to load, we increased the time for sampling the data-usage statistics to 20 seconds. As Tor on Android relies on two different applications, *i.e.*, the Orweb browser and the Orbot proxy, we investigated the information leakage for both applications. While the Orweb browser communicates with the Orbot proxy only, the Orbot proxy communicates with the browser as well as the Tor network. Thus, the data-usage statistics for the Orbot proxy are slightly higher. However, both applications revealed the exact same behavior, *i.e.*, the data-usage statistics yield stable and diverse plots, which can be exploited for website fingerprinting attacks. We also installed Firefox 42.0 and configured it to use the Orbot proxy. Repeating the experiment yields the same result, *i.e.*, the data-usage statistics gathered for Firefox allow us to perform website-fingerprinting attacks even if Firefox is configured to route the network traffic through the Tor network. We stress that this is not a vulnerability of Tor or any browser but a fundamental weakness of the Android OS.

8.3.4 Usage Statistics for Mobile Connections

The above performed experiments have been carried out with WLAN connections. For the sake of completeness, we also performed experiments with mobile data connections to be sure that we observe the same information leakage when

Table 8.1: Test devices and configurations.

Device	OS	Browser/Orbot
Acer Iconia A510	Android 4.1.2	Chrome 44.0
Alcatel One Touch Pop 2	Android 4.4.4	Browser 4.4.4 (default browser)
Nexus 9	Android 5.1.1	Chrome 40.0
Samsung Galaxy SII	Android 2.3.4	Internet 2.3.4 (default browser)
Samsung Galaxy SII	Android 2.3.4	Orweb 0.7 and Orbot 13.0.4a
Samsung Galaxy SII	Android 2.3.4	Firefox 42.0 and Orbot 13.0.4a
Samsung Galaxy SIII	Android 4.3	Internet 4.3 (default browser)

the device is connected, e.g., via the 3G wireless network. The results confirmed our initial observations regarding the data-usage statistics also for mobile connections. Irrespective of the actual Internet connection, the data-usage statistics can be exploited for website fingerprinting attacks.

8.3.5 API and /proc Support

Table 8.1 summarizes our test devices and their corresponding configurations. On most of these devices, we accessed the corresponding files within the `/proc` file system to retrieve the data-usage statistics. However, on the Alcatel One Touch Pop 2, the `uid_stat` file does not exist within the `/proc` file system, yet the `TrafficStats` API allows to retrieve the accumulated bytes received (`getUidRxBytes([uid])`) and transmitted (`getUidTxBytes([uid])`) for a given UID. Similarly, on the Samsung Galaxy SIII, we always retrieved 0 when querying the `TrafficStats` API, but still we were able to read the data-usage statistics from the `/proc` file system. To summarize our investigations, on some devices an attacker needs to rely on the `/proc` file system, while on others the attacker needs to rely on the `TrafficStats` API. However, all test devices showed the same information leakage through the data-usage statistics.

REAL_GET_TASKS Permission in Lollipop. Since Android Lollipop (5.0), the `REAL_GET_TASKS` permission is required to retrieve all running applications via the `ActivityManager.getRunningAppProcesses()` API. However, one can bypass this permission by retrieving a list of installed applications via `PackageManager.getInstalledApplications()`. The returned information also contains the UID for each application. Now, instead of relying on `getRunningAppProcesses()`, the malicious application can also wait for the `tcp_rcv` file to be created, which indicates that the application with the corresponding UID has been started. Another alternative to retrieve all running applications is the unprivileged `ps` command. Thus, even on Android Lollipop, our malicious service can be implemented without any suspicious permission and is still able to wait in the background for the browser application to start.

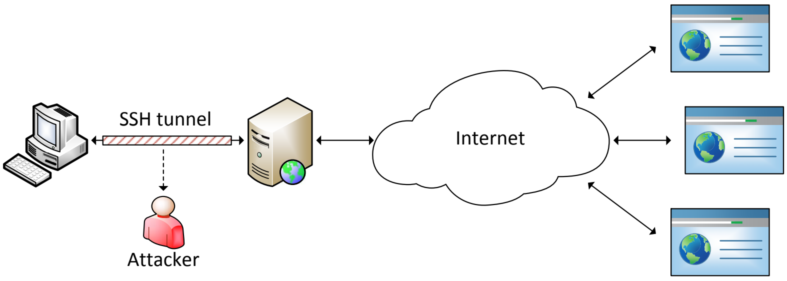

Figure 8.3: Traditional website fingerprinting attack considering a network attacker.

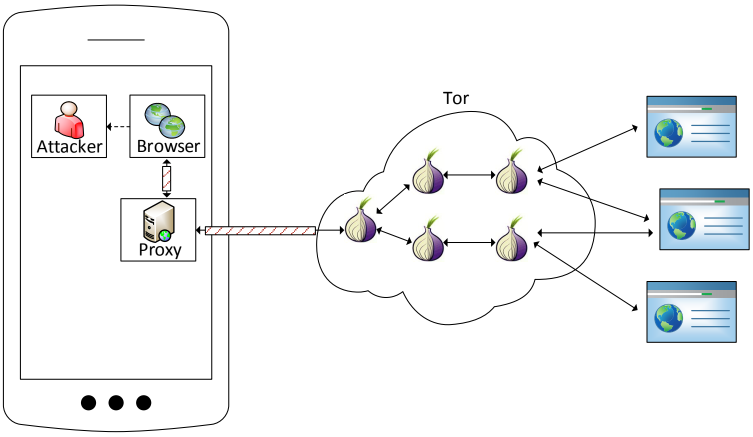

Figure 8.4: Client-side website fingerprinting attack exploiting a local side channel.

8.4 Adversary Model and Scenario

Figure 8.3 illustrates a traditional website fingerprinting attack, where the adversary observes the encrypted communication between a client and a proxy (or the encrypted communication between a client and the Tor network). In contrast, we consider a malicious application running in unprivileged mode that observes the incoming and outgoing traffic statistics for the targeted application, e.g., the browser. Figure 8.4 illustrates this scenario.

According to the notion of Diaz et al. [DSCP02], our attacker is passive as it cannot add, drop, or change packets. However, this also means that our attacker is lightweight in terms of resource usage as it runs in the background and waits for the browser application to start. Below we describe two possible attack scenarios, one where the training data is gathered on dedicated devices and another one where the attack application gathers the training data directly on the device under attack. Note that the `INTERNET` permission is not required at all due to the following reasons. Since Android Marshmallow (6.0), the `INTERNET` permission is granted automatically[3] and below Android 6.0, `ACTION_VIEW`[4] Intents can be used to access the Internet via the browser without this permission.

Since the application neither requires any suspicious permission nor exploits specific vulnerabilities except accesses to publicly readable files and the Android API, the application can be spread easily via available app markets. Based on the presented adversary model and the low effort to spread such a malicious application, there is a significantly higher attack potential than in previous fingerprinting attacks.

8.4.1 Possible Attack Scenarios

In order to exploit the information leakage, we consider two attack scenarios.

Scenario 1. For this scenario we assume that the malicious application does not capture the required training data on the device itself. Instead, a more powerful adversary gathers the training data on specifically deployed devices. The application only waits for the browser to start, gathers the traffic information, and sends the gathered data to the remote server that infers the visited websites. In order to match the device name of the attacked device with the training devices, the `android.os.Build` API can be used.

Scenario 2. For this scenario we assume that the malicious application captures the required training data directly on the device. Therefore, it triggers the browser to load a list of websites, one after each other, via the `ACTION_VIEW` Intent. While the browser loads the website, the malicious application captures the traffic statistics via `tcp_rcv` and `tcp_snd`, which then acts as training data. After collecting the required training data, the application waits in the background until the unsuspecting user opens the browser and starts browsing the web. Then, the application gathers the traffic statistics from `tcp_rcv` and `tcp_snd` again, and matches the collected information against the previously established training data to infer the visited websites.

A technicality that needs to be solved in case of scenario 2 is that users should not notice the gathering of training data. Therefore, we note that Zhou et al. [ZDH+13] demonstrated the possibility to (1) wait for the screen to dim before launching the browser, and (2) to close the browser after loading a website. Thereby, the user does not observe any suspicious behavior, even though the application launches the browser in the foreground. However, the main drawback of this approach is that the device might switch to sleep mode and pause all activities, which means that gathering the training data takes some time.

8.4.2 Assumptions

According to Wang et al. [WCN+14], existing fingerprinting attacks rely on two assumptions. We briefly summarize these assumptions and argue why our attack approach is more realistic than existing fingerprinting attacks.

---

[3] `http://developer.android.com/guide/topics/security/normal-permissions.html`

[4] `http://www.leviathansecurity.com/blog/zero-permission-android-applications`

1. *The attacker knows the start and the end of a packet trace.* This assumption is based on the observation that users take some time to load the next webpage. We justify this assumption by arguing that we are able to determine when the browser starts. Thus, we are able to observe the trace of the first webpage. Afterwards, we assume that the user takes some time to load the next page.

2. *The client does not perform any other activity that can be confused with page-loading activities, for example, a file download.* Hayes and Danezis [HD15] pointed out that it is highly unlikely that an attacker will be able to gather traffic information without background traffic, which limits the applicability of existing website fingerprinting attacks. However, Android captures the data-usage statistics on a per-application basis and, thus, our approach is invariant to network activities of other applications. For instance, our attack also works in case an e-Mail app, WhatsApp, or any other app fetches updates in the background while the user browses the web. In contrast, it is highly unlikely that the network traffic on the wire does not contain any background traffic.

Another thing that needs to be clarified is the browser's caching behavior. For our experiments, we do not clean the cache before loading a page as we assume that adversaries might be interested in identifying frequently visited websites of a user. If users frequently visit specific websites, then these sites are most probably already cached. Still, specific parts of the TCP packets are equal between cached and non-cached pages, as has been discussed in Section 8.3.1. Our experiments with the Orweb browser use the default settings, meaning that caching of websites is disabled. Thus, we provide insights for both settings of the caching behavior.

8.5 Attack Description

Based on the presented adversary model and attack scenarios, we now describe the attack in more detail. First, we discuss how to gather the required traffic statistics for a set of monitored websites. Afterwards, we describe the employed classifier to infer the visited websites.

8.5.1 Gathering Traffic Signatures

The list of monitored websites might, for example, be chosen according to specific interests like political interests, sexual orientation, illnesses, or websites that are supposed to be blocked. For our evaluation, we decided to use popular websites among different categories according to `alexa.com`. The fact that Tor browsers, e.g., Orweb, do not cache pages, leads to the realistic scenario that an adversary wants to monitor landing pages (cf. [HD15]). Thus, we consider our chosen setting for the evaluation as being realistic.

Algorithm 1 summarizes the steps to establish the required training data denoted as traffic signature database T. The algorithm is given a list of monitored websites W, the desired number of samples per website n, a profiling time τ, and a sampling frequency f. For each website $w_i \in W$, the algorithm loads this website within the browser. While the browser application loads the website w_i, we gather the data-usage statistics f times per second for a period of τ seconds. Each tuple (w_i, t_i), which is denoted as one *sample* for a specific website w_i, is added to T. These steps are repeated until n samples have been gathered for each website. Finally, the algorithm returns the traffic signature database T.

Algorithm 1: Gathering training samples.

```
Input: List of monitored websites $W$, number of samples per website $n$,
       profiling time $\tau$, sampling frequency $f$
Output: Traffic signature database $T$

Launch browser application and retrieve its UID
repeat $n$ times
    foreach website $w_i$ in $W$ do
        simultaneously
        Launch website $w_i$ in browser
        while profiling time $\tau$ not passed do
            $f$ times per second
            read tcp_rcv and append to $t_{IN}$
            read tcp_snd and append to $t_{OUT}$
        end
        $t_i \leftarrow \{t_{IN}, t_{OUT}\}$
        Add tuple $(w_i, t_i)$ to $T$
    end
end
```

8.5.2 Classification

The traffic signature database T requires only minor pre-processing before the classification. More specifically, we removed samples of websites that did not load, *i.e.*, we removed tuples (w_i, t_i) from T where all entries in t_i are 0. Furthermore, if $n - 1$ samples for a specific website are removed, we remove the remaining sample as well. We justify this as follows. If this single remaining sample of a specific website is used for training, then it cannot be used for evaluation purposes. Similarly, if we do not have a single sample for training, then this site will never be classified correctly.

We use the Jaccard index as a metric to determine the similarity between two websites. In case of our measurement samples, the Jaccard index as defined in Equation 8.1 compares two traces t_1 and t_2 based on unique and distinguishable packet lengths.

$$\text{Jaccard}(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} \tag{8.1}$$

Our classifier aims to find the maximum similarity for a given trace $t_A = \{t_{IN_A}, t_{OUT_A}\}$ compared to all traces $t_i = \{t_{IN_i}, t_{OUT_i}\}$ within the previously established traffic signature database T. We illustrate this similarity measure in Equation 8.2.

$$\begin{aligned} &\text{Sim}(t_A = \{t_{IN_A}, t_{OUT_A}\}, t_i = \{t_{IN_i}, t_{OUT_i}\}) = \\ &\quad \text{Jaccard}(t_{IN_A}, t_{IN_i}) + \text{Jaccard}(t_{OUT_A}, t_{OUT_i}) \end{aligned} \tag{8.2}$$

Based on this similarity metric, we implemented our classifier as outlined in Algorithm 2. The algorithm is given a list of monitored websites W, a traffic signature database T, and the signature t to be classified. As T contains multiple samples (w_i, t_i) for one website, we compute the similarity of t with all these traffic signatures t_j with $1 \leq j \leq n$ corresponding to a specific website w_i. Afterwards, we compute the average similarity with all these traces for this specific website w_i. Finally, we return the website w_i with the highest average similarity s_i compared to the given trace t.

Algorithm 2: Classification algorithm.

```
Input: List of monitored websites $W$, traffic signature database $T$, traffic
       signature $t$
Output: Website $w$

foreach website $w_i$ in $W$ do
    Retrieve all samples $(w_i, t_1), \ldots, (w_i, t_n) \in T$
    $s_i = \text{avg}(\text{Sim}(t_1, t) + \cdots + \text{Sim}(t_n, t))$
    $S \leftarrow \{S, (w_i, s_i)\}$
end
Return $(w_i, s_i) \in S$, s.t. $s_i$ is maximized
```

Compared to network-based fingerprinting attacks, our attack relies on a simple yet efficient classifier. More specifically, we do not need a dedicated training phase that requires several hundred CPU hours in case of some network-based fingerprinting attacks (cf. [WG13, WCN$^+$14]). Still, the testing time of our implemented classifier, *i.e.*, the actual time to classify a traffic trace, is comparable to the testing time of existing fingerprinting attacks and yields highly accurate results as will be discussed in Section 8.6.5.

8.6 Evaluation and Results

We now evaluate the classification rate for a setting where a standard browser application connects to websites directly. We use this setting to evaluate the intra-day classification rate, *i.e.*, when gathering training data and test data on

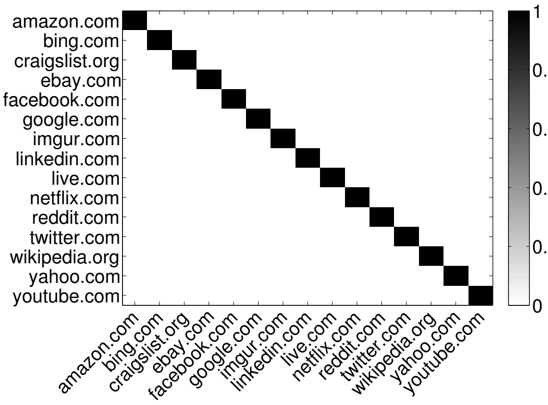

Figure 8.5: Confusion matrix for the 15 most popular websites in the US.

the same day. Subsequently, we perform a similar investigation in a setting where the traffic is routed through the Tor network. In addition, we also investigate how the classification rate decreases over time, *i.e.*, in case the test data is gathered a few days after gathering the training data, and we evaluate the scalability of our attack for larger sets of monitored websites. Finally, we compare our results to related work. All experiments in this section have been performed with data-usage statistics captured via WLAN connections.

8.6.1 Intra-Day Classification Rate

For our first experiment, we aim for a standard browser setting and gather the training data and the test data on the same day. Therefore, we took the 15 most popular websites in the US according to `alexa.com` and we gathered $n = 5$ samples for each of these websites to establish the signature database T. In order to estimate the performance of our classifier, we performed a leave-one-out cross validation. Thus, for each sample $(w_i, t_i) \in T$, we removed this sample (one at a time) from the traffic signature database T, and called the classification algorithm (Algorithm 2) with the traffic signature database $T \setminus (w_i, t_i)$ and the traffic signature t_i to be classified.

