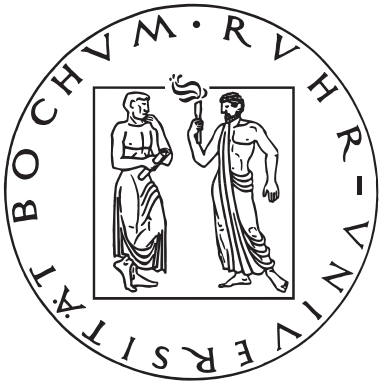
Assessment and Prevention of Side-Channel Leakage with Special Consideration of the Multivariate Case



Dissertation

---

zur Erlangung des Grades eines Doktor-Ingenieurs
der Fakultät für Elektrotechnik und Informationstechnik
an der Ruhr-Universität Bochum

---

by Florian Bache
Bochum, December 2021

Für Larissa und Lilo.

Florian Bache
Author's contact information:
www.seceng.rub.de

Thesis Advisor:	**Prof. Dr.-Ing. Tim Güneysu** Ruhr-Universität Bochum, Germany
Secondary Referee:	**Prof. Dr.-Ing. Thomas Eisenbarth** Universität zu Lübeck, Germany
Thesis submitted:	December 22, 2021
Thesis defense:	February, 21, 2022
Last revision:	March 26, 2023

Abstract

In the current age of the Internet of Things tens of billions of connected devices are spread over the world. They perform a plethora of different functions, from switching light bulbs over enabling cashless payments to controlling power plants. Many of these devices handle sensitive data or perform control critical systems and must therefore be protected using cryptographic protocols. Modern cryptographic algorithms used in these protocols have been studied by many experts for multiple years and deemed to provide excellent resistance against cryptoanalytical attacks. However, countless security critical devices are no longer hidden in computing centers behind strong doors but are potentially physically accessible to attackers. This access to a device allows the observance of unintentional information leakage in the form of side-channels, such as the power consumption or electromagnetic emanation. As the this information depends on the data physically processed by a device, side-channel attacks can extract secrets without having to break the underlying cryptographic algorithm. This thesis addresses this thread by developing methods to assess and prevent the side-channel leakage of cryptographic implementations and is divided into three parts.

The first part analyzes methods to assess the side-channel leakage of devices which execute cryptographic functions. A novel, state of the art measurement system is developed that allows the rapid collection of side-channel data. In traditional leakage assessment this data is typically used in statistical tests, such as Welch's t-test, in order to decide if the the side-channel is influenced by the data processed by the target device and can therefore potentially be exploited. In order to address multiple downsides of this approach these issues, we introduce a new assessment framework around confidence intervals. Subsequently, we extend this framework to the multivariate setting allowing the assessment of advanced threads. As the computational complexity is exponential in the attack order, we describe and compare multiple techniques to speedup this evaluation while retaining maximal assessment confidence.

The second part of this thesis studies how essential components of cryptographic implementations can be protected against side-channel attacks. First, we develop a efficent hardware implementation for a core part of many lattice-based post-quantum secure key encapsulation mechanisms: a masked comparison of two polynomials. Secondly, several protected hardware architectures for addition circuits are developed and compared. These circuits form an essential part of multiple cryptographic implementations such as Addition-Rotation-XOR (ARX)-ciphers and several post-quantum secure schemes. In our work, we can achieve resistance against arbitrarily chosen attack orders using provably secure gadgets.

In the third and final part of this thesis novel methods for the side-channel protection of complex integrated systems are developed. First, we show how a secure processor for ARX-ciphers can be designed and implemented. Our prototype can be programmed to execute arbitrary ARX-algorithms while automatically providing security against timing and first-order side-channel attacks. Finally, we present a novel automated procedure for protecting software im-

plementations of cryptographic algorithms on of the shelf microcontrollers against side-channel attacks.

Keywords.

Cryptography, Masking, Side-Channel Analysis, Leakage Assessment, Embedded Security

Kurzfassung

Bewertung und Vermeidung von Seitenkanallecks unter besonderer Betrachtung des multivariaten Falls.

Im heutigen Zeitalter des Internets der Dinge sind mehrere zehn Milliarden verbundene Geräte über die Erde verteilt. Diese führen eine Unmenge verschiedener Funktionen aus: vom Schalten von Lampen über das bargeldlose Bezahlen bis zur Steuerung von Kraftwerken. Viele dieser Geräte arbeiten mit sensitiven Daten oder kontrollieren kritische Systeme und müssen deshalb mittels kryptpographischer Protokolle geschützt werden. Moderne kryptograpische Algorithmen, die in diesen Protokollen Verwendung finden, werden seit vielen Jahren von einer großen Anzahl Experten untersucht und bieten exzellente Sicherheit gegen kryptoanalytische Angriffe. Jedoch sind unzählige sicherheitskritische Geräte nicht länger in Rechenzentren hinter schweren Türen versteckt, sondern potentiell physikalisch für Angreifer erreichbar. Zugriff auf diese Geräte erlaubt die Beobachtung von unbeabsichtigten Informationslecks in Form von Seitenkanälen wie dem Energieverbrauch oder elektromagnetischer Abstrahlung. Da diese Informationen von jenen Daten, welche physikalisch von einem Gerät verarbeitet werden, abhängen, können Seitenkanalangriffe Geheimnisse extrahieren, ohne die kryptographischen Algorithmen selbst brechen zu müssen. Die vorliegende Dissertation befasst sich mit dieser Bedrohung, indem Methoden zur Beurteilung (Leakage Assessment) und Vermeidung (Masking) von Seitenkanallecks von kryptograpischen Implementierungen entwickelt werden. Die Forschungsbeiträge können in drei Teile gegliedert werden:

Im ersten Teil werden Methoden zur Beurteilung von Seitenkanalsignalen erörtert. Ein Messsystem, welches das effiziente und schnelle Sammeln von Seitenkanalsignalen ermöglicht, wird vorgestellt. Traditionelles Leakage Assessment nutzt diese Daten, um mittels statistischer Tests wie dem Welch-Test zu entscheiden, ob der Seitenkanal von den vom Zielgerät verarbeiteten Daten abhängt und somit potentiell für einen Angriff genutzt werden kann. Wir führen ein neues, auf Konfidenzintervallen basiertes Assessment Modell ein, welches die vielen Probleme der testbasierten Verfahren adressiert. Nachfolgend erweitern wir dieses Modell für das multivariate Szenario, was die Beurteilung der Sicherheit gegen hochentwickelte Angriffe ermöglicht. Da die Berechnungskomplexität exponentiell in der Ordnung des Angriffs ist, diskutieren wir mehrere Verfahren zur Beschleunigung der Evaluation.

Der zweite Teil dieser Dissertation befasst sich mit dem Schutz von essentiellen Komponenten kryptographischer Implementierungen gegen Seitenkanalangriffe. Zunächst entwickeln wir eine effiziente Hardware-Architektur für einen wesentlichen Teil vieler gitterbasierter post-quantensicherer Key Encapsulation Mechanismen: Einen für hohe Ordnungen maskierten Vergleich zweier Polynome. Als zweites konzipieren und vergleichen wir mehrere maskierte Additonsschaltkreise. Diese spielen eine Schlüsselrolle bei vielen kryptographischen Implementierungen wie ARX-Chiffren und bei mehreren post-quanten-sicheren Schemata. Durch den Einsatz beweisbar sicherer Bausteine erreichen wir hierbei Resistenz gegen beliebig gewählte Angriffsordnung.

Im dritten und letzten Teil dieser Arbeit werden neue Methoden zum Schutz komplexer, integrierter Systeme entwickelt. Als erstes entwerfen und realisieren wir einen sicheren Prozessor für ARX-Chiffren. Der entstandene Prototyp kann beliebige ARX-Algorithmen ausführen und bietet intrinsischen Schutz gegen Timingangriffe und Seitenkanalangriffe erster Ordnung. Schließlich stellen wir eine neue Methode zum automatisierten Schutz kryptographischer Algorithmen auf industriellen Mikrokontrollern vor.

Schlagworte.

Kryptographie, Masking, Seitenkanalanalyse, Leakage Assessment, Eingebettete Sicherheit

Table of Contents

Imprint v
Abstract v
Kurzfassung viii

I Preliminaries **1**

1 Introduction **3**
1.1 Motivation 3
1.2 Summary of Research Contributions and Outline 5

2 Physical Side-Channel Attacks on Cryptography **9**
2.1 Side-Channel Attacks 9
2.1.1 Simple Power Analysis (SPA) 9
2.1.2 Differential Power Analysis (DPA) 10
2.1.3 Correlation Power Analysis (CPA) 10
2.2 Countermeasures 11
2.2.1 Hiding 11
2.2.2 Masking 11
2.2.3 Threshold Implementation (TI) 12
2.2.4 Domain-Oriented Masking (DOM) 13
2.2.5 Probe-Isolating Non-Interference and Hardware Private Circuits 13
2.3 Side-Channel Leakage Assessment 13
2.3.1 Test Vector Leakage Assessment (TVLA) using Welch's t-Test 14
2.3.2 Confidence Intervals 15

II Evaluation of the Side-Channel Security of Cryptographic Implementations **17**

3 Confident Leakage Assessment **19**
3.1 Introduction 19
3.2 Confidence-Interval-based Leakage Detection 21
3.2.1 From Statistical Tests to Confidence Intervals 21
3.2.2 Family-wise Error Rate Correction 24
3.2.3 Higher-Order Moments 25
3.2.4 Efficient Implementation 26

3.3 Side-Channel Evaluation Workflow . . . 26
3.3.1 Measurement Parameters . . . 28
3.3.2 Data Acquisition . . . 30
3.4 Evaluation Results . . . 31
3.5 The Influence of Noise . . . 33
3.6 Signal to Noise Ratio . . . 34
3.7 Pre-Evaluation Checks . . . 35
3.8 Conclusion . . . 38

4 Multivariate Leakage Assessment 39
4.1 Introduction . . . 39
4.1.1 Contribution . . . 40
4.1.2 Related Work . . . 40
4.2 Preliminaries . . . 41
4.2.1 Notation . . . 41
4.3 Multivariate Leakage Assessment . . . 41
4.3.1 Efficient Computation . . . 43
4.4 Case Study: Sequential PRESENT . . . 47
4.4.1 Target Architecture . . . 47
4.4.2 Measurement Setup . . . 47
4.4.3 Parameter Space Exploration and Results . . . 48
4.4.4 Discussion . . . 50
4.5 Case Study: AES-DOM . . . 51
4.5.1 Target Architecture . . . 53
4.5.2 Results and Discussion . . . 54
4.6 Performance Evaluation. . . . 54
4.7 Conclusion . . . 55