Figure 8.5 shows the resulting confusion matrix. We indicate the ground truth along the y-axis and the inferred website along the x-axis. Since all page visits to each of the 15 websites have been classified correctly, we achieve a success rate of 100%. More formally, each sample (w_i, t_i) has been classified correctly considering the signature database $T \setminus (w_i, t_i)$ for training the classifier.

If we have a look at the 100 most popular websites globally, then we observe a classification rate of 89% for a total of 500 page visits. After further investigations of the resulting confusion matrix, we noticed that several misclassifications occur because `google*.*` pages have been misclassified among each other. For example, `google.es` has been misclassified as either `google.fr`, `google.it`, or

Table 8.2: Google websites that have been merged.

google.co.in	google.co.jp	google.de
google.co.uk	google.com.br	google.fr
google.ru	google.it	google.es
google.ca	google.com.mx	google.com.hk
google.com.tr	google.co.id	google.pl
google.com.au	google.co.kr	googleadservices.com

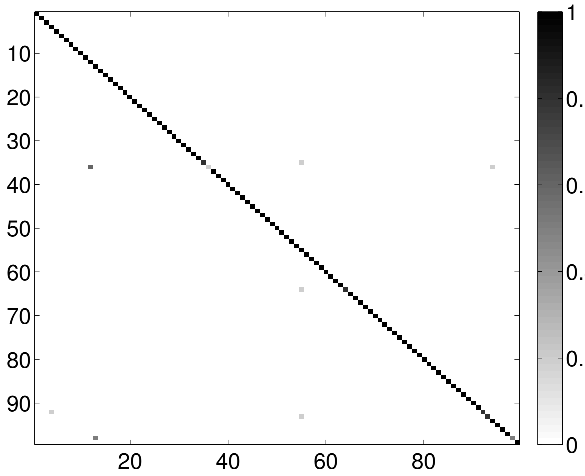

Figure 8.6: Confusion matrix for the 100 most popular websites globally with *google*.** pages merged.

`google.pl`. Nevertheless, we do not aim for a detailed classification of different Google domains and, hence, we merged all Google websites listed in Table 8.2 to be classified as `google.com`.

Performing the classification again, we achieve a classification rate of 98% for these 500 page visits. The corresponding confusion matrix can be seen in Figure 8.6. Merging these Google websites leads to a total of 9 misclassified websites among these 500 page visits, with `mail.ru` at index 36 being the most commonly misclassified website (4 times).

8.6.2 Classification Rate for Tor

We also evaluated our attack in the intra-day setting for traffic routed through Tor. Therefore, we gathered $n = 5$ samples for the top 100 websites in the US by capturing the data-usage statistics of the Orweb browser, but we used a profiling time of 20 seconds as websites accessed via Tor take more time to load. Again, we performed a leave-one-out cross validation resulting in a classification rate of 95% for these 500 page visits.

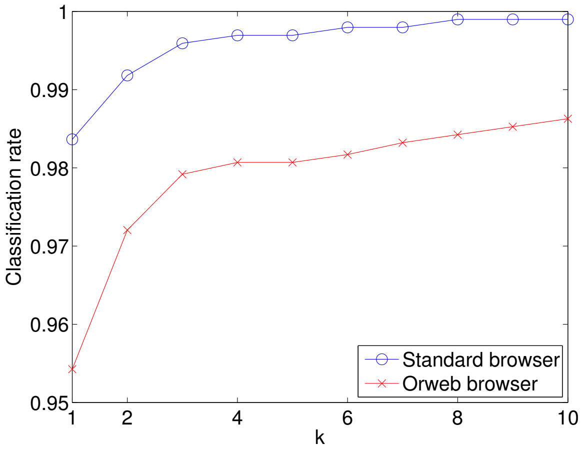

Figure 8.7: Classification rates considering the k most probable websites returned from the classifier.

If we instruct the classifier to return a set of k possible websites, which are sorted according to their probability for being the correct one, then the success rate steadily increases with the number of websites k taken into consideration. As can be seen in Figure 8.7, taking the two most probable websites ($k = 2$) into consideration, we achieve a classification rate of 97% and taking the three most probable websites ($k = 3$) into consideration yields a classification rate of 98%. Similar classification rates can be observed for the standard Android browser where the traffic is not routed through Tor. Although the standard browser yields slightly better classification rates, we did not observe significant differences between the classification rates of a browser accessing websites directly and a browser accessing websites via Tor.

Security Implications. Even though browsers (e.g., the Orweb browser) or specific browser modes (e.g., "private/incognito" modes) do not store the browsing history and the network traffic is protected against specific attacks while being routed through the Tor network, an unprivileged application can infer the users' browsing behavior for monitored websites. Thus, as our experiments demonstrate, the `READ_HISTORY_BOOKMARKS` permission does not protect the users' privacy and even routing the network traffic through Tor provides a false sense of privacy for Android users. Furthermore, given the fact that Orweb disables JavaScript and Flash by default, users do not use Orweb to access sites that heavily rely on these techniques. Hence, attackers explicitly targeting Tor users can significantly reduce the set of monitored websites, which increases the success rate. However, we do not blame the browsers for these security implications but the Android OS.

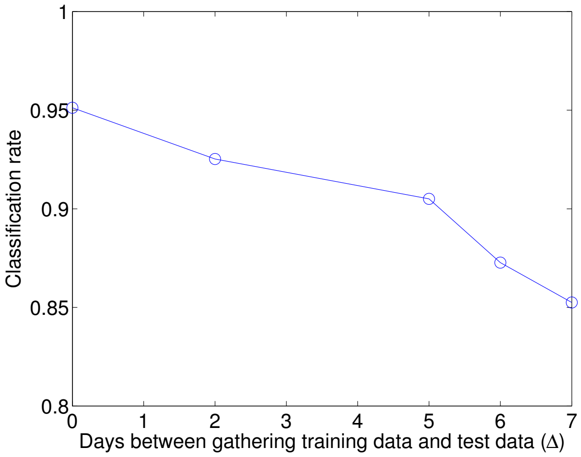

Figure 8.8: Decreasing accuracy for samples captured Δ days after the training data.

8.6.3 Inter-Day Classification Rate

We also performed experiments with an outdated training set, which allows us to investigate whether an attacker's workload can be reduced by relying on the same training data for several days. In order to do so, we used Orweb to collect measurement samples for the top 100 websites in the US a few days after gathering the training data. The evaluation in Figure 8.8 shows that the accuracy decreases rather slowly. For test data gathered on the same day as the training data ($\Delta = 0$), 95% of all page visits can be inferred. Using the same data to classify 500 page visits captured two days later ($\Delta = 2$) still yields a classification accuracy of 93%. Testing measurement samples captured after five days ($\Delta = 5$) yields a classification accuracy of 91%. Even data gathered one week later ($\Delta = 7$) achieves a classification rate of 85%. Hence, even slightly outdated training data allows to infer websites accurately. This slowly decreasing classification rate allows an adversary to keep the traffic signature database T for some time, meaning that the adversary does not need to update the database for the monitored websites on a daily basis. Thereby, the attacker's effort and workload can be reduced significantly, which leads to more practical attacks. Furthermore, this also indicates that training samples can be gathered after the attack samples, which represents another advantage for the attacker (cf. [LL06]).

Table 8.3 holds the worst websites (according to their classification accuracy) in case the attack samples are captured Δ days after the training data. As news sites tend to change frequently, it is not surprising that the classification rate decreases for such websites. Based on this observation, an attacker can selectively rebuild the training set for frequently changing websites like news portals, while the training data for more static websites can be kept for a longer period.

We point out that we did not employ any measures to ensure that we exit Tor with a specific country IP address. Instead, every time we gathered new

Table 8.3: Websites with the worst classifications in the inter-day setting.

Δ	Website	# misclassifications
2 days	`ask.com`	5 times
2 days	`twitch.tv`	5 times
2 days	`cnn.com`	3 times
5 days	`bbc.com`	5 times
5 days	`indeed.com`	5 times
5 days	`nytimes.com`	5 times
5 days	`twitch.tv`	5 times
5 days	`espn.go.com`	4 times
6 days	`bbc.com`	5 times
6 days	`indeed.com`	5 times
6 days	`nytimes.com`	5 times
6 days	`twitch.tv`	5 times
6 days	`espn.go.com`	4 times
7 days	`bbc.com`	5 times
7 days	`bleacherreport.com`	5 times
7 days	`indeed.com`	5 times
7 days	`nytimes.com`	5 times
7 days	`twitch.tv`	5 times
7 days	`xfinity.com`	5 times
7 days	`homedepot.com`	4 times

test samples, we reconnected to the Tor network by restarting the Orbot proxy. Nevertheless, the classification rates indicate a high success rate even with a training set that is not completely up to date.

8.6.4 Scalability for Larger World Sizes

In order to investigate the scalability of our attack for a larger set of monitored websites, we consider the top 500 websites. However, we did not route the network traffic through Tor as this would have taken significantly longer. Our results indicate that we are able to classify 97% of 2 500 page visits out of a set of 500 monitored websites.

8.6.5 Comparison with Related Work

As our attack only requires an unprivileged Android app that can be easily deployed via available app markets, it is easier to conduct than wiretapping attacks where the attacker observes TCP packets on the victim's network. Due to the ease of applicability, the computational performance of the classifier, and the classification rates, our attack outperforms existing fingerprinting attacks.

Table 8.4 provides a comprehensive comparison of website fingerprinting attacks in the closed-world setting. For each attack, we present the attacker or the exploited information in column 2. Column 3 indicates the caching behavior of the browser. Column 4 shows the employed classifier. For the sake of brevity, we do not list the exact features that are used for classification purposes, but we refer the interested reader to the corresponding works. Column 5 shows the attacked countermeasure, where "none" refers to no specific countermeasure, "SSH tunnel" means that the client hides its network traffic through a proxy where the encrypted communication between the client and the proxy is observed, and "Tor" means that the traffic is routed through the anonymity network Tor. As most website fingerprinting attacks consider a closed-world setting, we state the number of monitored websites in column 6. Furthermore, we indicate the accuracy within the last column.

The only works in Table 8.4 that exploit client-side side-channel information are the work of Jana and Shmatikov [JS12] as well as ours. To be more precise, Jana and Shmatikov do not exploit information that relates to the TCP packet lengths. Instead, they exploit the memory footprint of the browser while rendering the monitored website. However, our approach of exploiting the data-usage statistics allows a significantly more accurate inference of visited websites.

Compared to wiretapping attacks against the anonymity network Tor, we observe that our approach of exploiting client-side information leaks yields mostly better results. Though, the works of Wang and Goldberg [WG13] and Wang et al. [WCN$^+$14] achieve a rather similar classification accuracy, *i.e.*, they achieve a classification accuracy of 91% [WG13] and 95% [WCN$^+$14] on a set of 100 monitored websites. However, training their classifiers, which are based on the optimal string alignment distance and a weighted k-NN, takes a significant amount of time (608 000 CPU seconds [WG13]). This, however, is not feasible for every attacker. As our classifier does not require a training phase, we significantly reduce the computational effort. Our test algorithm scales linearly with the number of monitored websites and the corresponding samples, *i.e.*, $\mathcal{O}(|W| \cdot n)$. A naive implementation of our classifier (Algorithm 2) in Matlab—without any specific optimizations—takes a testing time of about 0.4 seconds on an Intel Core i7-5600U CPU for a set of 100 monitored websites and 5 samples per website. This testing time is comparable to the testing time of 0.7 seconds reported by Wang and Goldberg [WG13], but in contrast to their work we do not require an expensive training phase.

Wiretapping vs. Side-Channel Observations. The subtle difference between observing traffic information on the wire and the side-channel information of the data-usage statistics requires further considerations. In the wiretapping scenario it is impossible to miss single packets, whereas in the side-channel scenario the attacker might miss single packets which are then observed as one "large" packet due to the accumulation of TCP packets within the data-usage statistics. However, wiretapping attacks against Tor need to rely on the observation of padded packets only, whereas we also observe unpadded packets due to the separation of the browser and proxy application on the smartphone.

Table 8.4: Comparison of website fingerprinting attacks in the closed-world setting.

Work	Exploited information	Caching	Classifier	Countermeasure	# websites	Accuracy
Ours	Client-side data-usage statistics	Enabled	Jaccard index	None	500	**97%**
Jana and Shmatikov [JS12]	Client-side memory footprint	Enabled?[a]	Jaccard index	None	100	35%
Cai et al. [CZJJ12]	TCP packets captured via tshark	Disabled	Damerau-Levenshtein distance	SSH tunnel	100	92%
Herrmann et al. [HWF09]	Client-side tcpdump	Disabled	Naive-Bayes classifier	SSH tunnel	775	96%
Liberatore and Levine [LL06]	Client-side tcpdump	Disabled	Jaccard index	SSH tunnel	500	79%
Liberatore and Levine [LL06]	Client-side tcpdump	Disabled	Naive-Bayes classifier	SSH tunnel	500	75%
Ours	Client-side data-usage statistics	Disabled	Jaccard index	Tor	100	**95%**
Herrmann et al. [HWF09]	Client-side tcpdump	Disabled	Multinomial Naive-Bayes classifier	Tor	775	3%
Cai et al. [CZJJ12]	TCP packets captured via tshark	Disabled	Damerau-Levenshtein distance	Tor	100	84%
Panchenko et al. [PNZE11]	Client-side tcpdump	Disabled	Support vector machines	Tor	775	55%
Wang and Goldberg [WG13]	TCP packets[b]	Disabled	Optimal string alignment distance[c]	Tor	100	91%
Wang and Goldberg [WG13]	TCP packets[b]	Disabled	Levenshtein distance	Tor	100	70%
Wang et al. [WCN$^+$14]	TCP packets[d]	Disabled	k-nearest neighbor[e]	Tor	100	95%

---

[a] We are not sure whether caching has been disabled for the Android browser.

[b] They parsed TCP packets to obtain the underlying Tor cells but did not specify how to obtain the TCP packets.

[c] Training took 608 000 CPU seconds.

[d] They used the same data set as in [WG13].

[e] The computational complexity of the training phase is similar to [WG13] (cf. footnote c).

To summarize this comparison, we consider it easier to deploy an application than to wiretap a victim's network, but we trade the exact observation of single TCP packet lengths (or padded packets in case of Tor) for a slightly less accurate side-channel observation. The most significant advantage of our attack is that it only requires an unprivileged Android application and a simple (yet efficient) classifier, which allows to deploy this attack on a large scale. In addition, our client-side attack overcomes many limitations and assumptions of network-based fingerprinting attacks that are considered unrealistic, *i.e.*, our attack is invariant to background traffic and does not require expensive training phases.

8.7 Discussion of Countermeasures

We now provide an overview of existing countermeasures. However, most of these countermeasures have been proposed to mitigate wiretapping attacks and, thus, we discuss the relevance of these defense mechanisms against our attack.

8.7.1 Existing Countermeasures

Traffic Morphing. Wright et al. [WCM09] suggested traffic morphing, which requires the cooperation of the target web server or proxy as well as the browser. Each packet from a website is padded or split in such a way that the traffic information of the actual website matches the traffic information of a different website. As a result, an attacker observing the traffic information will most likely misclassify this website.

HTTPOS. Luo et al. [LZC$^+$11] proposed a browser-based defense mechanism denoted as HTTP or HTTPS with Obfuscation (HTTPOS). HTTPOS focuses on changing packet sizes and packet timings, which can be done, for instance, by adding bytes to the referer header or by using the HTTP range option to fetch specific portions of websites.

BuFLO. Dyer et al. [DCRS12] presented Buffered Fixed-Length Obfuscator (BuFLO) that sends fixed-length packets at fixed intervals for a fixed amount of time. While the authors claim that BuFLO significantly reduces the attack surface, it is rather inefficient in terms of bandwidth overhead. Again, BuFLO requires the cooperation of the involved proxies. Cai et al. [CNJ14] proposed an extension denoted as Congestion-Sensitive Buffered Fixed-Length Obfuscator (CS-BuFLO).

Glove. Nithyanand et al. [NCJ14] proposed Glove that tries to cluster similar websites according to pre-selected features. Based on these clusters, transcripts of packet sizes (denoted as super-traces) are computed, which are later transmitted whenever a page in the corresponding cluster is loaded. This super-trace is obtained by inserting, merging, splitting, and delaying packets. The major drawback of Glove is that the traces must be updated regularly, which is rather expensive.