III Side-Channel Resistance for Cryptographic Primitives 57

5 High-Speed Masking for Polynomial Comparison in Lattice-based KEMs 59
5.1 Introduction . . . 60
5.1.1 Related Work . . . 60
5.1.2 Contribution . . . 61
5.1.3 Outline . . . 61
5.2 Preliminaries . . . 61
5.2.1 Notation . . . 62
5.2.2 The Basic `NewHope` Scheme . . . 62
5.2.3 Fujisaki-Okamoto Transform as Applied to `NewHope` . . . 63
5.2.4 Security against Side Channel Attacks . . . 63
5.2.5 Arithmetic and Boolean Masking . . . 65
5.3 Higher-Order Masking of Comparison of Polynomials . . . 65
5.3.1 Evaluation of Previous Approaches . . . 66

5.3.2 Our Proposal . . . 66
5.3.3 Application to NewHope and Other Schemes . . . 72
5.4 Microcontroller Implementation . . . 72
5.4.1 Microcontroller Implementation . . . 72
5.4.2 Side-channel Measurements . . . 74
5.5 Results . . . 75
5.5.1 Performance Evaluation . . . 75
5.5.2 Randomness Consumption . . . 78
5.5.3 Leakage Evaluation . . . 78
5.6 Conclusions and Future Work . . . 79

6 Masking Addition with Boolean Shares 83
6.1 Introduction . . . 83
6.1.1 Contribution . . . 84
6.2 Preliminaries . . . 84
6.2.1 Notation . . . 84
6.2.2 Parallel Prefix Adders . . . 84
6.3 Boolean Masking for Addition Circuits . . . 85
6.3.1 Kogge-Stone Adder (KSA) . . . 86
6.3.2 Sklansky Adder (SA) . . . 87
6.3.3 Brent-Kung Adder (BKA) . . . 89
6.4 Implementation Results and Discussion . . . 91
6.4.1 TI Implementations . . . 91
6.4.2 HPC2 Implementations . . . 92
6.5 Conclusion . . . 92

IV Side-Channel Resistance for Integrated Systems 93

7 A Side-Channel Protected Processor for ARX-based Cryptography 95
7.1 Introduction . . . 95
7.1.1 Contribution . . . 96
7.1.2 Related work . . . 96
7.2 Preliminaries . . . 96
7.2.1 ARX Algorithms . . . 96
7.3 Design Considerations and Technical Description . . . 97
7.3.1 Protected ARX-Specific ALU . . . 98
7.3.2 Auxiliary General-Purpose ALU . . . 99
7.3.3 Data and Control Flow . . . 99
7.3.4 Memory Configuration . . . 100
7.3.5 Data Separation . . . 100
7.4 Implementation . . . 101
7.4.1 Hardware Instantiation of the SPARX Processor . . . 101
7.4.2 SPARX-Compliant Cipher Implementations . . . 101

7.5 Evaluation . . . 102
7.5.1 Leakage . . . 102
7.5.2 Performance and Size . . . 102
7.6 Conclusion . . . 104

8 Automated Masking of Software Implementations on Industrial Microcontrollers 105
8.1 Introduction . . . 105
8.1.1 Contribution . . . 106
8.1.2 Related work . . . 106
8.2 Concept . . . 107
8.3 Security in Practice . . . 108
8.4 Gadget Design . . . 109
8.5 Code Transformation . . . 110
8.5.1 Parsing and Tainting . . . 110
8.5.2 Instruction Replacement . . . 111
8.5.3 Security of Transformations . . . 112
8.6 Case Study . . . 112
8.6.1 Security Evaluation . . . 113
8.6.2 Performance . . . 114
8.7 Conclusion . . . 115

V Conclusion 117

9 Conclusion 119
9.1 Conclusion . . . 119
9.2 Related Research Areas and Future Work . . . 120

VI Appendix 123

A Supplementary Material 125
A.1 Moment Estimation using Histograms. . . . 125
A.2 Evaluation Results for DOM-AES. . . . 126
A.3 Subroutines of A2B Conversion . . . 127

Bibliography 127

List of Abbreviations 141

List of Figures 143

List of Tables 146

About the Author 149

Publications and Academic Activities **150**

Part I

Preliminaries

Chapter 1

Introduction

In this chapter we introduce the context of this thesis by providing the relevant background information regarding side-channel attacks. We summarize our research contributions in this field and detail the structure of this thesis.

Contents of this Chapter

1.1 Motivation **3**

1.2 Summary of Research Contributions and Outline **5**

1.1 Motivation

In today's Digital Age, computers became an essential backbone of our society. In the 1950s and 60s, the first relay- and electron tube-based computers where displaced by faster and more reliable transistor machines. Since then and continuing through today, all computers and more generally all digital devices rely on transistors acting as digital switches on the physical level.

The development of the personal computer in the 1970s, the advent of the internet in the 1980s and 1990s and the modern smart phone in the 2000s continued the trend of increasing the number of computing devices worldwide. Today, laptops, smartphones and tablets are among the most obvious types of computers that most people interact with around the world. They enable real-time communication across the whole globe and provide a plethora of other functions like productivity assistance, entertainment, navigation or healthcare support to billions of users worldwide. These devices are supported by massive data centers consisting of thousands of servers that provide data storage, computation power and telecommunication to consumers and businesses. Additionally, billions of smaller computers are deployed every year as embedded devices, constituting central components of many modern products and systems such as TVs, kitchen equipment, smart cards, cars, HVAC systems or industrial machines. Depending on the concrete requirements of the respective application, such as performance, power consumption and cost, these embedded devices are realized as microcontrollers, programmable hardware such as Field Programmable Gate Arrays (FPGAs) or dedicated as dedicated Application Specific Integrated Circuits (ASICs). In recent years, the connectivity between embedded devices increased dramatically, eventually forming the Internet of Things (IoT). In the IoT, devices communicate directly between each other over the internet or an internet-like network without the direct involvement of a human. Examples for the application of IoT devices include home

automation products, wearable computers, smart traffic control, package tracking, health monitoring and industrial applications. In 2021, there were around 12.3 billion connected IoT devices worldwide [iot21] while several tens of billions of additional unconnected computing devices are shipped every year.

Many of today's embedded and IoT devices operate with sensitive data - from personal information in home automation systems to trade secrets in industrial environments. Additionally, these devices are also used to control critical systems like security doors, traffic or parts of power grids.

Therefore, the protection of the confidentiality and integrity of data processed by computing devices is essential. Cryptography provides an integral part in securing today's digital infrastructure by providing security primitives such as encryption, hash functions and signature schemes that can be used to construct security protocols allowing to achieve the desired security goals. Today, the overwhelming majority of security primitives is very well studied and considered to be secure by experts. While for many primitives, hard security proofs were not yet found, their study is a continuing, open and international process which built trust in their security over the past years. The potential attack surface on digital infrastructure is therefore primarily found on the implementation level, i.e. the concrete realization of a security algorithm in software and hardware, as well as on the integration level.

In this work, we focus on a subset of attack surfaces presented on the implementation level, namely passive physical side-channel attacks. This type of attack aims to extract secret information from devices by exploiting the physical realization of security functions and more fundamentally, the connection between information and its physical representation. The storage of, as well as any operation on a bit of information is associated with a physical effect. For example, in the early relay-based computers a binary information of 0 or 1 could be represented by a closed or opened relay, while the change of a bit of information could be connected to the opening or closing of that relay. In modern technology, such as Complementary Metal Oxide Semiconductor (CMOS), a conducting or non-conducting set of transistors and a current flow to the control inputs of these transistors can be associated with a bit of information and a change of that information, respectively. Therefore, if such a physical effect could be measured on cryptographically protected device, the related, potentially sensitive information could be extracted without the need to break any cryptographic primitives. While the high integration levels in modern technology nodes, combining millions to billions of transistors into a single chip, severely limit direct access to single transistors, the aggregated signals are still accessible, for example through the instantaneous power consumption or electromagnetic emanation of a device. These signals can be collected and essentially viewed as an information channel (side channel) available to an attacker in addition to the intended input and output channels of the target device. Using statistical tools, an attacker can then combine the gathered information to learn sensitive data processed by a device such as cryptographic keys, which can in turn be used to completely break the security of the target system and potentially of connected systems. Due to the pervasiveness of computers, especially embedded systems, and their increasing physical accessibility, e.g., in the IoT context, this attack surface keeps growing and presents a serious threat to many information technology systems.

In view of this severe threat of side-channel attacks towards information technological systems, researchers in academia and industry are developing side-channel countermeasures. These countermeasures try to either lower the measurable side-channel information, increase unrelated

noise in the signal or decouple the signal from the actual information that is being processed. While several of the proposed countermeasures can be proven to be secure (sometimes in constraint models of the physical reality), their correct implementation in actual devices is not trivial in most cases. For this reason, a side-channel assessment of security critical devices which requires physical measurements of the side channel is often mandatory in practice. One approach towards this assessment involves trying to apply every known attack on the target device. As, however, the number of known attacks increases each year, alternative evaluation methodologies such as TVLA were put forward. These methods aim to evaluate whether a measurable side channel does or does not depend on the processed data and can therefore potentially exclude entire classes of attacks.

More advanced (multivariate) attacks try to exploit the information that can be present when combining side-channel leakage from multiple points in time during the computation of the target device. As these attacks can potentially extract more information for the side-channel leakage they need to be considered from the countermeasure as well as the leakage assessment perspective.

1.2 Summary of Research Contributions and Outline

The research contributions in this thesis address problems in some of the fields introduced above. Specifically, we describe new solutions for the assessment of side-channel leakage, develop efficient protection approaches for cryptographic primitives and apply side-channel countermeasures to integrated systems. Most of the contributions presented in this thesis have been published in peer-reviewed conference proceedings or journals ([BSMG17, BPG18, BPW$^+$19, BPO$^+$20, ABB$^+$21], [BWSG], [BG22]). These publications are reflected in the chapters of this thesis but are rearranged in some cases when this improves readability. Additional contributions of the author in [SBO$^+$15, RSBG20] are not part of this thesis as they were out of in scope.

While the research contributions in this thesis are all related to the field of passive side-channel attacks, they can be sub-classified into three main areas. The first is related to improved methods to access the magnitude of leakage that is present in the implementation of cryptographic algorithms in embedded devices. Here we do not focus on applying or developing attacks but on determining the leakage level in a univariate and multivariate setting. In the second area we develop new countermeasures for cryptographic primitives, specifically an secure comparison in quantum computer resistant encryption schemes and modular addition. These countermeasures can then be used as building blocks to secure more complex integrated systems. The final area focuses on these systems and introduces a side-channel secure cryptographic hardware accelerator for ARX-ciphers and a automated masking tool for software implementations on commercial microcontrollers.

Evaluation of the Side-Channel Security of Cryptographic Implementations

In traditional TVLA, the collected side-channel information is typically used in statistical tests, such as Welch's t-test, in order to decide if the the side-channel is influenced by the data processed by the target device and can therefore potentially be exploited.