8.7.2 Discussion

The above mentioned countermeasures aim at preventing website fingerprinting attacks. Nevertheless, network-level defenses do not provide effective countermeasures against client-side attacks. For instance, if the traffic is routed through the Tor network, then the traffic might be protected on the network. However, unless the browser application itself is actively involved in these defense mechanisms, these countermeasures do not provide protection against client-side attackers. We demonstrated this by exploiting the data-usage statistics of browser applications that route the traffic through the Tor network. Furthermore, application-level defenses like HTTPOS might prevent such attacks at first glance, but Cai et al. [CNJ14] already demonstrated that a network attacker is able to circumvent this countermeasure. Besides, many network-level defenses add a significant overhead in terms of bandwidth and data consumption, which is impractical for mobile devices with a limited data plan. We consider further investigations regarding the effectiveness of countermeasures against client-side attacks as an interesting open research problem. Furthermore, new proposals for countermeasures should consider mobile devices.

Client-Side Countermeasures. Besides these proposed countermeasures, which mostly target network-level attackers, fixing fundamental design flaws of Android should be considered as absolutely necessary. Zhou et al. [ZDH$^+$13] suggested two permission-based approaches. The first one is a new permission that allows applications to monitor the data-usage statistics. The second one is to let applications define how data-usage statistics should be published. We, however, do not consider these two approaches as viable countermeasures for the following reasons. First, many users either do not pay attention to the requested permissions during the installation or they do not understand the meaning of these permissions (cf. [FHE$^+$12, KCC$^+$12]). Second, the permission system also confuses developers which leads to overprivileged applications (cf. [FCH$^+$11]). Besides, developers might not be aware that the data-usage statistics of their application leaks sensitive information and that their applications should impose restrictions on how to publish these statistics.

A more general approach to prevent such side-channel attacks at a larger scale has been suggested by Zhang et al. [ZYN$^+$15]. The idea of their approach is that an application (*App Guardian*) pauses/stops suspicious background processes when the application to be protected (principal) is executed. This idea sounds quite appealing but still struggles with unsolved issues like a proper identification of malicious processes.

A first solution to defend against such attacks would be to update these statistics according to a more coarse-grained granularity. We stress that data-usage statistics capturing single TCP packet lengths represents a significant threat. Updating data-usage statistics in a more coarse-grained interval, e.g., on a daily basis, should suffice for users to keep an eye on their data consumption. Future work might come up with more advanced countermeasures and the above outlined approach of *App Guardian* [ZYN$^+$15] definitely follows the right direction towards the prevention of such attacks.

8.8 Conclusion

In this chapter, we investigated a new type of client-side website fingerprinting attack that exploits the data-usage statistics published by the Android OS. We argue that the private browsing mode (incognito mode) of browsers, the `READ_HISTORY_BOOKMARKS` permission, and even routing the network traffic through Tor provides a false sense of privacy for smartphone users. Even though the browser application itself does not store any information about visited websites and the traffic is protected while being routed through the anonymity network Tor, the data-usage statistics leak sensitive information. We demonstrated that any application is able to accurately infer a user's visited websites without any suspicious permission. Hence, the Android permission `READ_HISTORY_BOOKMARKS`—which is supposed to protect the browsing behavior—is actually irrelevant as it does not provide protection.

Compared to existing website fingerprinting attacks, our attack can be deployed significantly easier and allows for a more accurate classification of websites. The ease of applicability allows even less sophisticated attackers to perform accurate website fingerprinting attacks on a large scale, which clearly proves the immense threat arising from this information leak. As a user's browsing behavior reveals rather sensitive information, we urge the need to address this issue on the operating system level. Furthermore, due to the simple (yet accurate) classification algorithm, real-time detection of a user's browsing behavior in combination with sensor-based keyloggers allows for large-scale identity theft attacks which must be prevented by all means.

9

Conclusions

You have to fight for your privacy or you lose it.
— Eric Schmidt

In this thesis, we argued that privacy and security are inevitably tied to each other and, thus, both aspects must be addressed to ensure a future environment that can be considered as secure and privacy preserving. Although privacy-preserving mechanisms such as group signature schemes have already been proposed, the practical applicability of such mechanisms is somehow limited due to specific deficiencies or missing features. In the first part of this thesis we addressed challenges of group signature schemes that arise when implementing group signature schemes in practice. We proposed a generic compiler that adds controllable linkability to group signature schemes. While this feature might be interesting for privacy-preserving data mining applications, it also allows us to propose an efficient solution to the open problem of membership revocation in group signatures, which overcomes several limitations of existing revocation mechanisms. The proposed revocation mechanism represents the most flexible and efficient approach for membership revocation at the cost of an always-online revocation authority. Nevertheless, such a transition from offline/semi-online revocation mechanisms towards always-online revocation authorities seems to be reasonable due to the availability of reliable cloud infrastructures today. Overall the proposed revocation mechanism represents a valuable addendum to the existing portfolio of revocation mechanisms as it provides dedicated features and advantages compared to other existing revocation mechanisms.

In the second part of this thesis we aimed to advance the field of mobile security and to foster a thorough understanding of side-channel attacks on mobile devices. Especially the five key enablers identified in this thesis (cf. Chapter 5) allow for devastating side-channel attacks that can be conducted remotely and, thus, target an immense number of devices and users at the same time. We demonstrated the immense threat resulting from resources and seemingly harmless information being shared among all processes on multi-purpose devices by proposing new attacks and enhancements to existing attacks. Unfortunately, we expect the situation to become worse as the combination of multiple information leaks allows to launch even more powerful attacks, e.g., when combining cache attacks with sensor-based keyloggers, or when combining sensor-based keylogging with website fingerprinting attacks.

Although we put a strong focus on smartphones in this thesis, we stress that wearables in general must be considered in future research. Especially with the ever increasing number of smart devices, e.g., smartwatches, and the increasing number of devices in the IoT network being inter-connected and accessible via the Internet, the threat of side-channel attacks increases and will lead to new and even more sophisticated attacks. For instance, attacks exploiting accelerometer sensors embedded in smartwatches and attacks exploiting publicly available information on Android-based IoT devices have already been conducted. For example, Zhang et al. [ZYN⁺15] demonstrated that due to the communication between a WiFi camera and an Android smartphone, a side-channel attack on Android-based smartphone allows to determine whether or not the user's home is empty. This example demonstrates that side-channel leaks do not only pose a threat to a user's privacy and security from a system security point of view, but also pose a threat to smart home appliances and security systems, such as smart thermostats, cameras, and alarm systems. Although this sounds dystopian at first, side-channel leaks (on smartphones) also pose a threat to IoT appliances and even put users' physical possessions at risk.

At this point we do not know how to cope with side-channel attacks appropriately, but clearly countermeasures need to be researched. Proposed countermeasures, however, also need to take the mobile ecosystem of Android into consideration, which might impede a large-scale deployment of many defense mechanisms. Even if Google would apply defense mechanisms and patch vulnerabilities, multiple device manufacturers as well as carriers also need to apply these patches to deploy countermeasures in practice. Hence, chances are that such countermeasures will never be deployed in practice. Therefore, more generic countermeasures that (1) aim to prevent software-based side-channel attacks in general and (2) can be deployed at a larger scale are needed. We expect the situation to become worse, as side-channel attacks work on other platforms as well, and the popularity of HTML5 apps and the increasing availability of web APIs to access native resources from JavaScript significantly increases the scale of side-channel attacks as specific attacks possibly target multiple platforms at the same time.

Bibliography

[ABB+16] Yasemin Acar, Michael Backes, Sven Bugiel, Sascha Fahl, Patrick Drew McDaniel, and Matthew Smith. SoK: Lessons Learned from Android Security Research for Appified Software Platforms. In *IEEE Symposium on Security and Privacy – S&P 2016*, pages 433–451. IEEE, 2016.

[ACHO11] Masayuki Abe, Sherman S. M. Chow, Kristiyan Haralambiev, and Miyako Ohkubo. Double-Trapdoor Anonymous Tags for Traceable Signatures. In *Applied Cryptography and Network Security – ACNS 2011*, volume 6715 of *LNCS*, pages 183–200, 2011.

[AE13] Hassan Aly and Mohammed ElGayyar. Attacking AES Using Bernstein's Attack on Modern Processors. In *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *LNCS*, pages 127–139. Springer, 2013.

[AEHS14] Nuttapong Attrapadung, Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. A Revocable Group Signature Scheme from Identity-Based Revocation Techniques: Achieving Constant-Size Revocation List. In *Applied Cryptography and Network Security – ACNS 2014*, volume 8479 of *LNCS*, pages 419–437. Springer, 2014.

[AGM+10] Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith. Smudge Attacks on Smartphone Touch Screens. In *Workshop on Offensive Technologies – WOOT 2010*. USENIX Association, 2010.

[AK06] Onur Aciiçmez and Çetin Kaya Koç. Trace-Driven Cache Attacks on AES (Short Paper). In *Information and Communications Security – ICICS 2006*, volume 4307 of *LNCS*, pages 112–121. Springer, 2006.

[AK16] Hasan Faik Alan and Jasleen Kaur. Can Android Applications Be Identified Using Only TCP/IP Headers of Their Launch Time Traffic? In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 61–66. ACM, 2016.

[Alp10] Ethem Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning).* The MIT Press, 2nd edition, 2010.

[ALWS15] Kamran Ali, Alex X. Liu, Wei Wang, and Muhammad Shahzad. Keystroke Recognition Using WiFi Signals. In *Mobile Computing and Networking – MOBICOM 2015*, pages 90–102. ACM, 2015.

[ARM10] ARM Ltd. *ARM Technical Reference Manual, Cortex-A8, Revision: r3p2*, May 2010.

[ASBS12] Adam J. Aviv, Benjamin Sapp, Matt Blaze, and Jonathan M. Smith. Practicality of Accelerometer Side Channels on Smartphones. In *Annual Computer Security Applications Conference – ACSAC 2012*, pages 41–50. ACM, 2012.

[ASM06] Man Ho Au, Willy Susilo, and Yi Mu. Constant-Size Dynamic k-TAA. In *Security and Cryptography for Networks – SCN 2006*, volume 4116 of *LNCS*, pages 111–125. Springer, 2006.

[AST02] Giuseppe Ateniese, Dawn Xiaodong Song, and Gene Tsudik. Quasi-Efficient Revocation in Group Signatures. In *Financial Cryptography – FC 2002*, volume 2357 of *LNCS*, pages 183–197. Springer, 2002.

[ATOY13] Panagiotis Andriotis, Theo Tryfonas, George C. Oikonomou, and Can Yildiz. A Pilot Study on the Security of Pattern Screen-Lock Methods and Soft Side Channel Attacks. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2013*, pages 1–6. ACM, 2013.

[ATSM09] Man Ho Au, Patrick P. Tsang, Willy Susilo, and Yi Mu. Dynamic Universal Accumulators for DDH Groups and Their Application to Attribute-Based Anonymous Credential Systems. In *Topics in Cryptology – CT-RSA 2009*, volume 5473 of *LNCS*, pages 295–308. Springer, 2009.

[Avi12] Adam J. Aviv. *Side Channels Enable By Smartphone Interaction.* PhD thesis, University of Pennsylvania, 2012.

[Bar] Barclays PLC. Mobile Banking Services. `http://www.barclays.co.uk/Mobile/BarclaysPingit/P1242603570446`. Accessed: November 2013.

[BB04] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, 2004.

[BB08] Dan Boneh and Xavier Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *J. Cryptology*, 21:149–177, 2008.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, 2004.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct Anonymous Attestation. In *Conference on Computer and Communications Security – CCS*, pages 132–145. ACM, 2004.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the Importance of Checking Cryptographic Protocols for Faults (Extended Abstract). In *Advances in Cryptology – EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 37–51. Springer, 1997.

[BDSS16] Olivier Blazy, David Derler, Daniel Slamanig, and Raphael Spreitzer. Non-Interactive Plaintext (In-)Equality Proofs and Group Signatures with Verifiable Controllable Linkability. In *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *LNCS*, pages 127–143. Springer, 2016.

[BDU08] Michael Backes, Markus Dürmuth, and Dominique Unruh. Compromising Reflections-or-How to Read LCD Monitors around the Corner. In *IEEE Symposium on Security and Privacy – S&P 2008*, pages 158–169. IEEE Computer Society, 2008.

[Ber05] Daniel J. Bernstein. Cache-Timing Attacks on AES. Available online at `http://cr.yp.to/antiforgery/cachetiming-20050414.pdf`, 2005.

[BFMT16] Pierre Belgarric, Pierre-Alain Fouque, Gilles Macario-Rat, and Mehdi Tibouchi. Side-Channel Analysis of Weierstrass and Koblitz Curve ECDSA on Android Smartphones. In *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *LNCS*, pages 236–252. Springer, 2016.

[BGH$^+$13] Michael Backes, Sebastian Gerling, Christian Hammer, Matteo Maffei, and Philipp von Styp-Rekowsky. AppGuard – Enforcing User Requirements on Android Apps. In *Tools and Algorithms for the Construction and Analysis of Systems – TACAS 2013*, volume 7795 of *LNCS*, pages 543–548. Springer, 2013.

[BHMT16] Joppe W. Bos, Charles Hubain, Wil Michiels, and Philippe Teuwen. Differential Computation Analysis: Hiding Your White-Box Designs is Not Enough. In *Cryptographic Hardware and Embedded Systems – CHES 2016*, volume 9813 of *LNCS*, pages 215–236. Springer, 2016.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, 2006.

[BLJL05] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Privacy Enhancing Technologies – PET 2005*, volume 3856 of *LNCS*, pages 1–11. Springer, 2005.

[BMNB14] Hristo Bojinov, Yan Michalevsky, Gabi Nakibly, and Dan Boneh. Mobile Device Identification via Sensor Fingerprinting. *arXiv ePrint Archive, Report 1408.1416*, 2014.

[BMW03] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.

[BPA12] Joseph Bonneau, Sören Preibusch, and Ross J. Anderson. A Birthday Present Every Eleven Wallets? The Security of Customer-Chosen Banking PINs. In *Financial Cryptography – FC 2012*, volume 7397 of *LNCS*, pages 25–40. Springer, 2012.

[BS97] Eli Biham and Adi Shamir. Differential Fault Analysis of Secret Key Cryptosystems. In *Advances in Cryptology – CRYPTO 1997*, volume 1294 of *LNCS*, pages 513–525. Springer, 1997.

[BS01] Emmanuel Bresson and Jacques Stern. Efficient Revocation in Group Signatures. In *Public Key Cryptography – PKC 2001*, volume 1992 of *LNCS*, pages 190–206. Springer, 2001.

[BS04] Dan Boneh and Hovav Shacham. Group Signatures with Verifier-Local Revocation. In *Conference on Computer and Communications Security – CCS*, pages 168–177. ACM, 2004.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of Group Signatures: The Case of Dynamic Groups. In *Topics in Cryptology – CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, 2005.

[BZ04] Joonsang Baek and Yuliang Zheng. Identity-Based Threshold Decryption. In *Public Key Cryptography – PKC 2004*, volume 2947 of *LNCS*, pages 262–276. Springer, 2004.

[CC11] Liang Cai and Hao Chen. TouchLogger: Inferring Keystrokes on Touch Screen from Smartphone Motion. In *USENIX Workshop on Hot Topics in Security – HotSec*. USENIX Association, 2011.

[CC12] Liang Cai and Hao Chen. On the Practicality of Motion Based Keystroke Inference Attack. In *Trust and Trustworthy Computing – TRUST 2012*, volume 7344 of *LNCS*, pages 273–290. Springer, 2012.

[CCdMP10] Sébastien Canard, Iwen Coisel, Giacomo de Meulenaer, and Olivier Pereira. Group Signatures are Suitable for Constrained Devices. In *Information Security and Cryptology – ICISC 2010*, volume 6829 of *LNCS*, pages 133–150. Springer, 2010.

[CDDT12] Sébastien Canard, Nicolas Desmoulins, Julien Devigne, and Jacques Traoré. On the Implementation of a Pairing-Based Cryptographic Protocol in a Constrained Device. In *Pairing-Based Cryptography – Pairing 2012*, volume 7708 of *LNCS*, pages 210–217. Springer, 2012.

[Cho09] Sherman S. M. Chow. Real Traceable Signatures. In *Selected Areas in Cryptography – SAC 2009*, volume 5867 of *LNCS*, pages 92–107. Springer, 2009.

[CL02] Jan Camenisch and Anna Lysyanskaya. Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials. In *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature Schemes and Anonymous Credentials from Bilinear Maps. In *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, 2004.

[CM11] Sanjit Chatterjee and Alfred Menezes. On Cryptographic Protocols Employing Asymmetric Pairings – The Role of Ψ Revisited. *Discrete Applied Mathematics*, 159:1311–1322, 2011.

[CMC09] Liang Cai, Sridhar Machiraju, and Hao Chen. Defending Against Sensor-Sniffing Attacks on Mobile Phones. In *Workshop on Networking, Systems, and Applications for Mobile Handhelds – MobiHeld*, pages 31–36. ACM, 2009.

[CMSV16] Mauro Conti, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. Analyzing Android Encrypted Network Traffic to Identify User Actions. *IEEE Trans. Information Forensics and Security*, 11:114–125, 2016.

[CNJ14] Xiang Cai, Rishab Nithyanand, and Rob Johnson. CS-BuFLO: A Congestion Sensitive Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society – WPES 2014*, pages 121–130. ACM, 2014.

[CNRS16] Mauro Conti, Michele Nati, Enrico Rotundo, and Riccardo Spolaor. Mind The Plug! Laptop-User Recognition Through Power Consumption. In *Workshop on IoT Privacy, Trust, and Security – IoTPTS@AsiaCCS*, pages 37–44. ACM, 2016.

[CPY06] Seung Geol Choi, Kunsoo Park, and Moti Yung. Short Traceable Signatures Based on Bilinear Pairings. In *International Workshop on Security – IWSEC 2006*, volume 4266 of *LNCS*, pages 88–103. Springer, 2006.

[CQM14] Qi Alfred Chen, Zhiyun Qian, and Zhuoqing Morley Mao. Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks. In *USENIX Security Symposium 2014*, pages 1037–1052. USENIX Association, 2014.

[CS97] Jan Camenisch and Markus Stadler. Efficient Group Signature Schemes for Large Groups (Extended Abstract). In *Advances in Cryptology – CRYPTO 1997*, volume 1294 of *LNCS*, pages 410–424. Springer, 1997.

[CS98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology – CRYPTO 1998*, volume 1462 of *LNCS*, pages 13–25. Springer, 1998.

[CSY06] Sherman S. M. Chow, Willy Susilo, and Tsz Hon Yuen. Escrowed Linkability of Ring Signatures and Its Applications. In *Progress in Cryptology – VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 175–192. Springer, 2006.

[CvH91] David Chaum and Eugène van Heyst. Group Signatures. In *Advances in Cryptology – EUROCRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.

[CWWZ10] Shuo Chen, Rui Wang, XiaoFeng Wang, and Kehuan Zhang. Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow. In *IEEE Symposium on Security and Privacy – S&P 2010*, pages 191–206. IEEE Computer Society, 2010.

[CZJJ12] Xiang Cai, Xin Cheng Zhang, Brijesh Joshi, and Rob Johnson. Touching From a Distance: Website Fingerprinting Attacks and Defenses. In *Conference on Computer and Communications Security – CCS 2012*, pages 605–616. ACM, 2012.

[DBC14] Anupam Das, Nikita Borisov, and Matthew Caesar. Do You Hear What I Hear?: Fingerprinting Smart Devices Through Embedded Acoustic Components. In *Conference on Computer and Communications Security – CCS 2014*, pages 441–452. ACM, 2014.

[DBC16] Anupam Das, Nikita Borisov, and Matthew Caesar. Tracking Mobile Web Users Through Motion Sensors: Attacks and Defenses. In *Network and Distributed System Security Symposium – NDSS 2016*. The Internet Society, 2016.

[DCRS12] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-Boo, I Still See You: Why Efficient Traffic Analysis Countermeasures Fail. In *IEEE Symposium on Security and Privacy – S&P 2012*, pages 332–346. IEEE Computer Society, 2012.

[DLLZ16] Wenrui Diao, Xiangyu Liu, Zhou Li, and Kehuan Zhang. No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis. In *IEEE Symposium on Security and Privacy – S&P 2016*, pages 414–432. IEEE, 2016.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium 2004*, pages 303–320. USENIX, 2004.

[DoM] DoMobile. AppLock. `https://play.google.com/store/apps/details?id=com.domobile.applock`. Accessed: November 2013.

[DP06] Cécile Delerablée and David Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *Progress in Cryptology – VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 193–210. Springer, 2006.

[DRX$^+$14] Sanorita Dey, Nirupam Roy, Wenyuan Xu, Romit Roy Choudhury, and Srihari Nelakuditi. AccelPrint: Imperfections of Accelerometers Make Smartphones Trackable. In *Network and Distributed System Security Symposium – NDSS 2014*. The Internet Society, 2014.

[DSCP02] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards Measuring Anonymity. In *Privacy Enhancing Technologies – PET 2002*, volume 2482 of *LNCS*, pages 54–68. Springer, 2002.

[DY05] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography – PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.

[EH15] Keita Emura and Takuya Hayashi. A Light-Weight Group Signature Scheme with Time-Token Dependent Linking. In *Lightweight Cryptography for Security and Privacy – LightSec 2015*, volume 9542 of *LNCS*, pages 37–57. Springer, 2015.

[EMO14] Keita Emura, Atsuko Miyaji, and Kazumasa Omote. An r-Hiding Revocable Group Signature Scheme: Group Signatures with the Property of Hiding the Number of Revoked Users. *J. Applied Mathematics*, 2014:983040:1–983040:14, 2014.

[Eur10] European Parliament and Council. Directive 2010/40 on the Deployment of Intelligent Transport Systems in the Field of Road Transport and for Interfaces with Other Modes of Transport, July 2010.

[Eve] Evernote Corporation. Evernote. `https://play.google.com/store/apps/details?id=com.evernote`. Accessed: November 2013.

[FCH$^+$11] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android Permissions Demystified. In *Conference on Computer and Communications Security – CCS 2011*, pages 627–638. ACM, 2011.

[FEF$^+$12] Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, and David Wagner. How to Ask for Permission. In *USENIX Workshop on Hot Topics in Security – HotSec*. USENIX Association, 2012.

[FHE$^+$12] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Symposium On Usable Privacy and Security – SOUPS 2012*, page 3. ACM, 2012.

[FHM11] Chun-I Fan, Ruei-Hau Hsu, and Mark Manulis. Group Signature with Constant Revocation Costs for Signers and Verifiers. In *Cryptology and Network Security – CANS 2011*, volume 7092 of *LNCS*, pages 214–233. Springer, 2011.

[FKH14] Tobias Fiebig, Jan Krissler, and Ronny Hänsch. Security Impact of High Resolution Smartphone Cameras. In *Workshop on Offensive Technologies – WOOT 2014*. USENIX Association, 2014.

[FKMV12] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the Non-malleability of the Fiat-Shamir Transform. In *Progress in Cryptology – INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, 2012.

[FP01] Pierre-Alain Fouque and David Pointcheval. Threshold Cryptosystems Secure against Chosen-Ciphertext Attacks. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 351–368. Springer, 2001.

[FS86] Amos Fiat and Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Advances in Cryptology – CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.

[FS00] Edward W. Felten and Michael A. Schneider. Timing Attacks on Web Privacy. In *Conference on Computer and Communications Security – CCS 2000*, pages 25–32. ACM, 2000.

[GAL$^+$12] Gurleen Grewal, Reza Azarderakhsh, Patrick Longa, Shi Hu, and David Jao. Efficient Implementation of Bilinear Pairings on ARM

Processors. In *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *LNCS*, pages 149–165. Springer, 2012.

[GBK11] David Gullasch, Endre Bangerter, and Stephan Krenn. Cache Games – Bringing Access-Based Cache Attacks on AES to Practice. In *IEEE Symposium on Security and Privacy – S&P 2011*, pages 490–505. IEEE Computer Society, 2011.

[GBKS12] Xun Gong, Nikita Borisov, Negar Kiyavash, and Nabil Schear. Website Detection Using Remote Traffic Analysis. In *Privacy Enhancing Technologies – PET 2012*, volume 7384 of *LNCS*, pages 58–78. Springer, 2012.

[GBM15] Daniel Gruss, David Bidner, and Stefan Mangard. Practical Memory Deduplication Attacks in Sandboxed Javascript. In *European Symposium on Research in Computer Security – ESORICS 2015*, volume 9326 of *LNCS*, pages 108–122. Springer, 2015.

[GCEC12] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. In *Trust and Trustworthy Computing – TRUST 2012*, volume 7344 of *LNCS*, pages 291–307. Springer, 2012.

[Gha14] Essam Ghadafi. Efficient Distributed Tag-Based Encryption and Its Application to Group Signatures with Efficient Distributed Traceability. In *Progress in Cryptology – LATINCRYPT 2014*, volume 8895 of *LNCS*, pages 327–347. Springer, 2014.

[GHSS15] Hannes Gross, Marko Hölbl, Daniel Slamanig, and Raphael Spreitzer. Privacy-Aware Authentication in the Internet of Things. In *Cryptology and Network Security – CANS 2015*, volume 9476 of *LNCS*, pages 32–39. Springer, 2015.

[GHT05] Catherine H. Gebotys, Simon Ho, and C. C. Tiu. EM Analysis of Rijndael and ECC on a Wireless Java-Based PDA. In *Cryptographic Hardware and Embedded Systems – CHES 2005*, volume 3659 of *LNCS*, pages 250–264. Springer, 2005.

[GHV08] Steven D. Galbraith, Florian Hess, and Frederik Vercauteren. Aspects of Pairing Inversion. *IEEE Transactions on Information Theory*, 54:5719–5728, 2008.

[GK11] Jean-François Gallais and Ilya Kizhvatov. Error-Tolerance in Trace-Driven Cache Collision Attacks. In *Constructive Side-Channel Analysis and Secure Design – COSADE*, pages 222–232, 2011.

[GKB10] Xun Gong, Negar Kiyavash, and Nikita Borisov. Fingerprinting Websites Using Remote Traffic Analysis. In *Conference on Computer and Communications Security – CCS 2010*, pages 684–686. ACM, 2010.

[GKT10] Jean-François Gallais, Ilya Kizhvatov, and Michael Tunstall. Improved Trace-Driven Cache-Collision Attacks against Embedded AES Implementations. In *Information Security Applications – WISA 2010*, volume 6513 of *LNCS*, pages 243–257. Springer, 2010.

[GMM16] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In *Detection of Intrusions and Malware & Vulnerability Assessment – DIMVA 2016*, volume 9721 of *LNCS*, pages 300–321. Springer, 2016.

[Goo14] Google. Simplified Permissions on Google Play. `https://support.google.com/googleplay/answer/6014972`, 2014.

[GS08] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, 2008.

[GS13] Benoît Gérard and François-Xavier Standaert. Unified and Optimized Linear Collision Attacks and Their Application in a Non-Profiled Setting: Extended Version. *J. Cryptographic Engineering*, 3:45–58, 2013.

[GSAV16] Haritabh Gupta, Shamik Sural, Vijayalakshmi Atluri, and Jaideep Vaidya. Deciphering Text from Touchscreen Key Taps. In *Data and Applications Security and Privacy – DBSec 2016*, volume 9766 of *LNCS*, pages 3–18. Springer, 2016.

[GSM15] Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In *USENIX Security Symposium 2015*, pages 897–912. USENIX Association, 2015.

[Gür10] Fahriye Seda Gürses. *Multilateral Privacy Requirements Analysis in Online Social Network Services*. PhD thesis, Katholieke Universiteit Leuven, 2010.

[GYCH16] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A Survey of Microarchitectural Timing Attacks and Countermeasures on Contemporary Hardware. *Journal of Cryptographic Engineering*, pages 1–27, 2016.

[HAZ+16] Muzammil Hussain, Ahmed Al-Haiqi, A. A. Zaidan, B. B. Zaidan, M. L. Mat Kiah, Nor Badrul Anuar, and Mohamed Abdulnabi. The Rise of Keyloggers on Smartphones: A Survey and Insight Into Motion-Based Tap Inference Attacks. *Pervasive and Mobile Computing*, 25:1–25, 2016.

[HCCN15] Jung Yeon Hwang, Liqun Chen, Hyun Sook Cho, and DaeHun Nyang. Short Dynamic Group Signature Scheme Supporting Controllable Linkability. *IEEE Trans. Information Forensics and Security*, 10:1109–1124, 2015.

[HD15] Jamie Hayes and George Danezis. Better Open-World Website Fingerprinting. *arXiv ePrint Archive, Report 1509.00789*, 2015.

[HHH16] Thomas Hupperich, Henry Hosseini, and Thorsten Holz. Leveraging Sensor Fingerprinting for Mobile Device Authentication. In *Detection of Intrusions and Malware & Vulnerability Assessment – DIMVA 2016*, volume 9721 of *LNCS*, pages 377–396. Springer, 2016.

[Hin02] Andrew Hintz. Fingerprinting Websites Using Traffic Analysis. In *Privacy Enhancing Technologies – PET 2002*, volume 2482 of *LNCS*, pages 171–178. Springer, 2002.

[HL10] Carmit Hazay and Yehuda Lindell. *Sigma Protocols and Efficient Zero-Knowledge*, pages 147–175. Information Security and Cryptography. Springer, 2010.

[HLC+11] Jung Yeon Hwang, Sokjoon Lee, Byung-Ho Chung, Hyun Sook Cho, and DaeHun Nyang. Short Group Signatures with Controllable Linkability. In *Lightweight Security & Privacy: Devices, Protocols and Applications – LightSec*, pages 44–52, March 2011.

[HLC+13] Jung Yeon Hwang, Sokjoon Lee, Byung-Ho Chung, Hyun Sook Cho, and DaeHun Nyang. Group Signatures With Controllable Linkability for Dynamic Membership. *Inf. Sci.*, 222:761–778, 2013.

[HMK+15] Thomas Hupperich, Davide Maiorca, Marc Kührer, Thorsten Holz, and Giorgio Giacinto. On the Robustness of Mobile Device Fingerprinting: Can Mobile Users Escape Modern Web-Tracking Mechanisms? In *Annual Computer Security Applications Conference – ACSAC 2015*, pages 191–200. ACM, 2015.

[HMSS15] Bo-Jhang Ho, Paul D. Martin, Prashanth Swaminathan, and Mani B. Srivastava. From Pressure to Path: Barometer-based Vehicle Tracking. In *Embedded Systems for Energy-Efficient Built Environments – BuildSys*, pages 65–74. ACM, 2015.

[HNT13] Samuli Hemminki, Petteri Nurmi, and Sasu Tarkoma. Accelerometer-Based Transportation Mode Detection on Smartphones. In *Conference on Embedded Network Sensor Systems – SenSys 2013*, pages 13:1–13:14. ACM, 2013.

[HON$^+$12] Jun Han, Emmanuel Owusu, Le T. Nguyen, Adrian Perrig, and Joy Zhang. ACComplice: Location Inference Using Accelerometers on Smartphones. In *International Conference on Communication Systems and Networks – COMSNETS 2012*, pages 1–9. IEEE, 2012.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The Temperature Side Channel and Heating Fault Attacks. In *Smart Card Research and Advanced Applications – CARDIS 2013*, volume 8419 of *LNCS*, pages 219–235. Springer, 2013.

[HS14] Michael Hutter and Jörn-Marc Schmidt. The Temperature Side Channel and Heating Fault Attacks. *IACR Cryptology ePrint Archive, Report 2014/190*, 2014.

[HWF09] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier. In *Cloud Computing Security Workshop – CCSW*, pages 31–42. ACM, 2009.

[HYG$^+$14] Gaofeng He, Ming Yang, Xiaodan Gu, Junzhou Luo, and Yuanyuan Ma. A Novel Active Website Fingerprinting Attack Against Tor Anonymous System. In *Computer Supported Cooperative Work in Design – CSCWD 2014*, pages 112–117. IEEE, 2014.

[IiRPP15] Andreu Pere Isern-Deyà, Llorenç Huguet i Rotger, Magdalena Payeras-Capellà, and Macià Mut Puigserver. On the Practicability of Using Group Signatures on Mobile Devices: Implementation and Performance Analysis on the Android Platform. *Int. J. Inf. Sec.*, 14:335–345, 2015.

[Int13] International Organization for Standardization (ISO). ISO/IEC 20008-2: Information Technology – Security Techniques – Anonymous Digital Signatures – Part 2: Mechanisms Using a Group Public Key, November 2013.

[IVP$^+$13] Andreu Pere Isern-Deyà, Arnau Vives-Guasch, Macià Mut Puigserver, Magdalena Payeras-Capellà, and Jordi Castellà-Roca. A Secure Automatic Fare Collection System for Time-Based or Distance-Based Services with Revocable Anonymity for Users. *Comput. J.*, 56:1198–1215, 2013.

[JAA$^+$14] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. A Critical Evaluation of Website Fingerprinting Attacks. In *Conference on Computer and Communications Security – CCS 2014*, pages 263–274. ACM, 2014.

[JKO13] Marek Jawurek, Florian Kerschbaum, and Claudio Orlandi. Zero-Knowledge Using Garbled Circuits: How to Prove Non-Algebraic Statements Efficiently. In *Conference on Computer and Communications Security – CCS 2013*, pages 955–966. ACM, 2013.

[JL13] Markus Jakobsson and Debin Liu. Your Password is Your New PIN. In *Mobile Authentication — Problems and Solutions*, pages 25–36. Springer, 2013.