Chapter 3

In this chapter we introduce a new approach to TVLA that is based on confidence intervals. We discuss the downsides of the previously dominant TVLA scheme and describe how the application of confidence intervals can mitigate several severe ones. We portray a modern side-channel acquisition system and show how it can be integrated with the confidence interval-based approach to form an efficient evaluation framework. The work described in this chapter was published at DATE 2017 [BPG18] and in it - Information Technology 2019 [BPW+19]. The majority of contributions in this publication were performed by the author of this thesis except for the derivation of the confidence level for absolute differences, the proofs and the implementation of the measurement and evaluation system.

Chapter 4

The evaluation methodology put forward in the previously described chapter allows practical side-channel assessment in many cases. This univariate approach considers multiple points in time where side-channel leakage occurs separately and can therefore produce fast results without the need for excessive computational power. However, particularly in the context of protected implementations, this univariate assessment can not always sufficiently capture the relevant device leakage. This is especially problematic in software-based masking countermeasures where parts of secret values (shares) are processed at different points in time. To address this gap, an extension to the multivariate setting is presented in this chapter. Because of the high requirements regarding computational resources when analyzing the combined leakage of multiple samples, we discuss several approaches to reduce this complexity and analyze their impact on performance and validity of the evaluation results. The work described in this chapter was accepted for publishing at IEEE Transactions of Computers [BWSG]. The majority of contributions in this publication were performed by the author of this thesis.

Side-Channel Resistance for Cryptographic Primitives

As side-channel attacks represent an increasingly serious threat towards cryptographically protected devices, effective countermeasures are required. In this part of this thesis, we describe how important components of cryptographic algorithms can be protected in software and in hardware.

Chapter 5

Public-key cryptography classifies an important set of cryptographic algorithms that are used in multiple applications. For example, they enable security functions like digital signatures and encrypted email. Due to the potential threats of quantum computers towards the security of conventional public-key cryptography, Post-Quantum Cryptography (PQC) algorithms were and are being developed. These algorithms can (under certain mathematical assumptions) withstand attacks from quantum computers. However, without dedicated protection, their implementations can still be vulnerable to side-channel attacks. In this chapter, we propose a masking scheme for an essential component of a class of lattice-based PQC algorithms – a secure comparison – and discuss its implementation in software on a microcontroller. The work described in this chapter was published at CHES 2020 [BPO+20]. The author of this

thesis contributed the design of the proposed countermeasure the correctness proof, parts of the security proof, the leakage evaluation and part of the performance evaluation.

Chapter 6

Modular addition is an important part of several cryptographic algorithms, such as ARX ciphers, where additions are combined with boolean functions. As an addition lends itself to arithmetic masking while boolean operations can naturally be protected with boolean masking, securing such algorithms against side-channel attacks can be challenging. To address this issue, we study three different designs for addition circuits and analyze their suitability for boolean masking. We implement 32-bit variants of these designs using the TI masking scheme to achieve first-order security and HPC2 gadgets for arbitrary security order against multivariate attacks. The work described in this chapter was published at MDPI Applied Sciences [BG22]. The majority of contributions were performed by the author of this thesis.

Side-Channel Resistance for Integrated Systems

While the previous chapters showed how essential parts of cryptographic functions can be protected against side-channel attacks, in this part of this thesis we describe how complete systems can be designed to be resistant against side-channel attacks. To this end, we present a (hardware) solution for protected accelerator design and a software approach to automated masking.

Chapter 7

In this chapter, we design and implement a complete (co-)processor that can execute ARX-based cryptography. It realizes masking countermeasures that provide intrinsic resistance against timing attacks and first-order side-channel attacks for ARX-ciphers. The main advantage of this approach in comparison to traditional hardware implementations is the high flexibility, allowing to change the set of implemented algorithms after finalizing the hardware design. The work described in this chapter was published at CHES 2020 [BPO+20]. The author of this thesis performed the majority of contributions in this publication except for the leakage evaluation.

Chapter 8

The correct and secure realization of side-channel countermeasures is a complicated and error-prone task when implementing cryptographic functions on microcontrollers. A main reason for this problem lies in the fact that side-channel hardening needs to be repeated for every new cryptographic function a device should perform. Automatically applying countermeasures to implementations can help reducing the required skill, time and other resources required in this process. In this chapter, we develop an approach that can automatically generate side-channel protected code from unprotected code. The work described in this chapter was published at DATE 2021 [ABB+21]. The author of this thesis contributed the majority of the design of the countermeasure proposed in this publication.

Chapter 2

Physical Side-Channel Attacks on Cryptography

This section provides the background that is relevant to the research contributions presented in the following parts of this thesis. It introduces passive side-channel attacks, relevant countermeasures and explains the process of side-channel leakage assessment. Additional background information that is primarily relevant in specific chapters of this thesis is introduced in the respective chapter.

Contents of this Chapter

2.1 Side-Channel Attacks **9**
2.2 Countermeasures **11**
2.3 Side-Channel Leakage Assessment **13**

2.1 Side-Channel Attacks

Physical attacks impose serious threats towards all exposed hardware containing cryptographic functions such as smart cards, hardware security modules or IoT devices. These attacks target implementations of cryptographic algorithms and can corrupt the security of real-world systems [MS16] even if they rely on established ciphers such as AES, which are believed to be secure against cryptanalytic attacks. This is possible by exploiting additional side-channel information such as power consumption [KJJ99a], electromagnetic emanation [GMO01], thermal effects [HS13], timing behavior [Koc96], and acoustic [GST14] or photonic [SNK+12b] emissions, which traditional cryptanalysts do not have access to. Simple and differential attacks [KJJ99b] as well as their extensions and generalizations such as correlation [BCO04a], template [CRR03], mutual information [GBT07] or algebraic attacks [BKP08] all rely on the dependency between the data being processed by a device and the corresponding side-channel, allowing them to extract the most critical secret – the cryptographic key – from the implementation.

In this thesis we focus on the subgroup of passive side-channel attacks that exploit power consumption or electro-magnetic emanation generated by a device in order to reveal its secrets [KJJ99a, QS01]. We describe the two basic types of side-channel attacks, SPA and DPA, as well CPA below.

2.1.1 Simple Power Analysis (SPA)

Simple Power Analysis [KJJ99b] directly exploits the fact that different operations performed by a device and different data used by these operations influence the side-channel, e.g., power

consumption of the device. We illustrate how SPA works using an example: In the RSA decryption a simple square-and-multiply algorithm can be used for the exponentiation of the message with the private key. This algorithm sequentially scans every bit of the key and only performs a multiplication if the bit is equal to one. As the multiplication requires electrical power which is not needed if no multiplication is performed, the power and electromagnetic side channels directly depend on the secret key. In low-noise scenarios an SPA can be as simple as visually looking at the side-channel trace and seeing the key in binary. Even if the noise level is too high to break the implementation with a single trace, the target device can often be forced to perform multiple operations on the same data, allowing to reduce the noise by averaging.

2.1.2 Differential Power Analysis (DPA)

While an SPA can extract sensitive information from only a single or a small amount of side-channel traces, DPA [KJJ99b] generally requires several (from tens to millions) of traces for an evaluation. The large number of processed traces often enables successful attacks, even the dependency of the side channel from sensitive values is very small. In DPA, an attacker targets a single bit of the ciphertext, which depends on a part of the key and other known values. Then, several power measurements are collected during the encryption of multiple (unknown) plaintexts by the target device. For every possible value of the relevant sub-key the power traces are then assigned to two classes, depending on the targeted bit being either zero or one. If the sub-key bit was guessed correctly, the difference of the mean traces resulting from averaging each of the sets will contain a distinct peak, because the traces were grouped correctly in the previous step. For all other (incorrectly) guessed key bits the grouping was random and thus results in a net zero difference. This procedure can then be iterated over other key bits until an exhaustive search for the remaining bits is feasible. A similar (higher-order) attack can be performed by analyzing the variance differential instead of the mean differential. Similarly, a multivariate attack can be performed by pre-processing the traces in order to combine multiple samples and exploit their joint leakage.

2.1.3 Correlation Power Analysis (CPA)

Correlation Power Analysis (CPA) was proposed in [BCO04b] in order to alleviate some of the shortcomings of DPA such as implicit restriction on a hamming weight leakage model and the increased algorithmic noise due to only one bit being predicted at a time. The side-channel trace acquisition part of a CPA is the same as for the DPA, i.e., several traces are collected during the encryption of different inputs. In order to then perform the analysis, an intermediate value of the algorithm is selected that depends on a part of the key and the ciphertext. For every collected ciphertext and every possible value of the relevant part of the key, a hypothetical value for the selected intermediate is computed. Then, depending on the device under test, a leakage model is applied to these hypothetical intermediate value resulting in sets of hypothetical leakage values. The leakage model is usually one of Hamming weight (number of bits set to one), Hamming distance (Hamming weight of the bit-wise XOR between the current and previous value) or the identity function. Finally, for every Point of Interest (POI) of the power traces and every hypothesis for the relevant part of the key the correlation between the measured leakage and the hypothetical leakage is calculated, resulting in one correlation coefficient for every key hypothesis at every point of interest. If the attack was successful, the correct key candidate can

be identified as the one producing the highest correlation with the measurements over all POIs. Higher-order and multivariate attacks can be performed as in the DPA case.

2.2 Countermeasures

As the prevention of side-channel vulnerabilities is a critical part of a holistic security design, a multitude of countermeasures have been developed. These can generally be grouped into two classes: hiding [VMKS12] and masking [PR13, BNN+15b, RBN+15]. A combination of hiding and masking countermeasures can also be implemented in order to achieve practical security against higher-order attacks while keeping the overhead reasonable [MW15].

2.2.1 Hiding

The goal of hiding is to bury the information contained in the side-channel in noise. This is done by decreasing the Signal to Noise Ratio (SNR) in order to prevent an attacker from exploiting it. This can either be achieved by reducing the signal corresponding to the compromising information or by artificially increasing the noise an evaluator has to face. Signal reduction can be accomplished by using specialized logic styles like dual-rail pre-charge logic, e.g., Sense Amplifier Based Logic (SABL) [TV04a] and Wave Dynamic Differential Logic (WDDL) [TV04b]. These approaches try to cancel the data-dependency of the (power) side-channel signature by creating an inverted signal on-chip and thereby hiding the information from an external attacker. While these methods can increase the resistance of a device against side-channel attacks, they are not a perfect solution as manufacturing tolerances prevent constructing perfectly symmetrical logic gates.

A different hiding approach reduces the information that is available to an adversary by increasing the noise in the side-channel. This can be accomplished by randomizing the execution of cryptographic algorithms using random delays [CCD00], shuffling the order of operations [HOM06] or through dynamic reconfiguration of FPGAs [SMG17]. Additional noise can also be introduced into the side-channel by analog circuits on the die. Possible measures in this category include noise generators or artificial clock jitter. The pairing of other simultaneously active non-crypto cores, e.g., processors or peripherals, on the same die can also be considered a hiding countermeasure as this increases the overall noise an adversary needs to overcome when performing an attack.

All hiding countermeasures have in common, that they can be defeated by increasing the number of acquired measurements (e.g. power traces) of a side channel of the target device during operation. With enough traces, uncorrelated noise will average to zero when an attacker is trying to estimate the side-channel.

2.2.2 Masking

While hiding covers the relationship between the processed (sensitive) data and the side-channel in noise, masking countermeasures try to break the relation between sensitive data and processed values. This is achieved by splitting the intermediate values of an algorithm into (random) *shares* and processing them separately, thus forcing an adversary to collect information about all shares to reconstruct the secret. Masking schemes can be designed to provide protection

against attacks up to d-th order, i.e., asserting independence between any subset of up to d shares of intermediate values. While this directly implies resistance against an attacker with the capability to probe up to d wires in the circuit, the authors of [BDF+17] show that this d-probing security also results in a design that is secure in the more practical bounded moments leakage model, that we also focus on in this thesis. In this model masking countermeasures are able to disconnect sensitive values from some statistical moments of the side-channel leakage distribution. Therefore, in a first-order masking scheme there should be no relation between the sensitive values and the mean of the leakage distribution, while a second-order masking scheme must additionally remove the relation to the variance of the leakage. Note that, while higher-order attacks can defeat masking schemes that only protect against low-order attacks, the measurement complexity increases exponentially in the protection order [CJRR99].

To provide this level of security, masking schemes rely on certain assumptions. If these assumptions are violated, the level of security can be severely decreased. Therefore, it is important to implement a masked algorithm with particular care. Notably, glitches, which are temporary faulty states, are a problem for masked hardware circuits and render a straight-forward implementation of a masking scheme insecure [MPO05].

2.2.3 Threshold Implementation (TI)

A commonly used concept to achieve secure masking in the presence of glitches is TI. It provides provably first- and higher-order security for arbitrary functions and can be very efficient depending on the masked algorithm. Therefore, it has been (statically) applied to a wide variety of ciphers (e.g., [PMK+11, CBR+15, BDN+13]). In the following, we only briefly introduce the general concept of TI and refer the interested reader to the original publications [BNN+15a, RBN+15] for a more detailed description of the masking scheme.
A Threshold Implementation (TI) is a Boolean masking scheme, i.e., a sensitive value x is split into shares x^i such that $x = \bigoplus x^i$. The number of input (d_{in}) and output (d_{out}) shares of a function following the TI notion depends on the masking order and the algebraic degree of the function. However, we will mainly focus on the first-order secure variant here, where $d_{in} > d$ and $d_{out} > d$. While higher-order TI instantiations are possible [BGN+14], their security is limited to univariate attacks in practice[Rep15]. In order to correctly instantiate a threshold implementation, three properties have to be fulfilled. The *non-completeness* property requires any subset of output shares smaller than d to be independent of at least one input share. The *correctness* property guarantees that the output of a shared function can be unmasked yielding the result an unprotected realization of the function would produce, i.e., $\bigoplus f_{shared}^{(x^1,\ldots,x^{d_{in}})} = f_{unshared}(x)$. Finally, *uniformity* requires each possible output sharing to be equally likely when the input sharing is drawn from a uniform distribution. Note, that this last property is usually the hardest to fulfill and frequently requires the addition of fresh randomness. As with most masking schemes, linear functions (e.g. XOR) are trivially shared by applying the unshared function to all input shares independently, while non-linear functions (e.g. AND) are more difficult to share correctly. To ensure the resistance of TI against leakage through glitches, registers have to be placed between components when composing larger circuits from multiple functions.

2.2.4 Domain-Oriented Masking (DOM)

Another well-known approach to secure masking is DOM [GMK16a], a scheme that achieves d-th order security using the optimal solution of $d+1$ shares. In DOM every sensitive intermediate value is split into $d+1$ shares, which are assigned to $d+1$ domains. Then, all operations on these shares are implemented in a shared way in order to achieve independence between shares of each domain from shares of all other domains. This is trivial for linear operations as they can be composed from shares of one domain only. Non-linear functions are secured by adding fresh randomness and using registers to prevent the propagation of glitches through the circuit. This can result in smaller circuits compared to other well-established masking schemes such as TI due to a reduced number of shares. However, latency and randomness requirements are increased.

2.2.5 Probe-Isolating Non-Interference and Hardware Private Circuits

Several attack models have been developed in order to abstract the observation of side-channels in a meaningful way and allow the construction of masking schemes that are provably secure in the respective model. The probing model, formalized in [ISW03], generally requires any set of values that can be accessed by up to d probes in a circuit to be independent of the sensitive variables. As this property is insufficient to guarantee the security of a composition of gadgets, which are themselves secure in this model [CPRR13], the more restrictive notions of Non-Interference (NI) and Strong Non-Interference (SNI) [BBD+16] were introduced. In order to satisfy NI, a set of $t \leqslant d$ probes on a gadget needs to be simulatable using at most t shares of each input. For SNI-secure gadgets, every set of t_{int} internal probes and t_{out} probes on the gadget output, where $t_{out} + t_{out} \leqslant d$, can be simulated using only t_{int} shares of each input. While the authors of [BBD+16] showed that gadgets can be securely composed using NI or SNI gadgets, the area and randomness costs are typically high because refresh gadgets are needed to allow composition. The Probe Isolating Non-Interference (PINI) notion proposed in [CS20] aims to solve this problem by developing a model which allows trivial composition with reasonable cost. For a PINI-secure gadget, the simulatability is constraint by the share index a probe is associated with. The resulting model allows trivial realizations of linear functions as with TI.

Following this notion, the authors of [CGLS20] proposed several hardware realizations of PINI-secure gadgets. The HPC2 AND gadget (which we use in this 6) provides dth-order security using $d+1$ input and output shares. It has an asymmetric latency with respect to the two input arguments, where the first argument influences the output after one cycle and the second after two cycles. The required fresh randomness per gadget amounts to $d \cdot (d+1)/2$ bits.

2.3 Side-Channel Leakage Assessment

In order to evaluate the vulnerability of sensitive devices and assess if implemented countermeasures are effective different strategies have been developed. Besides applying as many attacks as possible against the target TVLA has proven very useful in this context.

2.3.1 TVLA using Welch's t-Test

Statistical hypothesis tests provide a method to test if observations or parameters of samples drawn from distributions deviate from a null hypothesis with a given significance level of α. For example, the fundamental question whether two sets of data are significantly different from each other is often answered using Welch's t-test. For this, the test statistics follows a Student's t distribution and provides a quantitative value that the mean values of both sets are different. Formulating this as the null hypothesis $H_0 : \mu_1 = \mu_2$, a t-test provides a probability on whether the samples in both sets could have been drawn from the same population, i.e., indicating that both sets are not distinguishable by the test.

Hence, given two data sets $\mathcal{Q}_1$ and $\mathcal{Q}_2$ of cardinalities n_1 and n_2 and their sample means $\bar{x}_1$ and $\bar{x}_2$ and sample variances $\mathbf{s}_1^2$ and s_2^2, the t statistic with associated degrees of freedom is computed as:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}} \qquad \nu = \frac{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)^2}{\frac{1}{n_1 - 1}\left(\frac{s_1^2}{n_1}\right)^2 + \frac{1}{n_2 - 1}\left(\frac{s_2^2}{n_2}\right)^2}$$

Then, assuming the null hypothesis to be true, the p-value indicates the probability of observing a difference of means as large as or larger than the one that was observed. It can be calculated as $p = 2(1 - \mathrm{CDF}_\mathrm{t}(|t|, \nu))$, where CDF_t is the cumulative density function of the Student's t-distribution. In general, larger t values indicate low probabilities for the null hypothesis to be valid and hence give evidence to conclude that both sets $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are distinguishable and drawn from different distributions. For the sake of simplicity, assuming a sample size $n_1 + n_2 > 1000$, a fixed threshold of $|t| > 4.5$ is often considered to reject the null hypothesis with high confidence, as this corresponds to a probability of $p < 10^{-5}$ for wrong rejection of the null hypothesis (type-I or α-error). Note that statistical tests by themselves do not provide a bound for the probability of an error made when wrongly not rejecting the null hypothesis (type-II error or β-error). The probability of this error, defined as $\beta = \mathbb{P}(\text{reject } H_0 | H_0)$, is directly related to the power of a test, denoted as $1 - \beta$.

The relative simplicity and generic nature of Welch's t-test led to its common use in TVLA. For the first-order, univariate case TVLA has been developed in the seminal paper of Goodwill et al. [GJJR11]. Here, two leakage assessment methodologies, commonly referred to as specific and non-specific t-tests, are proposed for leakage detection at different statistical orders.

In particular, the non-specific test enjoys great popularity due to its simplicity and high level of abstraction. The main advantage of this test procedure is its independence of the underlying physical architecture and other implementation details. Given a Device Under Test (DUT) with fixed secret, two different sets of observations are acquired, only classified by fixed or random inputs to the evaluated implementation. Then, using Welch's t-test, rejection of the null hypothesis is attempted with respect to distinguishing both sets. However, due to its abstract nature, rejection of the null hypothesis may confirm detectable leakage but does not give any evidence on successful attacks and their complexities. As a consequence, non-specific TVLA can be used to confirm presence of observable leakage in particular, but does not allow to draw any conclusion about successful attacks in general. In a specific t-test the sets are

separated regarding a chosen intermediate value that is expected to be sensitive in an attack scenario.

In [SM15] the authors extended the basic TVLA approach to higher statistical orders as well as the multivariate case. Additionally, they proposed algorithms for the efficient calculation of the required statistical moments of the leakage (e.g. mean, variance and kurtosis) allowing a single-pass computation. A TVLA approach relying on the χ^2-test was proposed in [MRSS18] which shows improved detection performance in some cases when compared to the t-test.

2.3.2 Confidence Intervals

As a tool of inferential statistics, confidence intervals can be used to estimate parameters of sampled distributions, often their mean. A confidence interval can be constructed around the parameter providing a region containing the true parameter of the distribution with a given probability or confidence level of $1 - \alpha$. Naturally, any given confidence interval either does or does not contain the parameter it estimates, i.e., the probability of it containing the parameter is either one or zero. Therefore the confidence level indicates the probability of *constructing* a confidence interval containing the parameter for a given (unknown) distribution.

Statistical tests and confidence intervals are dual in the following sense: A confidence interval $C_{1-\alpha} = [\gamma_{\min}, \gamma_{\max}]$ that is constructed with a confidence level of $1-\alpha$ for a parameter γ contains exactly all points for which a hypothesis test using the same samples of the distribution with the corresponding significance level α would not reject the null hypothesis. Confidence intervals can also be constructed for differences of parameters of two distributions, which will be used in the remainder of this thesis. Note that in contrast to statistical tests, there is only one type of error possible for confidence intervals: The interval either does or does not contain the true parameter. The concrete choice of the significance level α is up to the user applying the method and depends on the field of application. Common values for α are 0.01, 0.05, or 0.1, where smaller values require a larger sample size in order to limit the size of the confidence interval to a useful range. In the remainder of this thesis we will use a significance level of 0.01 unless noted otherwise. As we will show in chapter 3, confidence interval can be used to eliminate some problems related to test-based TVLA.