[JO10] Artur Janc and Lukasz Olejnik. Web Browser History Detection as a Real-World Privacy Threat. In *European Symposium on Research in Computer Security – ESORICS 2010*, volume 6345 of *LNCS*, pages 215–231. Springer, 2010.

[JS12] Suman Jana and Vitaly Shmatikov. Memento: Learning Secrets from Process Footprints. In *IEEE Symposium on Security and Privacy – S&P 2012*, pages 143–157. IEEE Computer Society, 2012.

[KCC+12] Patrick Gage Kelley, Sunny Consolvo, Lorrie Faith Cranor, Jaeyeon Jung, Norman M. Sadeh, and David Wetherall. A Conundrum of Permissions: Installing Applications on an Android Smartphone. In *Financial Cryptography and Data Security Workshops – USEC and WECSR*, volume 7398 of *LNCS*, pages 68–79. Springer, 2012.

[KDK+14] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors. In *International Symposium on Computer Architecture – ISCA 2014*, pages 361–372. IEEE Computer Society, 2014.

[Kee] KeepSafe. KeepSafe. `https://play.google.com/store/apps/details?id=com.kii.safe`. Accessed: November 2013.

[KGB+16] Andreas Kurtz, Hugo Gascon, Tobias Becker, Konrad Rieck, and Felix C. Freiling. Fingerprinting Mobile Devices Using Personalized Configurations. *PoPETs*, 2016:4–19, 2016.

[KHH16] Katharina Krombholz, Thomas Hupperich, and Thorsten Holz. Use the Force: Evaluating Force-Sensitive Authentication for Mobile Devices. In *Symposium On Usable Privacy and Security – SOUPS 2016*, pages 207–219. USENIX Association, 2016.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO 1999*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999.

[KLP$^+$15] Vireshwar Kumar, He Li, Jung-Min ”Jerry” Park, Kaigui Bian, and Yaling Yang. Group Signatures with Probabilistic Revocation: A Computationally-Scalable Approach for Providing Privacy-Preserving Authentication. In *Conference on Computer and Communications Security – CCS 2015*, pages 1334–1345. ACM, 2015.

[KN14] Taekyoung Kwon and Sarang Na. TinyLock: Affordable Defense Against Smudge Attacks on Smartphone Pattern Lock Systems. *Computers & Security*, 42:137–150, 2014.

[Koc96] Paul C. Kocher. Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology – CRYPTO 1996*, volume 1109 of *LNCS*, pages 104–113. Springer, 1996.

[KS04] Satoshi Koga and Kouichi Sakurai. A Distributed Online Certificate Status Protocol with a Single Public Key. In *Public Key Cryptography – PKC 2004*, volume 2947 of *LNCS*, pages 389–401. Springer, 2004.

[KSWH98] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. Side Channel Cryptanalysis of Product Ciphers. In *European Symposium on Research in Computer Security – ESORICS 1998*, volume 1485 of *LNCS*, pages 97–110. Springer, 1998.

[KTY04] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable Signatures. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 571–589. Springer, 2004.

[KY05] Aggelos Kiayias and Moti Yung. Group Signatures with Efficient Concurrent Join. In *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 198–214. Springer, 2005.

[LCC10] Liming Lu, Ee-Chien Chang, and Mun Choon Chan. Website Fingerprinting and Identification Using Ordered Feature Sequences. In *European Symposium on Research in Computer Security – ESORICS 2010*, volume 6345 of *LNCS*, pages 199–214. Springer, 2010.

[LGS$^+$16] Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. ARMageddon: Cache Attacks on Mobile Devices. In *USENIX Security Symposium 2016*, pages 549–564. USENIX Association, 2016.

[LL06] Marc Liberatore and Brian Neil Levine. Inferring the Source of Encrypted HTTP Connections. In *Conference on Computer and Communications Security – CCS*, pages 255–263. ACM, 2006.

[LML$^+$16] Mengyuan Li, Yan Meng, Junyi Liu, Haojin Zhu, Xiaohui Liang, Yao Liu, and Na Ruan. When CSI Meets Public WiFi: Inferring

Your Mobile Phone Password via WiFi Signals. In *Conference on Computer and Communications Security – CCS 2016*, pages 1068–1079. ACM, 2016.

[LPY12a] Benoît Libert, Thomas Peters, and Moti Yung. Group Signatures with Almost-for-Free Revocation. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *LNCS*, pages 571–589. Springer, 2012.

[LPY12b] Benoît Libert, Thomas Peters, and Moti Yung. Scalable Group Signatures with Revocation. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 609–627. Springer, 2012.

[LRT12] Victor Lomné, Thomas Roche, and Adrian Thillard. On the Need of Randomness in Fault Attack Countermeasures – Application to AES. In *Fault Diagnosis and Tolerance in Cryptography – FDTC 2012*, pages 85–94. IEEE Computer Society, 2012.

[LY09] Benoît Libert and Moti Yung. Efficient Traceable Signatures in the Standard Model. In *Pairing-Based Cryptography – Pairing 2009*, volume 5671 of *LNCS*, pages 187–205. Springer, 2009.

[LYL$^+$14] Bin Liang, Wei You, Liangkun Liu, Wenchang Shi, and Mario Heiderich. Scriptless Timing Attacks on Web Browser Privacy. In *Dependable Systems and Networks – DSN 2014*, pages 112–123. IEEE Computer Society, 2014.

[LZC$^+$11] Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. HTTPOS: Sealing Information Leaks with Browser-Side Obfuscation of Encrypted Flows. In *Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, 2011.

[Mat] MathWorks. Statistics Toolbox. `http://www.mathworks.com/products/statistics/`. Accessed: November 2013.

[MBN14] Yan Michalevsky, Dan Boneh, and Gabi Nakibly. Gyrophone: Recognizing Speech from Gyroscope Signals. In *USENIX Security Symposium 2014*, pages 1053–1067. USENIX Association, 2014.

[MFG$^+$12] Mark Manulis, Nils Fleischhacker, Felix Günther, Franziskus Kiefer, and Bertram Poettering. Group Signatures: Authentication with Privacy. Technical report, BSI – Federal Office for Information Security, 2012.

[MGB11] Federico Maggi, Simone Gasparini, and Giacomo Boracchi. A Fast Eavesdropping Attack Against Touchscreens. In *Information Assurance and Security – IAS 2011*, pages 320–325. IEEE, 2011.

[MHJT14] Brad Miller, Ling Huang, Anthony D. Joseph, and J. D. Tygar. I Know Why You Went to the Clinic: Risks and Realization of HTTPS Traffic Analysis. In *Privacy Enhancing Technologies – PET 2014*, volume 8555 of *LNCS*, pages 143–163. Springer, 2014.

[MLLB15] Stanislav Miskovic, Gene Moo Lee, Yong Liao, and Mario Baldi. AppPrint: Automatic Fingerprinting of Mobile Applications in Network Traffic. In *Passive and Active Measurement – PAM 2015*, volume 8995 of *LNCS*, pages 57–69. Springer, 2015.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks – Revealing the Secrets of Smart Cards*. Springer, 2007.

[MS13] Tilo Müller and Michael Spreitzenbarth. FROST – Forensic Recovery of Scrambled Telephones. In *Applied Cryptography and Network Security – ACNS 2013*, volume 7954 of *LNCS*, pages 373–388. Springer, 2013.

[MSV$^+$15] Yan Michalevsky, Aaron Schulman, Gunaa Arumugam Veerapandian, Dan Boneh, and Gabi Nakibly. PowerSpy: Location Tracking Using Mobile Device Power Analysis. In *USENIX Security Symposium 2015*, pages 785–800. USENIX Association, 2015.

[MTSH16a] Maryam Mehrnezhad, Ehsan Toreini, Siamak Fayyaz Shahandashti, and Feng Hao. Stealing PINs via Mobile Sensors: Actual Risk versus User Perception. *arXiv ePrint Archive, Report 1605.05549*, 2016.

[MTSH16b] Maryam Mehrnezhad, Ehsan Toreini, Siamak Fayyaz Shahandashti, and Feng Hao. TouchSignatures: Identification of User Touch Actions and PINs Based on Mobile Sensor Data via JavaScript. *J. Inf. Sec. Appl.*, 26:23–38, 2016.

[MVBC12] Emiliano Miluzzo, Alexander Varshavsky, Suhrid Balakrishnan, and Romit Roy Choudhury. Tapprints: Your Finger Taps Have Fingerprints. In *Mobile Systems – MobiSys 2012*, pages 323–336. ACM, 2012.

[NAB] NAB. NAB. `https://play.google.com/store/apps/details?id=au.com.nab.mobile`. Accessed: November 2013.

[Nah16] Ani Nahapetian. Side-Channel Attacks on Mobile and Wearable Systems. In *Consumer Communications & Networking Conference – CCNC 2016*, pages 243–247. IEEE, 2016.

[Nat01] National Institute of Standards and Technology (NIST). FIPS-197: Advanced Encryption Standard, November 2001.

[NCJ14] Rishab Nithyanand, Xiang Cai, and Rob Johnson. Glove: A Bespoke Website Fingerprinting Defense. In *Workshop on Privacy in the Electronic Society – WPES 2014*, pages 131–134. ACM, 2014.

[Nev06] Michael Neve. *Cache-based Vulnerabilities and SPAM Analysis.* PhD thesis, UCL, 2006.

[New16] NewAE Technology Inc. Fault Injection Raspberry PI. `https://wiki.newae.com`, 2016. Accessed: August 2016.

[NF05] Toru Nakanishi and Nobuo Funabiki. Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps. In *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 533–548. Springer, 2005.

[NF06] Toru Nakanishi and Nobuo Funabiki. A Short Verifier-Local Revocation Group Signature Scheme with Backward Unlinkability. In *International Workshop on Security – IWSEC 2006*, volume 4266 of *LNCS*, pages 17–32. Springer, 2006.

[NFHF09] Toru Nakanishi, Hiroki Fujii, Yuta Hira, and Nobuo Funabiki. Revocable Group Signature Schemes with Constant Costs for Signing and Verifying. In *Public Key Cryptography – PKC 2009*, volume 5443 of *LNCS*, pages 463–480. Springer, 2009.

[NFW99] Toru Nakanishi, Toru Fujiwara, and Hajime Watanabe. A Linkable Group Signature and Its Application to Secret Voting. *Trans. of Information Processing Society of Japan*, 40(7), 1999.

[NKHF05] Toru Nakanishi, Fumiaki Kubooka, Naoto Hamada, and Nobuo Funabiki. Group Signature Schemes with Membership Revocation for Large Groups. In *Information Security and Privacy – ACISP 2005*, volume 3574 of *LNCS*, pages 443–454. Springer, 2005.

[NM14] Sarfraz Nawaz and Cecilia Mascolo. Mining Users' Significant Driving Routes With Low-Power Sensors. In *Conference on Embedded Network Sensor Systems – SenSys 2014*, pages 236–250. ACM, 2014.

[NS04a] Toru Nakanishi and Yuji Sugiyama. A Group Signature Scheme with Efficient Membership Revocation for Reasonable Groups. In *Information Security and Privacy – ACISP 2004*, volume 3108 of *LNCS*, pages 336–347. Springer, 2004.

[NS04b] Lan Nguyen and Reihaneh Safavi-Naini. Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 372–386. Springer, 2004.

[NS06a] Arvind Narayanan and Vitaly Shmatikov. How To Break Anonymity of the Netflix Prize Dataset. *arXiv ePrint Archive, Report 0610105*, 2006.

[NS06b] Michael Neve and Jean-Pierre Seifert. Advances on Access-Driven Cache Attacks on AES. In *Selected Areas in Cryptography – SAC 2006*, volume 4356 of *LNCS*, pages 147–162. Springer, 2006.

[NSN+14a] Yuto Nakano, Youssef Souissi, Robert Nguyen, Laurent Sauvage, Jean-Luc Danger, Sylvain Guilley, Shinsaku Kiyomoto, and Yutaka Miyake. A Pre-processing Composition for Secret Key Recovery on Android Smartphone. In *Information Security Theory and Practice – WISTP 2014*, volume 8501 of *LNCS*, pages 76–91. Springer, 2014.

[NSN14b] Sashank Narain, Amirali Sanatinia, and Guevara Noubir. Single-Stroke Language-Agnostic Keylogging Using Stereo-Microphones and Domain Specific Machine Learning. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2014*, pages 201–212. ACM, 2014.

[NSW06] Michael Neve, Jean-Pierre Seifert, and Zhenghong Wang. A Refined Look at Bernstein's AES Side-Channel Analysis. In *Conference on Computer and Communications Security – CCS 2006*, page 369. ACM, 2006.

[NVBN16] Sashank Narain, Triet D. Vo-Huu, Kenneth Block, and Guevara Noubir. Inferring User Routes and Locations Using Zero-Permission Mobile Sensors. In *IEEE Symposium on Security and Privacy – S&P 2016*, pages 397–413. IEEE, 2016.

[NY90] Moni Naor and Moti Yung. Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks. In *Symposium on the Theory of Computing – STOC*, pages 427–437. ACM, 1990.

[O'F16] Colin O'Flynn. Fault Injection using Crowbars on Embedded Systems. *IACR Cryptology ePrint Archive, Report 2016/810*, 2016.

[OGSM16] S. Ordas, L. Guillaume-Sage, and P. Maurine. Electromagnetic Fault Injection: The Curse of Flip-Flops. *Journal of Cryptographic Engineering*, pages 1–15, 2016.

[OHD+12] Emmanuel Owusu, Jun Han, Sauvik Das, Adrian Perrig, and Joy Zhang. ACCessory: Password Inference Using Accelerometers on Smartphones. In *Mobile Computing Systems and Applications – HotMobile 2012*, page 9. ACM, 2012.

[OKSK15] Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. The Spy in the Sandbox: Practical Cache Attacks in JavaScript and their Implications. In *Conference on Com-*

puter and Communications Security – CCS 2015, pages 1406–1418. ACM, 2015.

[Pag02] Dan Page. Theoretical Use of Cache Memory as a Cryptanalytic Side-Channel. *IACR Cryptology ePrint Archive, Report 2002/169*, 2002.

[PGS$^+$12] Hao Peng, Christopher S. Gates, Bhaskar Pratim Sarma, Ninghui Li, Yuan Qi, Rahul Potharaju, Cristina Nita-Rotaru, and Ian Molloy. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In *Conference on Computer and Communications Security – CCS 2012*, pages 241–252. ACM, 2012.

[PLP$^+$16] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website Fingerprinting at Internet Scale. In *Network and Distributed System Security Symposium – NDSS 2016*. The Internet Society, 2016.

[PNZE11] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website Fingerprinting in Onion Routing Based Anonymization Networks. In *Workshop on Privacy in the Electronic Society – WPES 2011*, pages 103–114. ACM, 2011.

[Pon15] Ponemon Institute LLC. 2015 PKI Global Trends Study, 2015.

[PR04] Rasmus Pagh and Flemming Friche Rodler. Cuckoo hashing. *J. Algorithms*, 51:122–144, 2004.

[PS16] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. In *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, 2016.

[PSM15] Dan Ping, Xin Sun, and Bing Mao. TextLogger: Inferring Longer Inputs on Touch Screen Using Motion Sensors. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2015*, pages 24:1–24:12. ACM, 2015.

[PWH$^+$13] Klaus Potzmader, Johannes Winter, Daniel M. Hein, Christian Hanser, Peter Teufl, and Liqun Chen. Group Signatures on Mobile Devices: Practical Experiences. In *Trust and Trustworthy Computing – TRUST 2013*, volume 7904 of *LNCS*, pages 47–64. Springer, 2013.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards. In *Smart Card Programming and Security – E-smart 2001*, volume 2140 of *LNCS*, pages 200–210. Springer, 2001.

[RGKS11] Andrew Raij, Animikh Ghosh, Santosh Kumar, and Mani B. Srivastava. Privacy Risks Emerging From the Adoption of Innocuous Wearable Sensors in the Mobile Environment. In *Conference on Human Factors in Computing Systems – CHI 2011*, pages 11–20. ACM, 2011.

[RKM+12] Franziska Roesner, Tadayoshi Kohno, Alexander Moshchuk, Bryan Parno, Helen J. Wang, and Crispin Cowan. User-Driven Access Control: Rethinking Permission Granting in Modern Operating Systems. In *IEEE Symposium on Security and Privacy – S&P 2012*, pages 224–238. IEEE Computer Society, 2012.

[RNR+15] Lionel Rivière, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. High Precision Fault Injections on the Instruction Cache of ARMv7-M Architectures. In *Hardware Oriented Security and Trust – HOST 2015*, pages 62–67. IEEE Computer Society, 2015.