Part II

Evaluation of the Side-Channel Security of Cryptographic Implementations

Chapter 3

Confident Leakage Assessment

In this chapter we propose a new side-channel assessment framework that combines an efficient data acquisition process with an evaluation methodology based on confidence intervals which extends established t-test-based approaches for TVLA. In comparison to previous TVLA approaches the new methodology does not only enable the detection of leakage but can also assert its absence. The framework is robust against noise in the evaluation system and thereby avoids false negatives. These improvements can be achieved without overhead in measurement complexity and with a minimum of additional computational costs compared to previous approaches. We illustrate the steps in the evaluation process by applying them to a protected implementation of AES.

The work described in this chapter was published at DATE 2017 [BPG18] and in it - Information Technology 2019 [BPW+19].

Contents of this Chapter

3.1 Introduction **19**
3.2 Confidence-Interval-based Leakage Detection **21**
3.3 Side-Channel Evaluation Workflow **26**
3.4 Evaluation Results **31**
3.5 The Influence of Noise **33**
3.6 Signal to Noise Ratio **34**
3.7 Pre-Evaluation Checks **35**
3.8 Conclusion **38**

3.1 Introduction

TVLA methods are frequently used to validate the effectiveness of countermeasures in cryptographic devices. These methods avoid the very costly application of many known attacks and rather use generic approaches to provide statements about physical security. In (moments-based) TVLA, the relevant side-channel, e.g., power consumption, is measured under different inputs, yielding side-channel traces. Then the evaluation procedure tries to decide whether the statistical moments of these traces are distinguishable. A widely adopted scheme (e.g. [GJJR11, SM15]) is based on Welch's t-test that decides if the mean values of two random

variables are different using the t-statistic. It should be noted that TVLA can not assert a device's security in all cases. It may fail if the noise present in a device is too low [Sta18].

Motivation A main feature of leakage detection schemes is their ability to assure the presence of leakage in a cryptographic computation with a given confidence $1-\alpha$. This is possible because hypothesis tests, such as the often-used Welsh's t-test, are designed to limit the error-probability for false-positive results (α-error) – in the case of side-channel analysis this would correspond to detected leakage – to an arbitrary level α. The downside of this approach is missing assurance about the error-probability for false-negative results (β-error). Therefore, these methods cannot support the statement: "No leakage is present". In consequence, a t-test-based SCA evaluation is not guaranteed to find existing leakage in a device. A negative result merely proves the inability of the test to find leakage, independent of its actual presence.

This issue is strongly related to the dependence of the t-test's result on the sample size. As the (Welch's) t-statistic is computed as

$$t = \frac{\bar{x} - \bar{y}}{\sqrt{\frac{{s_x}^2}{n_x} + \frac{{s_y}^2}{n_y}}} \propto \sqrt{n}\,,$$

given a positive absolute difference in the sample means, the test statistic will increase as the sample size increases and the sample means converge to the means of the underlying distributions. If the difference in means is small and/or the the variances are high, a very large n is required to detect leakage. Therefore, an insecure implementation might be evaluated as secure if the sample size is insufficient.

If a fixed threshold (e.g. [GJJR11, SM15]) is used, current TVLA methods fail to account for the number of sample points in the measurements. When measuring the leakage of a device, a trace with more sample points will generally have a higher maximal t-value than a trace with less horizontal resolution. This problem was identified by the authors of [ZDD+17].

Contribution We develop a new framework for TVLA based on confidence intervals. This solves two related problems with hypothesis test approaches:

- It allows the establishment of an upper bound for leakage that is robust against noisy evaluation systems. This allows statements about an implementations security even if a t-test can not identity leakage.
- It provides natural cut-off values for the number of measurements required in order to assert or reject security claims about cryptographic implementations. By choosing an allowable leakage level, measurements can be stopped when either the maximum lower limit or the maximum upper limit crosses that threshold.

We also solve the problem of sample-point dependence by applying the Šidák correction to confidence intervals. We demonstrate the overall workflow including hardware and software interaction for a time efficient acquisition of power measurements and exemplify the process and its parameters with an evaluation of an implementation with countermeasures. Finally, we name possible pitfalls when assessing a device within the proposed framework.

3.2 Confidence-Interval-based Leakage Detection

In the following section we will introduce a new method to detect the presence of leakage based on confidence intervals as well as discuss some improvements of our basic methodology. To clarify formalisms the following paragraph introduces the notation used throughout the chapter.

Notation Uppercase letters (X) denote random variables, whereas lowercase letters (x) denote their realization through a random sample. Sample sizes are denoted by n, where a subscript denotes the affiliated random variable when necessary. Arithmetic means (as random variable as well as realization) are denoted with $\bar{X}$, $\bar{x}$ resp. . The sample variance (again in both notations) is denoted with S_X^2, the sample standard deviation with S_X. Parameters of a normally distributed random variable are μ and σ, subscripts denoting the affiliated random variable. As for the normal distribution we have $\mu_X = E(X)$ and $\sigma_X^2 = E((X - E(X))^2) = Var(X)$ and use this denotations interchangeably as well as μ and σ^2 for higher moments (normal and centralized with even order) with the respective subscript. A $1-\alpha$ quantile of the t-distribution is denoted by $t_{1-\alpha}$. If the degree of freedom is relevant or unclear from context, it is added as a subscript.

3.2.1 From Statistical Tests to Confidence Intervals

In t-test TVLA, the comparison with a fixed number (e.g. the common 4.5) relates to the comparison of the computed t-value from the evaluation to a quantile of the t-distribution (hence the name t-test) for a significance level α. The significance level α states the probability that although the null-hypothesis (*there is no leakage*) is true, the evaluator accepts the alternative hypothesis (*there is leakage*) - and thus makes a mistake. This mistake can be controlled through choice of α and accordingly the corresponding quantile of the t-distribution. Therefore, a smaller α corresponds to a higher confidence when accepting the alternative hypothesis (*there is leakage*). It does not, however, yield any confidence for accepting the null-hypothesis (*there is no leakage*). The probability of making a mistake when deciding for the null-hypothesis, i.e. when instead the alternative hypothesis is true, the so called β-error, can not be determined a priori.

A standard method known in statistics to cope with the problems that arise when using hypothesis tests is the introduction of confidence intervals. They are constructed in a similar way but their significance level can be determined without assuming the null-hypothesis to hold. Confidence intervals are usually constructed symmetrically around a good estimator of the investigated parameter. Given a random sample and a desired confidence, the following statement holds: With a probability of said confidence the process to construct a confidence interval yields an interval that contains the unknown parameter.

We are now interested in a confidence interval for the absolute difference of means of two random samples of power consumption stemming from an implementation of a cryptographic primitive under different inputs. To that end, given samples of the random variables X and Y, two confidence intervals for the difference of means $\bar{X} - \bar{Y}$ and $\bar{Y} - \bar{X}$ are constructed for a given confidence. These are then combined to the required interval for $\left|\bar{X} - \bar{Y}\right|$. As we are now combining two statistical and thus probabilistic statements, the confidence for the final interval differs from its predecessors, namely the actual error α_t varies between $[\alpha; 2\alpha]$, where 2α is the

original error. Turning this procedure around, we devised the following framework to compute a confidence interval with a given confidence $1-\alpha_t$ for the absolute difference of means on the basis of given random samples:

(1) Choose a confidence level $1-\alpha_t$.

(2) Draw samples from both random variables X and Y.

(3) Compute $\widetilde{t}_{X,Y} = \frac{\bar{X}-\bar{Y}}{\widetilde{s}_n}$ with $\widetilde{s}_n = \sqrt{\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}}$

(4) Interpolate $\alpha \in \left[\frac{\alpha_t}{2}, \alpha_t\right]$ and compute a lower and upper bound $\Delta_{min}, \Delta_{max}$:

a) $|\widetilde{t}_{X,Y}| \leqslant t_{1-\alpha_t}$:

$$\begin{aligned} \alpha_t &= \alpha + T(-2|\widetilde{t}_{X,Y}| - t_{1-\alpha}) \\ \Delta_{min} &= 0 \\ \Delta_{max} &= \widetilde{s}_n(|\widetilde{t}_{X,Y}| + t_{1-\alpha}) \end{aligned}$$

b) $|\widetilde{t}_{X,Y}| > t_{1-\alpha_t}$:

$$\begin{aligned} \alpha_t &= 2\alpha - T(2\widetilde{t}_{X,Y} + t_{1-\alpha}) + T(2\widetilde{t}_{X,Y} - t_{1-\alpha}) \\ \Delta_{min} &= \widetilde{s}_n\left(|\widetilde{t}_{X,Y}| - t_{1-\alpha}\right) \\ \Delta_{max} &= \widetilde{s}_n\left(|\widetilde{t}_{X,Y}| + t_{1-\alpha}\right) \end{aligned}$$

If the random samples are drawn identically and independently distributed from X (Y resp.), the construction of $I = [\Delta_{min}, \Delta_{max}]$ yields with probability $1-\alpha_t$ an interval that contains $|\mu_X - \mu_Y|$.

Theorem 1. *Let $X \sim \mathcal{N}(\mu_X, \sigma_X)$ and $Y \sim \mathcal{N}(\mu_Y, \sigma_Y)$ be normally distributed random variables with all their parameters unknown. Then $|\mu_X - \mu_Y| \in [\Delta_{min}, \Delta_{max}]$ with confidence of $1-\alpha_t$, where Δ_{min} and Δ_{max} are computed as above.*

Proof. For brevity we will only sketch the proof here.
The proof works in three steps. First, the intervals containing the absolute difference are determined. Second, the probability for those intervals is determined. Third, the turnaround to yield a framework needs to be verified.
Let X,Y be distributed as in the theorem and $X_i \overset{iid}{\sim} X$ and $Y_i \overset{iid}{\sim} Y$ two random samples of size n_X and n_Y of X and Y. From basic statistics we then know that

$$T_{X,Y} = \frac{\left(\bar{X} - \bar{Y}\right) - (\mu_X - \mu_Y)}{\sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}}} \overset{approx.}{\sim} t_\nu \tag{3.1}$$

with

$$\nu = \frac{\left(\frac{s_X^2}{n_X} + \frac{s_Y^2}{n_Y}\right)^2}{\frac{1}{n_X - 1}\left[\frac{s_X^2}{n_X}\right]^2 + \frac{1}{n_Y - 1}\left[\frac{s_Y^2}{n_Y}\right]^2} \overset{n_X \approx n_Y}{\sim} n_X - 1 \tag{3.2}$$

the estimated degree of freedom from the sample. Standard algebraic techniques then yield a confidence interval for $\mu_X - \mu_Y$ for a given confidence $1 - 2\alpha$:

$$\begin{aligned}\mathbb{P}\left(\mu_X - \mu_Y \in \Big[\quad & \left(\bar{X} - \bar{Y}\right) - t_{\nu,1-\alpha} \cdot \widetilde{S}_n; \right. \\ & \left. \left(\bar{X} - \bar{Y}\right) + t_{\nu,1-\alpha} \cdot \widetilde{S}_n \Big]\right) \\ & = 1 - 2\alpha\end{aligned}$$

where $\widetilde{S}_n = \sqrt{\frac{S_X^2}{n_X} + \frac{S_Y^2}{n_Y}}$ for brevity. Repeating that calculation for $\mu_Y - \mu_X$, we receive the mirrored confidence interval with the same confidence. As we are only interested in the distance between means and not its direction, we can now combine these intervals to one, which yields after some algebra and case-by-case analysis:

$$\begin{aligned} I &= \left[0; \widetilde{S}_n \left(\left|\widetilde{t}_{X,Y}\right| + t_{1-\alpha}\right)\right] \ \ if \ \ \left|\widetilde{t}_{X,Y}\right| \leqslant t_{1-\alpha} \\ I &= \left[\widetilde{S}_n \left(\left|\widetilde{t}_{X,Y}\right| - t_{1-\alpha}\right); \widetilde{S}_n \left(\left|\widetilde{t}_{X,Y}\right| + t_{1-\alpha}\right)\right] \ \ else \end{aligned}$$

This concludes step one.
The second step investigates what confidence those intervals carry. It is clear, that this confidence should vary between $1 - 2\alpha$ and $1 - \alpha$, as it should at least have the confidence of one single directed confidence interval and might have, at best, half that error probability (this only exactly occurs when $\left|\widetilde{t}_{X,Y}\right| = t_{1-\alpha}$). Using probability theory, one obtains

$$\begin{aligned} \alpha_t &= \alpha + T(-2|\widetilde{t}_{X,Y}| - t_{1-\alpha}) \ \ if \ \ \left|\widetilde{t}_{X,Y}\right| \leqslant t_{1-\alpha} \\ \alpha_t &= 2\alpha - T(2\widetilde{t}_{X,Y} + t_{1-\alpha}) + T(2\widetilde{t}_{X,Y} - t_{1-\alpha}) \ \ else \end{aligned}$$

Finally, to be able to distinct cases only with the knowledge of $t_{1-\alpha_t}$ and not $t_{1-\alpha}$, we show that $\left|\widetilde{t}_{X,Y}\right| > t_{1-\alpha}$ if $\alpha \in \left[\frac{\alpha_t}{2}; \alpha_t\right]$, completing the proof. □

Relaxation of assumptions As introduced above, the statistics hold for normally distributed random variables. However, in the case of power consumption the distribution is not clear. Nevertheless, it is not per se necessary for the distribution of X and Y to follow the normal distribution but for their arithmetic means $\bar{X}$ and $\bar{Y}$. Fortunately, due to the central limit theorem, as soon as the sample size n increases, the arithmetic mean of random variables with arbitrary (but still iid.) distribution follows a normal distribution [Rüg02]. Furthermore, the condition of independence is not a strict one. A small level of dependence in drawing the random samples (as is to be expected when using the same device for the measurement process) is tolerable (if n is big enough) as it does not influence the distribution of the respective arithmetic means. This results in the applicability of our approach for the given task at hand.

Comparison with t-Test Method The resulting confidence interval subsumes the t-test's result. If the confidence interval has a positive lower bound (and is computed with the same significance level as the t-tests), this will be equivalent to the t-test asserting that there is some leakage. At the same time, the confidence interval does also contain information about the magnitude of the leakage. The closer the lower bound is to zero, the tighter the decision towards leakage detection has been. If the lower bound is zero, this will be equivalent to the t-test

failing to demonstrate any leakage. In addition to making both decisions statistically *valid*, the confidence interval method bounds the maximal difference between the means of both power consumptions (with a significance level). That is, it is possible to make a statement of the form: With a confidence of $1 - \alpha_t$, both means do not differ more than the value of the upper bound.

Influence of parameters There are two ways of looking at this method's parameters. First, to interpret the results (e.g. length of the confidence interval) and second, for the evaluator to construct the sample in a way that some properties are satisfied in the end.

- For a given confidence level, the length of a confidence interval is initially influenced by two factors: Deviation and sample size. Obviously, they work in opposite directions. A higher sample size will lead to a smaller confidence interval and a higher deviation will lead to a larger confidence interval. Additionally, the interval's length is also influenced by the position of $\widetilde{t}_{X,Y}$ in relation to $t_{1-\alpha_t}$. This stems from the fact, that the computed α from the above framework is closer to α_t if $\widetilde{t}_{X,Y}$ is close to $t_{1-\alpha_t}$ - thus resulting in a smaller confidence interval. If $\widetilde{t}_{X,Y}$ is closer to zero or its absolute value is large, α will converge to $\frac{\alpha_t}{2}$ and thus, the confidence interval will turn out to be relatively larger.
- For a given sample (and thus fixed sample size and deviation) the length of a confidence interval is bigger, if the confidence level $1-\alpha_t$ is higher (or the probability of error smaller).

If the evaluator wants to keep the length of the confidence interval in a certain range and also wants his confidence to fulfill some requirements, he needs to specify his sample size accordingly and, additionally, needs to keep his deviation as small as possible, i.e., reduce measurement noise. This is in contrast to t-test-based evaluation, where an unsuitable measurement system may indicate a false sense of security by producing low t-test results or requiring a high sample size.

3.2.2 Family-wise Error Rate Correction

Up until now, we only considered one timepoint j at a time. For each of those timepoints the introduced procedure yields an intervall $[\Delta_{min,j}, \Delta_{max,j}]$ that contains the expected difference of means with significance level of α.

However, to evaluate an implementation in terms of leakage, we are not only interested in every single point but in a statement of the form: With significance level α the difference of means (expected values) is contained in its respective interval at every time point.

Generally speaking, if we have m timepoints of interest to us each with their respective interval $I_j = [\Delta_{min,j}, \Delta_{max,j}]$ and a significance level $\alpha_{t,j}$, the statement

$$\forall j \in \{1, ..., m\} : |\mu_X - \mu_Y| \in [\Delta_{min,j}, \Delta_{max,j}] \tag{3.3}$$

holds (under the assumption of independence) with significance level $\alpha_{total} = 1 - \prod_{j=1}^{m} (1 - \alpha_{t,j})$.

But this results in three problems: First, we can not simply assume independence. Second, α_{total} is obviously highly dependent on m which should not be the case (that would build an incentive for an evaluator to test less timepoints to get a better confidence level). Third, if we assume sensible significance levels, α_{total} converges to 1 very fast.

To adress the second and third problem, we propose the usage of Šidák correction[Sid67].

(1) Choose total confidence level $1 - \alpha_{total}$

(2) Compute $1 - \alpha_t = \sqrt[m]{1 - \alpha_{total}}$

(3) Execute the above procedure with $1-\alpha_t$ as confidence level for every time point $1 \leqslant j \leqslant m$.

This method yields confidence intervals I_j for every time point, such that statement 3.3 holds with confidence $1 - \alpha_{total}$. This solves problems two and three to the extent that it shifts the problem towards the length of the confidence intervals. In this setting it is possible for the evaluator to set a confidence at which he wants to evaluate the given implementation which is then by design neither close to 1 nor dependent on m. Obviously, a large m constrains α_t to be rather small and thus the confidence intervals to be larger. Still, the resulting confidence intervals contain useful information (instead of an α_{total} that converges to 1) and as the size of $t_{1-\alpha_t}$, which is relevant to the confidence intervals' size, is highly affected by the sample size n, i.e. how many traces an evaluator uses, it is possible to counter large values of m with large values of n.

Problem 1, independence, is both more critical and less at the same time. On the one hand, it is unclear what kind of dependence exits and how it was to be measured if one wanted to include it in an evaluation. The authors of [WO19] note that this approach only produces conservative results if there is no dependence between the analyzed parameters at different points. We therefore propose to use the Bonferroni correction procedure which does not rely on this assumption. Here, the per-point confidence is computed as $1 - \alpha_{\text{pp}} = 1 - \frac{\alpha}{m'}$.
All Family-Wise Error Rate (FWER) correction procedures reduce the statistical power of the analysis by reducing the accepted (type-I) error rate for each point. For statistical tests, e.g., Welch's t-test this results in increased type-II error rate which leads to potentially undiscovered leakage in the context of TVLA. When confidence intervals are used, increased per-point confidence produces larger intervals. While the Bonferroni correction does reduce the statistical power of the analysis when compared the Šidák correction, the loss is small in practically relevant cases[1] and outweighed by the strict FWER control in case of dependencies.

3.2.3 Higher-Order Moments

Our confidence interval based approach for leakage assessment can be extended to provide limits for higher-order leakage using methods analogous to the ones described in previous TVLA-schemes [GJJR11], [SM15]. There, the authors reduce the task of detecting higher-order leakage to the problem of detecting first-order leakage in pre-processed traces. First, the acquired traces are pre-processed, combining samples with an appropriate function. The result is then evaluated with the same metric as in the first-order case. For example, second-order univariate leakage is analyzed using mean-free, squared traces, yielding bounds for the difference of the traces' variance. In general, while the first-order confidence interval yields bounds for the difference of means of the signals, the higher order test can provide bounds for the difference of the higher-order moments. For statistical moments μ_d for $d > 2$, the authors of [SM15] construct a t-test based on standardized moments. We deviate from this approach by using centralized moments

[1] For example, given $\alpha \leqslant 0.05$, $m \geqslant 5$, $d \geqslant 2$: $\frac{(1-\alpha)^{\frac{1}{m'}}}{1-\alpha/m'} > 0.99992$.

instead, allowing the analysis of the SNR if required. For $d > 1$, the means μ_d used to calculate the dth-order confidence interval are given as the dth central moments of the signals:

$$\mu_d = \frac{1}{n}\sum (x - \mu)^d. \tag{3.4}$$

The variance used to estimate the dth-order confidence interval can then be computed based on the central moments:

$$\begin{aligned} s_d^2 &= \frac{1}{n}\sum \Big((x-\mu)^d - \mu_d\Big)^2 \\ &= \frac{1}{n}\sum \Big((x-\mu)^{2d} - 2(x-\mu)^d\mu_d + \mu_d^2\Big) \\ &= \mu_{2d} - \mu_d. \end{aligned} \tag{3.5}$$

The confidence intervals can then be constructed according to Sect. 3.2.1.

If the evaluator has access to the masks, the traces can be pre-processed by averaging before calculating the higher-order moments, in order to reduce the measurement noise as suggested in [ZDD+17]. This will in general result in reduced measurement complexity. However, for SNR calculation, the variance over all randomly chosen masks should be used as this is the noise an attacker has to face.