[RS91] Charles Rackoff and Daniel R. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Advances in Cryptology – CRYPTO 1991*, volume 576 of *LNCS*, pages 433–444. Springer, 1991.

[RSDT13] Cyril Roscian, Alexandre Sarafianos, Jean-Max Dutertre, and Assia Tria. Fault Model Analysis of Laser-Induced Faults in SRAM Memory Cells. In *Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*, pages 89–98. IEEE Computer Society, 2013.

[RWG+11] Rahul Raguram, Andrew M. White, Dibyendusekhar Goswami, Fabian Monrose, and Jan-Michael Frahm. iSpy: Automatic Reconstruction of Typed Input From Compromising Reflections. In *Conference on Computer and Communications Security – CCS 2011*, pages 527–536. ACM, 2011.

[RWX+13] Rahul Raguram, Andrew M. White, Yi Xu, Jan-Michael Frahm, Pierre Georgel, and Fabian Monrose. On the Privacy Risks of Virtual Keyboards: Automatic Reconstruction of Typed Input from Compromising Reflections. *IEEE Trans. Dependable Sec. Comput.*, 10:154–167, 2013.

[SA02] Sergei P. Skorobogatov and Ross J. Anderson. Optical Fault Induction Attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, volume 2523 of *LNCS*, pages 2–12. Springer, 2002.

[SA13] Laurent Simon and Ross Anderson. PIN Skimmer: Inferring PINs Through the Camera and Microphone. In *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*, pages 67–78. ACM, 2013.

[Sama] Samsung. Samsung Galaxy S4 Specifications. `http://www.samsung.com/global/microsite/galaxys4/`. Accessed: November 2013.

[Samb] Samsung. Samsung Galaxy SIII Specifications. `http://www.samsung.com/global/galaxys3/specifications.html`. Accessed: November 2013.

[Samc] Samsung. Samsung KNOX. `http://www.samsung.com/global/business/mobile/solution/security/samsung-knox`. Accessed: November 2013.

[Sam13] Samsung. What You May Not Know About GALAXY S4. `http://global.samsungtomorrow.com/?p=23610`, 2013. Accessed: November 2013.

[SD15] Mark Seaborn and Thomas Dullien. Exploiting the DRAM Rowhammer Bug to Gain Kernel Privileges, 2015. Blackhat.

[SFKD14] Vishal Saraswat, Daniel Feldman, Denis Foo Kune, and Satyajit Das. Remote Cache-Timing Attacks Against AES. In *Cryptography and Security in Computing Systems – CS2@HiPEAC*, pages 45–48. ACM, 2014.

[SFSM13] Tim Stöber, Mario Frank, Jens B. Schmitt, and Ivan Martinovic. Who Do You Sync You Are?: Smartphone Fingerprinting via Application Behaviour. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2013*, pages 7–12. ACM, 2013.

[SG14] Raphael Spreitzer and Benoît Gérard. Towards More Practical Time-Driven Cache Attacks. In *Information Security Theory and Practice – WISTP 2014*, volume 8501 of *LNCS*, pages 24–39. Springer, 2014.

[SGKM16] Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 49–60. ACM, 2016.

[Sha79] Adi Shamir. How to Share a Secret. *Commun. ACM*, 22:612–613, 1979.

[SIYA11] Kuniyasu Suzaki, Kengo Iijima, Toshiki Yagi, and Cyrille Artho. Memory Deduplication as a Threat to the Guest OS. In *European Workshop on System Security – EUROSEC 2011*, pages 1–6. ACM, 2011.

[SJC$^+$16] Jingchao Sun, Xiaocong Jin, Yimin Chen, Jinxue Zhang, Yanchao Zhang, and Rui Zhang. VISIBLE: Video-Assisted Keystroke Inference from Tablet Backside Motion. In *Network and Distributed*

System Security Symposium – NDSS 2016. The Internet Society, 2016.

[SKH+16] Matthias Schulz, Patrick Klapper, Matthias Hollick, Erik Tews, and Stefan Katzenbeisser. Trust The Wire, They Always Told Me!: On Practical Non-Destructive Wire-Tap Attacks Against Ethernet. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 43–48. ACM, 2016.

[Sko16] Sergei Skorobogatov. The Bumpy Road Towards iPhone 5c NAND Mirroring. *arXiv ePrint Archive, Report 1609.04327*, 2016.

[SKSP14] Diksha Shukla, Rajesh Kumar, Abdul Serwadda, and Vir V. Phoha. Beware, Your Hands Reveal Your Secrets! In *Conference on Computer and Communications Security – CCS 2014*, pages 904–917. ACM, 2014.

[SMA+13] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Carlisle Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP. RFC 6960, Internet Engineering Task Force (IETF), June 2013. `https://www.ietf.org/rfc/rfc6960.txt`.

[SMdH15] Eloi Sanfelix, Cristofaro Mune, and Job de Haas. Unboxing the White-Box: Practical Attacks Against Obfuscated Ciphers, 2015. Blackhat.

[SMKM16] Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. *arXiv ePrint Archive, Report 1611.03748*, 2016. In submission.

[SMS16] Prakash Shrestha, Manar Mohamed, and Nitesh Saxena. Slogger: Smashing Motion-based Touchstroke Logging with Transparent System Noise. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 67–77. ACM, 2016.

[SP13a] Raphael Spreitzer and Thomas Plos. Cache-Access Pattern Attack on Disaligned AES T-Tables. In *Constructive Side-Channel Analysis and Secure Design – COSADE 2013*, volume 7864 of *LNCS*, pages 200–214. Springer, 2013.

[SP13b] Raphael Spreitzer and Thomas Plos. On the Applicability of Time-Driven Cache Attacks on Mobile Devices. In *Network and System Security – NSS 2013*, volume 7873 of *LNCS*, pages 656–662. Springer, 2013.

[Spr14] Raphael Spreitzer. PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices. In *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*, pages 51–62. ACM, 2014.

[SS14] Raphael Spreitzer and Jörn-Marc Schmidt. Group-Signature Schemes on Constrained Devices: The Gap Between Theory and Practice. In *Cryptography and Security in Computing Systems – CS2@HiPEAC*, pages 31–36. ACM, 2014.

[SSU14] Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security – ISC 2014*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014.

[SSU16] Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group Signatures with Linking-Based Revocation: A Pragmatic Approach for Efficient Revocation Checks. In *Conference on Cryptology & Malicious Security – Mycrypt 2016*, 2016. In press.

[SSW$^+$02] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical Identification of Encrypted Web Browsing Traffic. In *IEEE Symposium on Security and Privacy – S&P 2002*, pages 19–30. IEEE Computer Society, 2002.

[SXA16] Laurent Simon, Wenduan Xu, and Ross Anderson. Don't Interrupt Me While I Type: Inferring Text Entered Through Gesture Typing on Android Keyboards. *PoPETs*, 2016:136–154, 2016.

[SZZ$^+$11] Roman Schlegel, Kehuan Zhang, Xiao-yong Zhou, Mehool Intwala, Apu Kapadia, and XiaoFeng Wang. Soundcomber: A Stealthy and Context-Aware Sound Trojan for Smartphones. In *Network and Distributed System Security Symposium – NDSS 2011*. The Internet Society, 2011.

[Tan12a] Qiang Tang. Public Key Encryption Schemes Supporting Equality Test with Authorisation of Different Granularity. *IJACT*, 2:304–321, 2012.

[Tan12b] Qiang Tang. Public Key Encryption Supporting Plaintext Equality Test and User-Specified Authorization. *Security and Communication Networks*, 5:1351–1362, 2012.

[TFAF13] Junko Takahashi, Toshinori Fukunaga, Kazumaro Aoki, and Hitoshi Fuji. Highly Accurate Key Extraction Method for Access-Driven Cache Attacks Using Correlation Coefficient. In *Information Security and Privacy – ACISP 2013*, volume 7959 of *LNCS*, pages 286–301. Springer, 2013.

[TFS04] Isamu Teranishi, Jun Furukawa, and Kazue Sako. k-Times Anonymous Authentication (Extended Abstract). In *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 308–322. Springer, 2004.

[TML+13] Karim Tobich, Philippe Maurine, Pierre-Yvan Liardet, Mathieu Lisart, and Thomas Ordas. Voltage Spikes on the Substrate to Obtain Timing Faults. In *Digital System Design – DSD 2013*, pages 483–486. IEEE Computer Society, 2013.

[TOS10] Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient Cache Attacks on AES, and Countermeasures. *J. Cryptology*, 23:37–71, 2010.

[TSCM16] Vincent F. Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. AppScanner: Automatic Fingerprinting of Smartphone Apps from Encrypted Network Traffic. In *IEEE European Symposium on Security and Privacy – EURO S&P 2016*, pages 439–454. IEEE, 2016.

[TSS+03] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES Implemented on Computers with Cache. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *LNCS*, pages 62–76. Springer, 2003.

[UK 13] UK Office of Communications. Communications Market Report 2013. `http://media.ofcom.org.uk/2013/08/01/the-reinvention-of-the-1950s-living-room-2/`, 2013.

[UW14] Thomas Unterluggauer and Erich Wenger. Efficient Pairings and ECC for Embedded Systems. In *Cryptographic Hardware and Embedded Systems – CHES 2014*, volume 8731 of *LNCS*, pages 298–315. Springer, 2014.

[vdVFL+16] Victor van der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. Drammer: Deterministic Rowhammer Attacks on Mobile Platforms. In *Conference on Computer and Communications Security – CCS 2016*, pages 1675–1689. ACM, 2016.

[Ver16] Eric R. Verheul. Practical Backward Unlinkable Revocation in FIDO, German e-ID, Idemix and U-Prove. *IACR Cryptology ePrint Archive, Report 2016/217*, 2016.

[VGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and Its Application to Side-Channel Attacks. In *Selected Areas in Cryptography – SAC 2012*, volume 7707 of *LNCS*, pages 390–406. Springer, 2012.

[VGS13] Nicolas Veyrat-Charvillon, Benoît Gérard, and François-Xavier Standaert. Security Evaluations Beyond Computing Power. In *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 126–141. Springer, 2013.

[Vir] VirusTotal. VirusTotal. `https://play.google.com/store/apps/details?id=com.virustotal`. Accessed: November 2013.

[vWWM11] Jasper G. J. van Woudenberg, Marc F. Witteman, and Federico Menarini. Practical Optical Fault Injection on Secure Microcontrollers. In *Fault Diagnosis and Tolerance in Cryptography – FDTC 2011*, pages 91–99. IEEE Computer Society, 2011.

[WCM09] Charles V. Wright, Scott E. Coull, and Fabian Monrose. Traffic Morphing: An Efficient Defense Against Statistical Traffic Analysis. In *Network and Distributed System Security Symposium – NDSS 2009*. The Internet Society, 2009.

[WCN+14] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective Attacks and Provable Defenses for Website Fingerprinting. In *USENIX Security Symposium 2014*, pages 143–157. USENIX Association, 2014.

[Wei05] Victor K. Wei. Tracing-by-Linking Group Signatures. In *Information Security – ISC 2005*, volume 3650 of *LNCS*, pages 149–163. Springer, 2005.

[WG13] Tao Wang and Ian Goldberg. Improved Website Fingerprinting on Tor. In *Workshop on Privacy in the Electronic Society – WPES 2013*, pages 201–212. ACM, 2013.

[WHS12] Michael Weiß, Benedikt Heinz, and Frederic Stumpf. A Cache Timing Attack on AES in Virtualization Environments. In *Financial Cryptography – FC 2012*, volume 7397 of *LNCS*, pages 314–328. Springer, 2012.

[WWAS14] Michael Weiß, Benjamin Weggenmann, Moritz August, and Georg Sigl. On Cache Timing Attacks Considering Multi-core Aspects in Virtualized Embedded Systems. In *International Conference on Trusted Systems – INTRUST 2014*, volume 9473 of *LNCS*, pages 151–167. Springer, 2014.

[WYKH15] Qinglong Wang, Amir Yahyavi, Bettina Kemme, and Wenbo He. I Know What You Did on Your Smartphone: Inferring App Usage Over Encrypted Data Traffic. In *Communications and Network Security – CNS 2015*, pages 433–441. IEEE, 2015.

[XBZ12] Zhi Xu, Kun Bai, and Sencun Zhu. TapLogger: Inferring User Inputs on Smartphone Touchscreens Using On-Board Motion Sensors. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2012*, pages 113–124. ACM, 2012.

[XHW+13] Yi Xu, Jared Heinly, Andrew M. White, Fabian Monrose, and Jan-Michael Frahm. Seeing Double: Reconstructing Obscured Typed

Input From Repeated Compromising Reflections. In *Conference on Computer and Communications Security – CCS 2013*, pages 1063–1074. ACM, 2013.

[YF14] Yuval Yarom and Katrina Falkner. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack. In *USENIX Security Symposium 2014*, pages 719–732. USENIX Association, 2014.

[YGCM15] Lin Yan, Yao Guo, Xiangqun Chen, and Hong Mei. A Study on Power Side Channels on Mobile Devices. In *Symposium on Internetware – Internetware 2015*, pages 30–38. ACM, 2015.

[YGZ^{+}16] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, and Kiran S. Balagani. On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-Channel. *IEEE Trans. Information Forensics and Security*, 14, 2016. In press.

[YLF^{+}14] Qinggang Yue, Zhen Ling, Xinwen Fu, Benyuan Liu, Kui Ren, and Wei Zhao. Blind Recognition of Touched Keys on Mobile Devices. In *Conference on Computer and Communications Security – CCS 2014*, pages 1403–1414. ACM, 2014.

[YTHW10] Guomin Yang, Chik How Tan, Qiong Huang, and Duncan S. Wong. Probabilistic Public Key Encryption with Equality Test. In *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *LNCS*, pages 119–131. Springer, 2010.

[ZDH^{+}13] Xiao-yong Zhou, Soteris Demetriou, Dongjing He, Muhammad Naveed, Xiaorui Pan, XiaoFeng Wang, Carl A. Gunter, and Klara Nahrstedt. Identity, Location, Disease and More: Inferring Your Secrets From Android Public Resources. In *Conference on Computer and Communications Security – CCS 2013*, pages 1017–1028. ACM, 2013.

[ZDLZ14] Zhe Zhou, Wenrui Diao, Xiangyu Liu, and Kehuan Zhang. Acoustic Fingerprinting Revisited: Generate Stable Device ID Stealthily with Inaudible Sound. In *Conference on Computer and Communications Security – CCS 2014*, pages 429–440. ACM, 2014.

[ZGS^{+}15] Lukas Zoscher, Jasmin Grosinger, Raphael Spreitzer, Ulrich Muehlmann, Hannes Gross, and Wolfgang Bösch. Concept for a Security Aware Automatic Fare Collection System Using HF/UHF Dual Band RFID Transponders. In *European Solid State Device Research Conference – ESSDERC 2015*, pages 194–197. IEEE, 2015.

[ZL06] Sujing Zhou and Dongdai Lin. Shorter Verifier-Local Revocation Group Signatures from Bilinear Maps. In *Cryptology and Network Security – CANS 2006*, volume 4301 of *LNCS*, pages 126–143. Springer, 2006.

[ZMHS16] Andreas Zankl, Katja Miller, Johann Heyszl, and Georg Sigl. Towards Efficient Evaluation of a Time-Driven Cache Attack on Modern Processors. In *European Symposium on Research in Computer Security – ESORICS 2016*, volume 9879 of *LNCS*, pages 3–19. Springer, 2016.

[ZSG+16] Lukas Zoscher, Raphael Spreitzer, Hannes Gross, Jasmin Grosinger, Ulrich Muehlmann, Dominik Amschl, Hubert Watzinger, and Wolfgang Bösch. HF/UHF Dual Band RFID Transponders for an Information-Driven Public Transportation System. *Elektrotechnik und Informationstechnik*, 133:163–175, 2016.

[ZXL+12] Yang Zhang, Peng Xia, Junzhou Luo, Zhen Ling, Benyuan Liu, and Xinwen Fu. Fingerprint Attack Against Touch-Enabled Devices. In *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*, pages 57–68. ACM, 2012.

[ZXZ16] Xiaokuan Zhang, Yuan Xiao, and Yinqian Zhang. Return-Oriented Flush-Reload Side Channels on ARM and Their Implications for Android Devices. In *Conference on Computer and Communications Security – CCS 2016*, pages 858–870. ACM, 2016.

[ZYN+15] Nan Zhang, Kan Yuan, Muhammad Naveed, Xiao-yong Zhou, and XiaoFeng Wang. Leave Me Alone: App-Level Protection against Runtime Information Gathering on Android. In *IEEE Symposium on Security and Privacy – S&P 2015*, pages 915–930. IEEE Computer Society, 2015.