3.2.4 Efficient Implementation

In order to compute confidence intervals for moments up to order d an evaluator needs to estimate the mean values and the central moments μ_i for $1 < i < 2d$ of the signals. This can be achieved by fast iterative methods described in [SM15] or using histograms as demonstrated in [RGV17]. The actual source of the traces is independent of our framework: it is applicable to non-specific fixed-versus-random or fixed-vs-fixed measurements, as well as analysis regarding specific intermediate values. The confidence interval can then be computed in regular intervals, e.g., after each 1000 measurements. As the calculations are independent for each sample point, they can easily be parallelized.

3.3 Side-Channel Evaluation Workflow

All aforementioned statistical methods used in side-channel attacks and evaluation settings have in common that they require the acquisition of a large number of measurements of the power consumption of the targeted device during its operation. A measurement setup capable of acquiring the side channel traces within reasonable time is therefore crucial in performing attacks and assessments.

For a setup with the purpose of evaluating hardware implementations, the SAKURA-G evaluation board [SAK], specifically designed for power trace acquisition has been used throughout the literature (e.g. in [CBG+16, SM15, MOBW13]). The board features two *Xilinx Spartan 6* FPGAs, one of which is programmed with the design under test (target).

Since the only communication channel between the SAKURA-G and the measurement computer (host) is via a slow serial link, the second FPGA (control) is commonly used to generate the inputs to the target. The actual acquisition of the current consumption of the target is

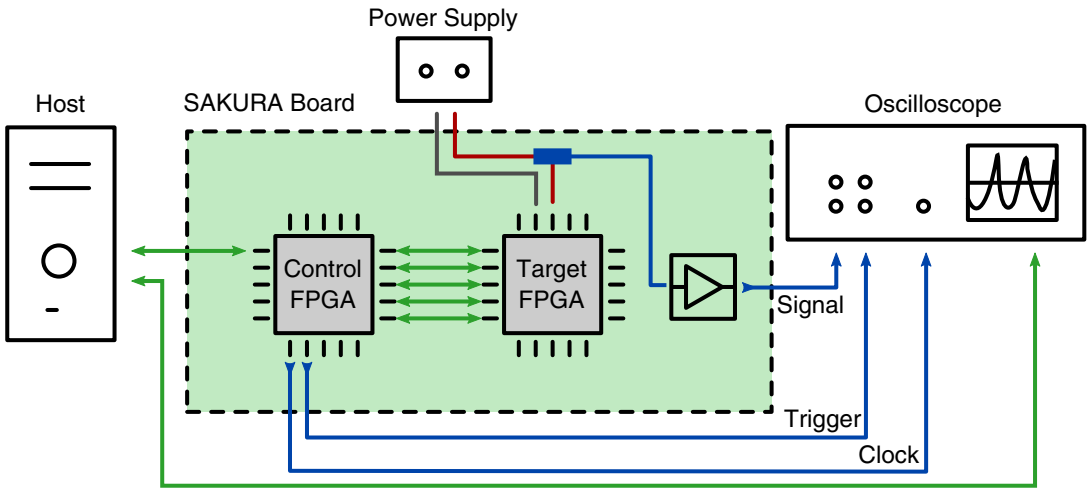


Figure 3.1: Measurement setup

achieved by measuring the voltage drop at a small resistor in the its supply line with an oscilloscope. As charging an output pin of the FPGA to trigger the oscilloscope's measurement significantly increases the chips power consumption, this should be done on the control FPGA to prevent any influence on the measurement of the power usage of the target.

To be able to have the target design run at different clock speeds its clock should also be generated by the control FPGA. For synchronization with the internal clock of the oscilloscope this clock can also be forwarded to the oscilloscope. This general setup is depicted in Fig. 3.1.

To reduce the amount of data transferred over the serial link between host and control, following [SM15] a setup should feature seedable RNG on the control FPGA to generate random input data and random masks, and to randomly decide whether a trace with fixed or random data should be collected.

This setup enables the use of so-called *sequence* or *block* mode of most oscilloscopes, in which a fixed number of identical traces are captured successively with only short time in between trigger signals. These traces are usually buffered in the internal memory of the oscilloscope. The acquired power traces are transferred to the host as a batch after all measurements in a block have finished. Depending on the duration of the operation to be analyzed and the amount of the scope's memory this enables to capture thousands of traces without any data exchanged between the board, the computer and the oscilloscope.

Communication from host to control FPGA is therefore limited to seeding the RNG and some initial configuration. Additionally, after each block of measurements has been collected, a hash over all the (unmasked) ciphertext generated by the target FPGA is sent back to the host.

Since all operations of the control and target FPGA only depend on the RNG's seeds and thus they can be performed identically on the host, the expected output should be hashed and compared against a hash of the target's outputs generated on the control FPGA. Thus, correct operation of the target design during the acquisition phase is confirmed at regular intervals.

Since most communication during measurements is between the two FPGAs, an efficient design should feature a fast communication channel, e.g. a bidirectional parallel interface or a high speed serial link, between them.

As the target FPGA is fully controlled (clock, reset, data) by the control FPGA, the latter can reset the target, issue the trigger signal to the oscilloscope and start the target's operation after some constant time. After each operation, the control FGPA reads the target's result and puts the target back into reset state. The result is iteratively hashed to ensure correct operation, and the next measurement is started. The sequence of steps during a measurement is shown in Fig. 3.2.

Throughout the remainder of this chapter we will repeatedly refer to an example implementation to illustrate key points of the assessment process more clearly. We chose a 128-bit AES engine protected with a domain-oriented masking scheme [GMK16a] for several reasons: It is a masking-protected core of a very well-known cipher, it does exhibit (small) leakage in our assessment and the design is available online [Gro16a]. The implementation allows arbitrary protection order by setting a VHDL-generic accordingly and uses $d+1$ shares in order to achieve d-order security as described in Section 2.2.2. For our evaluation, we chose the first-order secure variant of the design. We synthesized the interleaved variant with 5-stage S-boxes on our FPGA platform. The bitstream was generated using Xilinx ISE 14.7 with the options `keep hierarchy` on and `register duplication` off, in order to prevent interference with the masking countermeasure. If the synthesis tool supports register retiming, it must be disabled as the position of registers is critical for the security of masked implementations.

For the confidence intervals and t-test evaluations we assume a significance level α of 0.01, unless stated otherwise.

3.3.1 Measurement Parameters

As flawed measurements might give an assessor the false impression of a secure implementation, care has to be taken to select appropriate parameters and to reduce noise within the system.

Input Range A first step in this process is to ensure that the full input range of the oscilloscope is used. While most oscilloscopes feature different settings for their input ranges and the target FPGA's power signal is already amplified on the SAKURA-G, we found that especially for smaller designs consuming less current – such as a single protected s-box –, an additional Variable-Gain Amplifier (VGA) is necessary. Otherwise, the measured samples only yield a few bits of information, and show immense quantization noise. If the oscilloscope's analog front-end features a VGA it can be used instead. Note that while an additional amplifier enables the use of the full vertical resolution of the oscilloscope, it may also affects the measurement as it effectively functions as a bandpass filter, as it has been shown in [MM13].

Bandwidth While the bandwidth of single switching cells on the FPGA is typically in the GHz range, capacitances on the chip's die, its bonding wires and traces on the PCB essentially act as low-pass filters (see [MOP07]). Thus, the relevant information is contained in lower frequencies. To reduce externally introduces noise in the power traces, one should limit the oscilloscope's bandwidth to, e.g. 25 MHz to obtain a cleaner signal, as shown in Fig. 3.3. However, additional (digital) filters can also be applied during postprocessing. When using filters in leakage assessment care must be taken that the side-channel does not contain leakage in the suppressed frequency range in order to avoid a false negative evaluation. When in doubt, filtering should be not be applied and increased measurement complexity must be accepted.

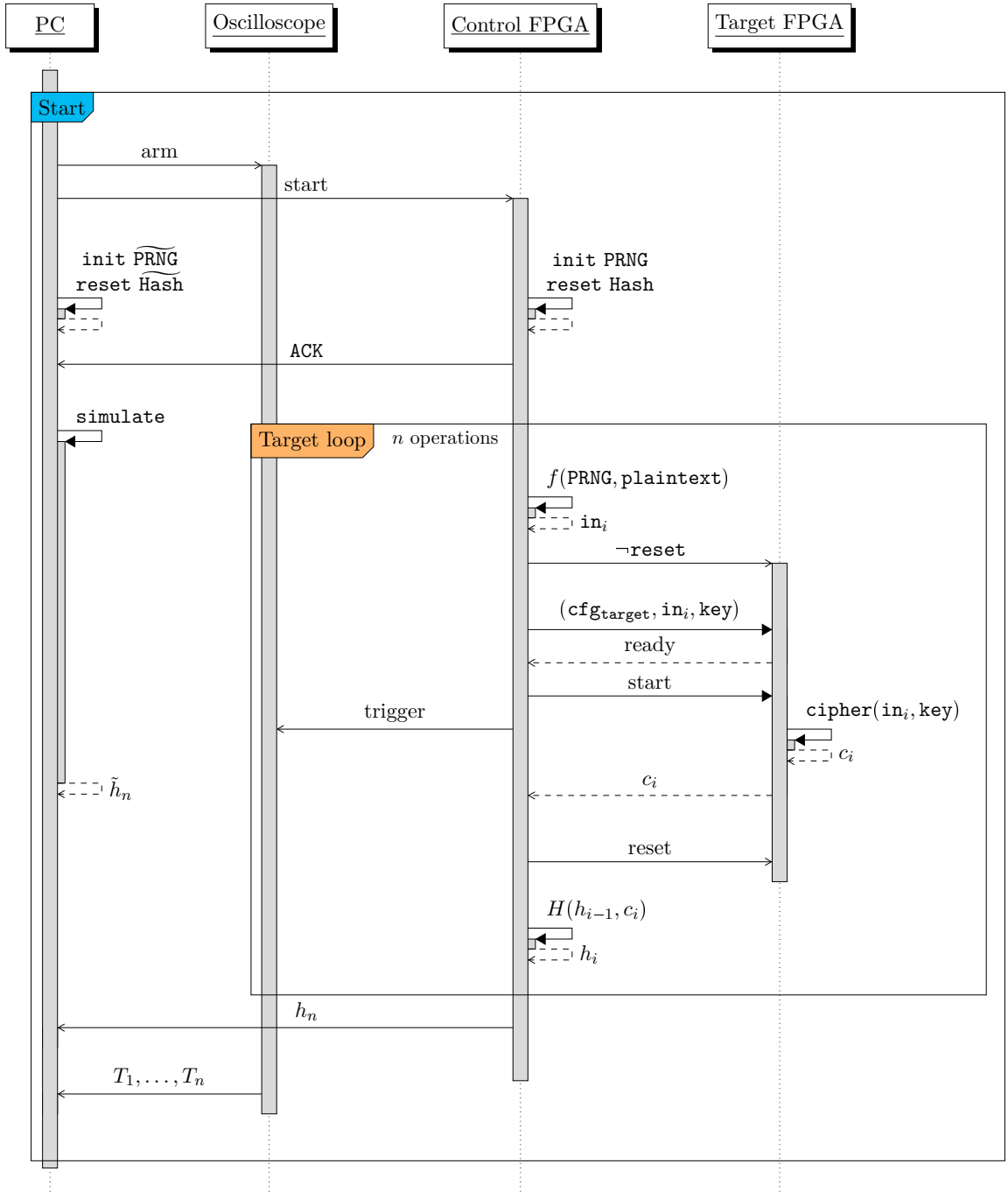
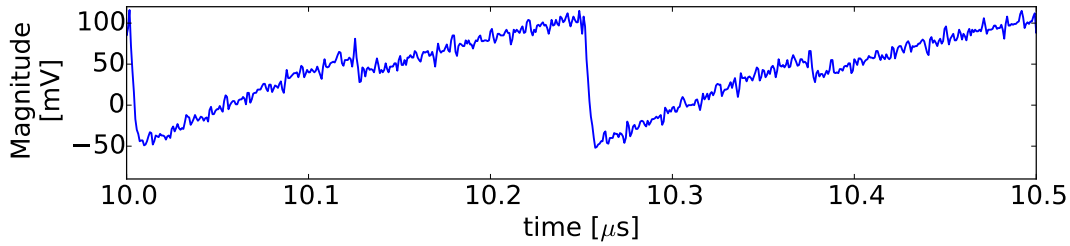
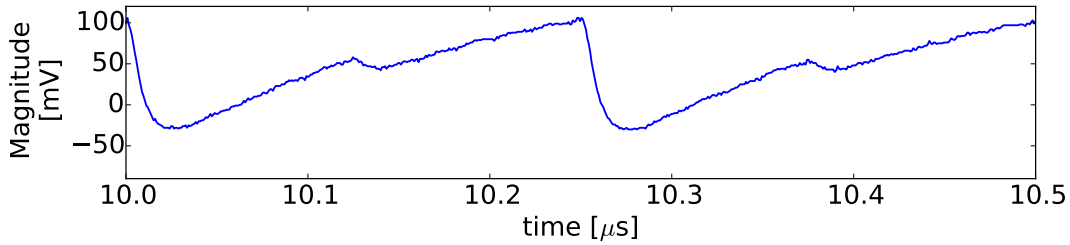