[ZZT+16] Jie Zhang, Xiaolong Zheng, Zhanyong Tang, Tianzhang Xing, Xiaojiang Chen, Dingyi Fang, Rong Li, Xiaoqing Gong, and Feng Chen. Privacy Leakage in Mobile Sensing: Your Unlock Passwords Can Be Leaked through Wireless Hotspot Functionality. *Mobile Information Systems*, 2016:8793025:1–8793025:14, 2016.

Index

Σ protocols, 14

Active side-channel attacks, 60
Advanced Encryption Standard (AES), 81
All-or-Nothing Public Key Encryption with Equality Tests, 23
Ambient-light sensor, 94

BSZ model, *see* group signature scheme (GSS)

Cache attacks, 69
  Bernstein's timing attack, 82
CCA-full-anonymity, 22
CDH, 11
co-CDH, 11
co-CDH*, 11
Controllable linkability, 20, 21
  E-linkability, 31
  JP-unforgeability, 31
  LO-linkability, 31
CPA-full-anonymity, 22

Data-usage statistics, 69, 116
DDH, 12
Digital signature
  EUF-CMA, 14
Digital signature scheme, 13
DLIN, 12

Group signature scheme (GSS), 10
  BSZ model, 15
  Group manager (GM), 10
  Linking-based revocation, 39
  Verifier local revocation, 41

Microarchitectural attacks, 69

Passive side-channel attacks, 60
procfs leaks, 68
Public key encryption, 13
  IND-CCA, 13
  IND-CPA, 13
  OW-CCA, 13
  OW-CPA, 13
Public key encryption with equality tests (PKEET), 23

q-SDH, 12

RGB(W) sensor, 97

Sensor-based keyloggers, 70
  based on ambient-light sensor, 93
Side-channel classification system, 62
Side-channel information leaks, 61
Sign-and-encrypt-and-prove (SEP) paradigm, 17
Software-only side-channel attack, 62
SXDH, 12

Threshold secret sharing, 17

Website fingerprinting
  based on client-side side-channel information, 113
  based on wiretapping, 67, 115

Author Index

Abdulnabi, Mohamed 71
Abe, Masayuki 36, 38
Acar, Gunes 67
Acar, Yasemin 58, 76
Aciiçmez, Onur 80
Adams, Carlisle 42
Afroz, Sadia 67
Akhawe, Devdatta 110
Al-Haiqi, Ahmed 71
Alan, Hasan Faik 67
Ali, Kamran 67
Alpaydin, Ethem 100, 102
Aly, Hassan 80, 84, 90, 91
Amschl, Dominik 5
Anderson, Ross 68, 71, 98, 99, 101, 106, 111
Anderson, Ross J. 66, 74, 98
Andriotis, Panagiotis 64
Ankney, Rich 42
Anuar, Nor Badrul 71
Aoki, Kazumaro 84
ARM Ltd. 87
Artho, Cyrille 69
Ateniese, Giuseppe 41
Atluri, Vijayalakshmi 71
Attrapadung, Nuttapong 40, 42
Au, Man Ho 21, 41
August, Moritz 69
Aviv, Adam J. 64, 70, 74, 76, 94, 98, 102, 105, 109–111
Azarderakhsh, Reza 45

Backes, Michael 58, 64, 76, 110
Baek, Joonsang 18
Bai, Kun 70
Balagani, Kiran S. 67
Balakrishnan, Suhrid 70, 94, 102
Baldi, Mario 67
Bangerter, Endre 80
Barclays PLC 99
Becker, Tobias 71, 74
Belgarric, Pierre 63, 74
Bellare, Mihir 1, 10, 15–17, 30
Bernstein, Daniel J. 4, 69, 80, 82, 83
Bian, Kaigui 40, 42, 44
Bidner, David 69, 74, 116
Biham, Eli 65
Bishop, Christopher M. 100, 102, 103
Bissias, George Dean 115
Blaze, Matt 64, 70, 74, 94, 98, 102, 105, 109–111
Blazy, Olivier 5, 51
Block, Kenneth 71
Bojinov, Hristo 71
Boneh, Dan 12, 22, 26, 29, 37, 38, 40, 41, 44, 65, 71, 72, 74
Bonneau, Joseph 98
Boracchi, Giacomo 64
Borisov, Nikita 71, 77, 116
Bos, Herbert 73, 74
Bos, Joppe W. 63, 74
Bösch, Wolfgang 5
Boyen, Xavier 12, 22, 26, 29, 41, 44
Bresson, Emmanuel 41, 44
Brickell, Ernest F. 20
Bringer, Julien 65
Bugiel, Sven 58, 76

Caesar, Matthew 71, 77
Cai, Liang 59, 70, 74
Cai, Xiang 67, 76, 114, 115, 122, 125, 131–134

Camenisch, Jan 10, 17, 20, 29, 41
Canard, Sébastien 45
Castellà-Roca, Jordi 1, 45
Chan, Edmond W. W. 76, 133
Chan, Mun Choon 115
Chang, Ee-Chien 115
Chang, Rocky K. C. 76, 133
Chatterjee, Sanjit 11
Chaum, David 1, 10
Chen, Feng 67, 74
Chen, Hao 59, 70, 74, 110
Chen, Liqun 20, 42, 45, 46
Chen, Qi Alfred 68
Chen, Shuo 116
Chen, Xiangqun 63, 69
Chen, Xiaojiang 67, 74
Chen, Yimin 64, 76
Chin, Erika 76, 110, 134
Cho, Hyun Sook 20–22, 29–33, 42, 44–46
Choi, Seung Geol 35
Choudhury, Romit Roy 70, 71, 74, 94, 102
Chow, Sherman S. M. 36, 38, 42
Chung, Byung-Ho 20–22, 29–33, 42, 45, 46
Claessens, Joris 121
Cock, David 59, 70, 74
Coisel, Iwen 45
Consolvo, Sunny 110, 134
Conti, Mauro 67, 74
Coull, Scott E. 76, 133
Cowan, Crispin 110
Cramer, Ronald 28
Cranor, Lorrie Faith 110, 134
Crussell, Jonathan 110

Daly, Ross 73
Danezis, George 123
Danger, Jean-Luc 63, 65
Das, Anupam 71, 77
Das, Satyajit 80
Das, Sauvik 70, 76, 110
de Haas, Job 66, 74
de Meulenaer, Giacomo 45
Delerablée, Cécile 22, 27, 28, 42, 45, 52
Demetriou, Soteris 3, 59, 69, 72, 74, 75, 116, 117, 122, 134
DeMillo, Richard A. 65
Derler, David 5, 51
Desmoulins, Nicolas 45
Devigne, Julien 45
Dey, Sanorita 71, 74
Diao, Wenrui 68, 71
Díaz, Claudia 67, 121
Dingledine, Roger 118
Dodis, Yevgeniy 29, 36
DoMobile 99
Dullien, Thomas 73
Dürmuth, Markus 64
Dutertre, Jean-Max 66
Dyer, Kevin P. 76, 133

Egelman, Serge 76, 110, 134
ElGayyar, Mohammed 80, 84, 90, 91
Emura, Keita 40–42
Engel, Thomas 67, 115, 132
Erickson, Jeremy 110
European Parliament and Council 2
Evernote Corporation 99

Fahl, Sascha 58, 76
Falkner, Katrina 70, 80
Fallin, Chris 73
Fan, Chun-I 40, 41, 44
Fang, Dingyi 67, 74
Farajidavar, Aydin 67
Faust, Sebastian 15
Federrath, Hannes 132
Feldman, Daniel 80
Felt, Adrienne Porter 76, 110, 134
Felten, Edward W. 116
Fiat, Amos 15, 17
Fiebig, Tobias 71
Finifter, Matthew 110
Fleischhacker, Nils 2, 42, 54
Fouque, Pierre-Alain 33, 63, 74
Frahm, Jan-Michael 64, 74, 76
Frank, Mario 67
Fratantonio, Yanick 73, 74

Freiling, Felix C. 71, 74
Fu, Xinwen 64
Fuji, Hitoshi 84
Fujii, Hiroki 41, 44
Fujiwara, Toru 20, 35, 38
Fukunaga, Toshinori 84
Funabiki, Nobuo 40, 41, 44
Furukawa, Jun 35, 38

Galbraith, Steven D. 27
Gallais, Jean-François 80
Gallais, Jean-François 80
Galperin, Slava 42
Gascon, Hugo 71, 74
Gasparini, Simone 64
Gasti, Paolo 67
Gates, Christopher S. 110
Ge, Qian 59, 70, 74
Gebotys, Catherine H. 63, 74
Georgel, Pierre 64, 76
Gérard, Benoît 4, 69, 79, 85–87, 91
Gerling, Sebastian 110
Ghadafi, Essam 47
Ghosh, Animikh 70
Giacinto, Giorgio 71
Gibler, Clint 110
Gibson, Katherine 64, 74
Giuffrida, Cristiano 73, 74
Goldberg, Ian 67, 114, 115, 122, 125, 131, 132
Gong, Xiaoqing 67, 74
Gong, Xun 116
Google 110
Goswami, Dibyendusekhar 64, 74, 76
Greenstadt, Rachel 67
Grewal, Gurleen 45
Griesmayr, Simone 4, 69, 76, 113
Grosinger, Jasmin 5
Gross, Hannes 5
Groth, Jens 42
Gruss, Daniel 5, 69, 70, 73, 74, 77, 80, 116
Gu, Xiaodan 68, 74, 115
Guillaume-Sage, L. 65, 74
Guilley, Sylvain 63
Gullasch, David 80
Gunter, Carl A. 3, 59, 69, 72, 74, 75, 116, 117, 122, 134
Günther, Felix 2, 42, 54
Guo, Yao 63, 69
Gupta, Haritabh 71
Gürses, Fahriye Seda 10

Ha, Elizabeth 76, 110, 134
Hall, Chris 80
Hamada, Naoto 41
Hammer, Christian 110
Han, Jun 70, 71, 74, 76, 110
Hanaoka, Goichiro 40, 42
Haney, Ariel 76, 110, 134
Hanna, Steve 110, 134
Hänsch, Ronny 71
Hanser, Christian 45
Hao, Feng 70, 71
Haralambiev, Kristiyan 36, 38
Hayashi, Takuya 41
Hayes, Jamie 123
Hazay, Carmit 15
He, Dongjing 3, 59, 69, 72, 74, 75, 116, 117, 122, 134
He, Gaofeng 68, 74, 115
He, Wenbo 67
Heiderich, Mario 116
Hein, Daniel M. 45
Heinly, Jared 64
Heinz, Benedikt 69, 80
Heiser, Gernot 59, 70, 74
Hemminki, Samuli 72
Henze, Martin 67
Herrmann, Dominik 132
Hess, Florian 27
Heyszl, Johann 69
Hintz, Andrew 114, 115
Hira, Yuta 41, 44
Ho, Bo-Jhang 72
Ho, Simon 63, 74
Hölbl, Marko 5
Hollick, Matthias 67, 74
Holz, Thorsten 71, 76
Hosseini, Henry 71
Hsu, Ruei-Hau 40, 41, 44
Hu, Shi 45

Huang, Ling 115
Huang, Qiong 23
Hubain, Charles 63, 74
Hupperich, Thomas 71, 76
Hussain, Muzammil 71
Hutter, Michael 66, 74
Hwang, Jung Yeon 20–22, 29–33, 42, 44–46

i Rotger, Llorenç Huguet 45
Iijima, Kengo 69
International Organization for Standardization (ISO) 42, 45
Intwala, Mehool 72, 74
Isern-Deyà, Andreu Pere 1, 45

Jaffe, Joshua 58, 63
Jakobsson, Markus 98
Jana, Suman 3, 61, 68, 74, 116, 118, 131, 132
Janc, Artur 116
Jao, David 45
Jawurek, Marek 23
Jensen, David 115
Jin, Xiaocong 64, 76
Johnson, Rob 67, 76, 114, 115, 122, 125, 131–134
Joseph, Anthony D. 115
Joshi, Brijesh 67, 115, 132
Juárez, Marc 67
Jun, Benjamin 58, 63
Jung, Jaeyeon 110, 134

Kapadia, Apu 72, 74
Katzenbeisser, Stefan 67, 74
Kaur, Jasleen 67
KeepSafe 99
Kelley, Patrick Gage 110, 134
Kelsey, John 80
Kemerlis, Vasileios P. 70, 116
Kemme, Bettina 67
Keromytis, Angelos D. 70, 116
Kerschbaum, Florian 23
Kiah, M. L. Mat 71
Kiayias, Aggelos 1, 10, 17, 35, 38
Kiefer, Franziskus 2, 42, 54
Kim, Jeremie 73
Kim, Yoongu 73
Kiyavash, Negar 116
Kiyomoto, Shinsaku 63
Kizhvatov, Ilya 80
Klapper, Patrick 67, 74
Koç, Çetin Kaya 80
Kocher, Paul C. 2, 58, 63, 80
Koga, Satoshi 51
Kohlweiss, Markulf 15
Kohno, Tadayoshi 110
Korak, Thomas 4, 57, 69, 76, 113
Krenn, Stephan 80
Krissler, Jan 71
Krombholz, Katharina 76
Kubooka, Fumiaki 41
Kührer, Marc 71
Kumar, Rajesh 64, 76
Kumar, Santosh 70
Kumar, Vireshwar 40, 42, 44
Kune, Denis Foo 80
Kurtz, Andreas 71, 74
Kwon, Taekyoung 76

Lai, Konrad 73
Lanze, Fabian 67
Lee, Donghyuk 73
Lee, Gene Moo 67
Lee, Ji-Hye 73
Lee, Sokjoon 20–22, 29–33, 42, 44–46
Lee, Wenke 76, 133
Levine, Brian Neil 115, 129, 132
Li, He 40, 42, 44
Li, Mengyuan 68
Li, Ninghui 110
Li, Rong 67, 74
Li, Zhou 68
Liang, Bin 116
Liang, Xiaohui 68
Liao, Yong 67
Liardet, Pierre-Yvan 74
Liberatore, Marc 115, 129, 132
Libert, Benoît 35, 42
Lin, Dongdai 41
Lindell, Yehuda 15
Lindorfer, Martina 73, 74

Ling, Zhen 64
Lipp, Moritz 5, 70, 74, 77, 80
Lipton, Richard J. 65
Lisart, Mathieu 74
Liu, Alex X. 67
Liu, Benyuan 64
Liu, Debin 98
Liu, Junyi 68
Liu, Liangkun 116
Liu, Xiangyu 68, 71
Liu, Yao 68
Lomné, Victor 75
Longa, Patrick 45
Lu, Liming 115
Luo, Junzhou 64, 68, 74, 115
Luo, Xiapu 76, 133
Lysyanskaya, Anna 29, 41

Ma, Yuanyuan 68, 74, 115
Macario-Rat, Gilles 63, 74
Machiraju, Sridhar 70
Maffei, Matteo 110
Maggi, Federico 64
Maiorca, Davide 71
Malpani, Ambarish 42
Mancini, Luigi Vincenzo 67
Mangard, Stefan 2, 4, 5, 57–59, 69, 70, 73–77, 80, 113, 116
Manulis, Mark 2, 40–42, 44, 54
Mao, Bing 70
Mao, Zhuoqing Morley 68
Marson, Giorgia Azzurra 15
Martin, Paul D. 72
Martinovic, Ivan 67, 74
Mascolo, Cecilia 71
Mathewson, Nick 118
MathWorks 102
Maurice, Clémentine 5, 70, 73, 74, 77, 80
Maurine, P. 65, 74
Maurine, Philippe 74
McDaniel, Patrick Drew 58, 76
Mehrnezhad, Maryam 70, 71
Mei, Hong 63, 69
Menarini, Federico 66
Menezes, Alfred 11
Meng, Yan 68
Micciancio, Daniele 1, 10
Michalevsky, Yan 71, 72, 74
Michiels, Wil 63, 74
Miller, Brad 115
Miller, Katja 69
Miluzzo, Emiliano 70, 94, 102
Miskovic, Stanislav 67
Miyaji, Atsuko 40
Miyake, Yutaka 63
Miyauchi, Hiroshi 80
Mohamed, Manar 77
Molloy, Ian 110
Monrose, Fabian 64, 74, 76, 133
Moonsamy, Veelasha 4, 57
Moshchuk, Alexander 110
Mossop, Evan 64, 74
Mu, Yi 21, 41
Muehlmann, Ulrich 5
Müller, Tilo 66
Mune, Cristofaro 66, 74
Mutlu, Onur 73
Myers, Michael 42

Na, Sarang 76
NAB 99
Nahapetian, Ani 71
Nahrstedt, Klara 3, 59, 69, 72, 74, 75, 116, 117, 122, 134
Najm, Zakaria 65
Nakanishi, Toru 20, 35, 38, 40, 41, 44
Nakano, Yuto 63
Nakibly, Gabi 71, 72, 74
Naor, Moni 22, 27, 33
Narain, Sashank 71
Narayanan, Arvind 2
Nati, Michele 67, 74
National Institute of Standards and Technology (NIST) 81
Naveed, Muhammad 3, 59, 62, 69, 72, 74, 75, 77, 116, 117, 122, 134, 138
Nawaz, Sarfraz 71
Nelakuditi, Srihari 71, 74
Neve, Michael 80, 82, 85, 89, 90
NewAE Technology Inc. 65
Nguyen, Lan 27, 42, 45

Nguyen, Le T. 71, 74
Nguyen, Robert 63
Niessen, Lukas 67, 115, 132
Nita-Rotaru, Cristina 110
Nithyanand, Rishab 76, 114, 122, 125, 131–134
Noubir, Guevara 71
Nurmi, Petteri 72
Nyang, DaeHun 20–22, 29–33, 42, 44–46

O'Flynn, Colin 65
Ohkubo, Miyako 36, 38
Oikonomou, George C. 64
Olejnik, Lukasz 116
Omote, Kazumasa 40
Ordas, S. 65, 74
Ordas, Thomas 74
Oren, Yossef 70, 116
Orlandi, Claudio 23
Osvik, Dag Arne 70, 80, 84
Oswald, Elisabeth 2, 58, 59, 75
Owusu, Emmanuel 70, 71, 74, 76, 110

Padmanabhan, Venkata N. 114, 115
Page, Dan 80
Pagh, Rasmus 50
Pan, Xiaorui 3, 59, 69, 72, 74, 75, 116, 117, 122, 134
Panchenko, Andriy 67, 115, 132
Park, Jung-Min "Jerry" 40, 42, 44
Park, Kunsoo 35
Parno, Bryan 110
Payeras-Capellà, Magdalena 1, 45
Peng, Hao 110
Pennekamp, Jan 67
Perdisci, Roberto 76, 133
Pereira, Olivier 45
Perrig, Adrian 70, 71, 74, 76, 110
Peters, Thomas 42
Phoha, Vir V. 64, 76
Ping, Dan 70
Plos, Thomas 5, 69, 70, 80, 84
Poettering, Bertram 2, 42, 54
Pointcheval, David 22, 27–29, 33, 42, 45, 52
Ponemon Institute LLC 42
Popp, Thomas 2, 58, 59, 75
Potharaju, Rahul 110
Potzmader, Klaus 45
Preibusch, Sören 98
Preneel, Bart 121
Puigserver, Macià Mut 1, 45

Qi, Yuan 110
Qian, Zhiyun 68
Qiu, Lili 114, 115
Quisquater, Jean-Jacques 58

Rackoff, Charles 22, 33
Raguram, Rahul 64, 74, 76
Raij, Andrew 70
Rauzy, Pablo 65
Razavi, Kaveh 73, 74
Ren, Kui 64
Renauld, Mathieu 85, 86, 91
Rieck, Konrad 71, 74
Ristenpart, Thomas 76, 133
Rivière, Lionel 65
Roche, Thomas 75
Rodler, Flemming Friche 50
Roesner, Franziska 110
Roscian, Cyril 66
Rotundo, Enrico 67, 74
Roy, Nirupam 71, 74
Ruan, Na 68
Russell, Wilf 114, 115

Sadeh, Norman M. 110, 134
Safavi-Naini, Reihaneh 27, 42, 45
Sahai, Amit 42
Saito, Teruo 80
Sakai, Yusuke 40, 42
Sako, Kazue 35, 38
Sakurai, Kouichi 51
Samsung 97, 99
Samyde, David 58
Sanatinia, Amirali 71
Sanders, Olivier 29
Sanfelix, Eloi 66, 74
Santesson, Stefan 42

Sapp, Benjamin 70, 94, 98, 102, 105, 109–111
Sarafianos, Alexandre 66
Saraswat, Vishal 80
Sarma, Bhaskar Pratim 110
Sauvage, Laurent 63, 65
Saxena, Nitesh 77
Schear, Nabil 116
Schlegel, Roman 72, 74
Schmidt, Jörn-Marc 5, 66, 74
Schmitt, Jens B. 67
Schneider, Michael A. 116
Schneier, Bruce 80
Schulman, Aaron 72
Schulz, Matthias 67, 74
Seaborn, Mark 73
Seifert, Jean-Pierre 80, 90
Serwadda, Abdul 64, 76
Sethumadhavan, Simha 70, 116
Seys, Stefaan 121
Shacham, Hovav 12, 22, 26, 29, 37, 38, 40, 41, 44
Shahandashti, Siamak Fayyaz 70, 71
Shahzad, Muhammad 67
Shamir, Adi 15, 17, 18, 65, 70, 80, 84
Shi, Haixia 1, 10, 15–17, 30
Shi, Wenchang 116
Shigeri, Maki 80
Shmatikov, Vitaly 2, 3, 61, 68, 74, 116, 118, 131, 132
Shoup, Victor 28
Shrestha, Prakash 77
Shrimpton, Thomas 76, 133
Shukla, Diksha 64, 76
Sigl, Georg 69
Simon, Daniel R. 22, 33, 114, 115
Simon, Laurent 68, 71, 98, 99, 101, 106, 111
Skorobogatov, Sergei 66, 74
Skorobogatov, Sergei P. 66, 74
Slamanig, Daniel 3–5, 9, 19, 39, 51
Smith, Jonathan M. 64, 70, 74, 94, 98, 102, 105, 109–111
Smith, Matthew 58, 76
Song, Dawn 110, 134
Song, Dawn Xiaodong 41
Souissi, Youssef 63
Spolaor, Riccardo 67, 74
Spreitzenbarth, Michael 66
Spreitzer, Raphael 3–5, 9, 19, 39, 51, 57, 69, 70, 74, 76, 77, 79, 80, 84, 93, 113
Srivastava, Mani B. 70, 72
Stadler, Markus 10, 17
Standaert, François-Xavier 85–87, 91
Stern, Jacques 41, 44
Stöber, Tim 67
Stumpf, Frederic 69, 80
Sugiyama, Yuji 41
Sun, Jingchao 64, 76
Sun, Qixiang 114, 115
Sun, Xin 70
Sural, Shamik 71
Susilo, Willy 21, 41, 42
Suzaki, Kuniyasu 69
Suzaki, Tomoyasu 80
Swaminathan, Prashanth 72
Syverson, Paul F. 118

Takahashi, Junko 84
Tan, Chik How 23
Tang, Qiang 23, 24
Tang, Zhanyong 67, 74
Tarkoma, Sasu 72
Taylor, Vincent F. 67, 74
Teranishi, Isamu 35, 38
Teufl, Peter 45
Teuwen, Philippe 63, 74
Tews, Erik 67, 74
Thillard, Adrian 75
Tibouchi, Mehdi 63, 74
Tiu, C. C. 63, 74
Tobich, Karim 74
Toreini, Ehsan 70, 71
Traoré, Jacques 45
Tria, Assia 66
Tromer, Eran 70, 80, 84
Tryfonas, Theo 64
Tsang, Patrick P. 41
Tsiounis, Yiannis 35, 38
Tsudik, Gene 41

Tsunoo, Yukiyasu 80
Tunstall, Michael 80
Tygar, J. D. 115

UK Office of Communications 98
Unruh, Dominique 64
Unterluggauer, Thomas 3, 4, 9, 19, 39, 45

Vaidya, Jaideep 71
van der Veen, Victor 73, 74
van Heyst, Eugène 1, 10
van Woudenberg, Jasper G. J. 66
Varshavsky, Alexander 70, 94, 102
Veerapandian, Gunaa Arumugam 72
Venturi, Daniele 15
Vercauteren, Frederik 27
Verde, Nino Vincenzo 67
Verheul, Eric R. 43
Veyrat-Charvillon, Nicolas 85, 86, 91
Vigna, Giovanni 73, 74
VirusTotal 110
Vives-Guasch, Arnau 1, 45
Vo-Huu, Triet D. 71
von Styp-Rekowsky, Philipp 110

Wagner, David 76, 80, 110, 134
Wang, Helen J. 110
Wang, Qinglong 67
Wang, Rui 116
Wang, Tao 67, 114, 115, 122, 125, 131, 132
Wang, Wei 67
Wang, XiaoFeng 3, 59, 62, 69, 72, 74, 75, 77, 116, 117, 122, 134, 138
Wang, Yi-Min 114, 115
Wang, Zhenghong 80, 90
Warinschi, Bogdan 1, 10
Watanabe, Hajime 20, 35, 38
Watzinger, Hubert 5
Weggenmann, Benjamin 69
Wehrle, Klaus 67
Wei, Victor K. 20
Weiß, Michael 69, 80
Wendolsky, Rolf 132
Wenger, Erich 45
Wetherall, David 110, 134
White, Andrew M. 64, 74, 76
Wilkerson, Chris 73
Winter, Johannes 45
Witteman, Marc F. 66
Wong, Duncan S. 23
Wright, Charles V. 76, 133

Xia, Peng 64
Xiao, Yuan 70
Xing, Tianzhang 67, 74
Xu, Wenduan 68
Xu, Wenyuan 71, 74
Xu, Yi 64, 76
Xu, Zhi 70

Yagi, Toshiki 69
Yahyavi, Amir 67
Yampolskiy, Aleksandr 29, 36
Yan, Lin 63, 69
Yang, Guomin 23
Yang, Ming 68, 74, 115
Yang, Qing 67
Yang, Yaling 40, 42, 44
Yarom, Yuval 59, 70, 74, 80
Yildiz, Can 64
You, Wei 116
Yuan, Kan 59, 62, 77, 116, 134, 138
Yue, Qinggang 64
Yuen, Tsz Hon 42
Yung, Moti 1, 10, 17, 22, 27, 33, 35, 38, 42

Zaidan, A. A. 71
Zaidan, B. B. 71
Zankl, Andreas 69
Zhang, Chong 1, 10, 15–17, 30
Zhang, Jie 67, 74
Zhang, Jinxue 64, 76
Zhang, Joy 70, 71, 74, 76, 110
Zhang, Kehuan 68, 71, 72, 74, 116
Zhang, Nan 59, 62, 77, 116, 134, 138
Zhang, Rui 64, 76
Zhang, Xiaokuan 70
Zhang, Xin Cheng 67, 115, 132
Zhang, Yanchao 64, 76

Zhang, Yang 64
Zhang, Yinqian 70
Zhao, Wei 64
Zheng, Xiaolong 67, 74
Zheng, Yuliang 18
Zhou, Gang 67
Zhou, Peng 76, 133
Zhou, Sujing 41
Zhou, Xiao-yong 3, 59, 62, 69, 72, 74, 75, 77, 116, 117, 122, 134, 138
Zhou, Zhe 71
Zhu, Haojin 68
Zhu, Sencun 70
Zinnen, Andreas 67, 115, 132
Zoscher, Lukas 5

About the Author

Author information as of April 2017.

Personal Information

Name:	Raphael Spreitzer
Date of birth:	December 17, 1986
Place of birth:	Friesach, Austria.

Education

- **2013/01 – present:** Doctoral programme in Technical Sciences at Graz University of Technology, Austria
- **2012/10/25:** Graduated to Dipl.-Ing. (corresponds to MSc) with distinction at Graz University of Technology, Austria
- **2011 – 2012:** Master's programme in Software Development and Business Management at Graz University of Technology, Austria.
- **2007 – 2011:** Bachelor's programme in Software Development and Business Management at Graz University of Technology, Austria

Professional and Academic Experience

- **08/2012 – present:** Research assistant at the Institute for Applied Information Prcessing and Communications (IAIK), Graz University of Technology, Austria.
- **Summer 2010:** Software developer, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Austria.
- **Summer 2009:** Software developer, FERK Systems, Graz, Austria.
- **Summer 2008:** Software developer, ITS Immobilien Treuhand Software GmbH, Graz, Austria.
- **02/2007 – 11/2007:** Software developer, GTL-Data GmbH, Klagenfurt, Austria.

Author's Publications

Author's publications as of April 2017 mapped to the corresponding chapters.

Chapter 2

- Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security – ISC 2014*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014
- Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group Signatures with Linking-Based Revocation: A Pragmatic Approach for Efficient Revocation Checks. In *Conference on Cryptology & Malicious Security – Mycrypt 2016*, 2016. In press

Chapter 3

- Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Adding Controllable Linkability to Pairing-Based Group Signatures for Free. In *Information Security – ISC 2014*, volume 8783 of *LNCS*, pages 388–400. Springer, 2014

Chapter 4

- Daniel Slamanig, Raphael Spreitzer, and Thomas Unterluggauer. Group Signatures with Linking-Based Revocation: A Pragmatic Approach for Efficient Revocation Checks. In *Conference on Cryptology & Malicious Security – Mycrypt 2016*, 2016. In press

Chapter 5

- Raphael Spreitzer, Veelasha Moonsamy, Thomas Korak, and Stefan Mangard. Systematic Classification of Side-Channel Attacks: A Case Study for Mobile Devices. *arXiv ePrint Archive, Report 1611.03748*, 2016. In submission

Chapter 6

- Raphael Spreitzer and Benoît Gérard. Towards More Practical Time-Driven Cache Attacks. In *Information Security Theory and Practice – WISTP 2014*, volume 8501 of *LNCS*, pages 24–39. Springer, 2014

Chapter 7

- Raphael Spreitzer. PIN Skimming: Exploiting the Ambient-Light Sensor in Mobile Devices. In *Security and Privacy in Smartphones & Mobile Devices – SPSM@CCS*, pages 51–62. ACM, 2014

Chapter 8

- Raphael Spreitzer, Simone Griesmayr, Thomas Korak, and Stefan Mangard. Exploiting Data-Usage Statistics for Website Fingerprinting Attacks on Android. In *Security and Privacy in Wireless and Mobile Networks – WISEC 2016*, pages 49–60. ACM, 2016

Further Contributions

Conference Publications

- Moritz Lipp, Daniel Gruss, Raphael Spreitzer, Clémentine Maurice, and Stefan Mangard. ARMageddon: Cache Attacks on Mobile Devices. In *USENIX Security Symposium 2016*, pages 549–564. USENIX Association, 2016

- Olivier Blazy, David Derler, Daniel Slamanig, and Raphael Spreitzer. Non-Interactive Plaintext (In-)Equality Proofs and Group Signatures with Verifiable Controllable Linkability. In *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *LNCS*, pages 127–143. Springer, 2016

- Lukas Zoscher, Jasmin Grosinger, Raphael Spreitzer, Ulrich Muehlmann, Hannes Gross, and Wolfgang Bösch. Concept for a Security Aware Automatic Fare Collection System Using HF/UHF Dual Band RFID Transponders. In *European Solid State Device Research Conference – ESSDERC 2015*, pages 194–197. IEEE, 2015

- Hannes Gross, Marko Hölbl, Daniel Slamanig, and Raphael Spreitzer. Privacy-Aware Authentication in the Internet of Things. In *Cryptology and Network Security – CANS 2015*, volume 9476 of *LNCS*, pages 32–39. Springer, 2015

- Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. Cache Template Attacks: Automating Attacks on Inclusive Last-Level Caches. In *USENIX Security Symposium 2015*, pages 897–912. USENIX Association, 2015

- Raphael Spreitzer and Jörn-Marc Schmidt. Group-Signature Schemes on Constrained Devices: The Gap Between Theory and Practice. In *Cryptography and Security in Computing Systems – CS2@HiPEAC*, pages 31–36. ACM, 2014

- Raphael Spreitzer and Thomas Plos. Cache-Access Pattern Attack on Disaligned AES T-Tables. In *Constructive Side-Channel Analysis and Secure Design – COSADE 2013*, volume 7864 of *LNCS*, pages 200–214. Springer, 2013

- Raphael Spreitzer and Thomas Plos. On the Applicability of Time-Driven Cache Attacks on Mobile Devices. In *Network and System Security – NSS 2013*, volume 7873 of *LNCS*, pages 656–662. Springer, 2013

Journal Publications

- Lukas Zoscher, Raphael Spreitzer, Hannes Gross, Jasmin Grosinger, Ulrich Muehlmann, Dominik Amschl, Hubert Watzinger, and Wolfgang Bösch. HF/UHF Dual Band RFID Transponders for an Information-Driven Public Transportation System. *Elektrotechnik und Informationstechnik*, 133:163–175, 2016

Deutsche Fassung:
Beschluss der Curricula-Kommission für Bachelor-, Master- und Diplomstudien vom 10.11.2008
Genehmigung des Senates am 1.12.2008

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Graz, am …………………………… ………………………………………………..
(Unterschrift)

Englische Fassung:

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…………………………… ………………………………………………..
date (signature)