Figure 3.2: Measurement sequence



(a)

Without lowpass filter



(b)

With lowpass filter

Figure 3.3: Comparison of a power trace recorded without and with a 25MHz lowpass input filter.

Sample Rate and Clock Frequency Furthermore, the oscilloscope's sample rate is the main measurement parameter. In the general case the sample rate should be high enough to clearly separate the power consumption peaks at each clock cycle, and thus is dependent on the target device's clock frequency. While a higher sample rate gives a more detailed picture of the current consumption during a single clock cycle of the target it significantly increases the amount of data to be processed later on. In contrast to this, a higher clock frequency leads to the current peaks of clock cycles overlapping. As argued in Section 3.2.2, when later performing statistical tests or computing confidence intervals, a large number of sample points skews the analysis. Thus, choosing the clock frequency of the target and the sample rate of the oscilloscope is a tradeoff between fine-granular power traces, and the time effort for measurement, postprocessing and evaluation.

We found that a clock frequency of 4 MHz, combined with a sample rate of 1.25 GS/s, using a 25 MHz bandwidth limiter gives a good starting point. In our example, where one trace of the AES target design covers 55.25 µs, these parameters yield 69062 samples per trace. To fully use the oscilloscope's input range, we add an external amplifier (Mini Circuits ZFL-1000LN+) after the SAKURA-G's onboard amplifier. A sample trace with these parameters from our setup is given in Fig. 3.4.

3.3.2 Data Acquisition

Having determined suitable parameters for the target's clock frequency, the oscilloscope's input range and the sample rate, to ensure the fastest possible data acquisition, the block size, i.e. the number of successively captured traces, should be determined to fully utilize the oscilloscope's

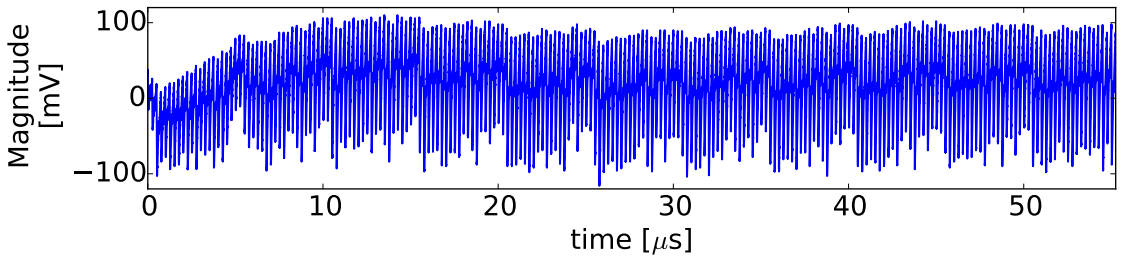


Figure 3.4: A sample trace of the DOM AES core.

internal memory. This reduces the impact of the overhead for host-to-control FPGA and host-to-oscilloscope communication on total acquisition time.

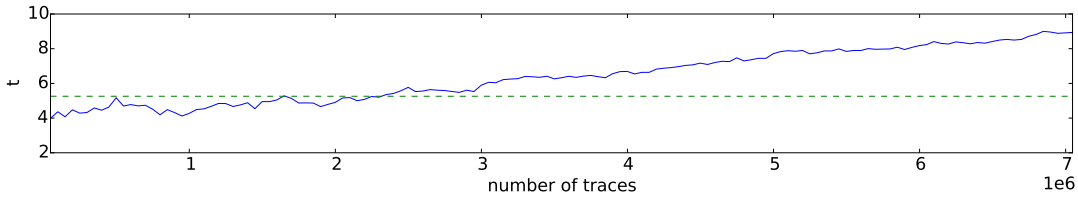
In our example, we used a PicoScope 6404D oscilloscope equipped with memory for 2 GS. However, the oscilloscope distributes all available memory evenly among active input channels – including the trigger channel. Thus, at 69062 samples per trace, we would be able to capture around 14k traces in a single block measurement. However, our analysis showed that the maximum data transfer rate over the oscilloscope's USB3 interface is roughly 1.5 Gbit/s and not increasing with larger blocks of data. As the setup overhead for each block at this size is already negligible, we captured 10k traces in each block. Other oscilloscopes might benefit more from larger blocks.

Including an overhead for reset, random number generation and communication between both FPGAs, the capture of each trace takes 76.2 µs, while transferring the full block of data from the oscilloscope's internal memory to the host accounts for more than 930 µs per trace. With incremental and parallel computation of the centralized moments (in parallel to the acquisition of traces) taking far less, data transfer forms the bottleneck of our measurement system. To summarize these metrics, our system acquires and processes 990 traces per seconds of the example AES core with aforementioned parameters.

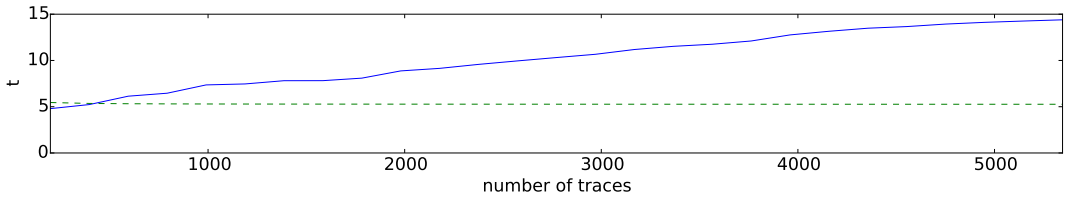
3.4 Evaluation Results

In this section we study the effectiveness of our evaluation framework by applying it to the previously described first-order secure implementation of the AES, protected by a domain-oriented masking scheme. We analyze the absolute difference in the statistical moments of the power consumptions using our proposed framework and compare the results to a Welch's t-test. More precisely, we use our framework to answer the question: Which band contains the absolute difference between the means of the moments of X and Y for all sample points. As an encryption on the target device took 55.25 µs, the resulting traces consist of 69062 samples each. Therefore, the corrected confidence level is $1 - \alpha_t = 0.99^{\frac{1}{69062}} = 1 - 1.46 * 10^{-7}$ as by Sect. 3.2.2.

Figure 3.5 depicts the result of a t-test trying to prove the hypothesis H_1: $\exists j : \mu_{j,X} \neq \mu_{j,Y}$ which states that at some sample point(s) there is a difference in the mean power consumption. The results indicate that there is some first and second order leakage, as the maximum t-value exceeds the corresponding threshold after a sufficient number of measurements. However, the t-test can not provide a confident assurance of the magnitude of leakage. Even worse, if we



(a) 1st-order t-test



(b) 2nd-order t-test

Figure 3.5: Maximum of absolute t-statistics and detection threshold for moments 1 and 2.

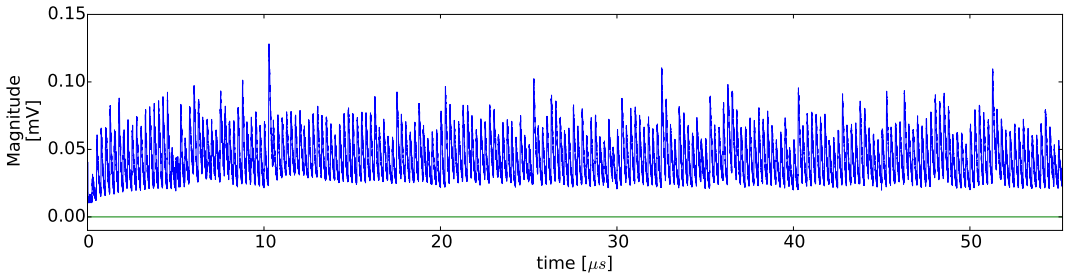
only captured two million traces the t-test would not allow any confident statement regarding the first order leakage, as the maximum t-value would not have crossed the threshold. There could be two reasons for failing to detect leakage: either there is no leakage, or the number of measurements is too low to discover it. In contrast, Fig. 3.6a shows the confidence intervals, as constructed by our method, using two million traces. While the interval does not prove the presence of leakage (the lower limit is zero), it limits (with confidence 0.99) the maximum possible leakage at each point by the corresponding upper interval limit.

If more traces are available, the interval limits become tighter, allowing a closer estimation of the leakage. Figure 3.6b shows the intervals computed using three million traces. As the threshold of 5.26 (corresponding to $\alpha_t = 1.46 * 10^{-7}$) is exceeded in the t-test, by definition of the interval, the lower limit becomes non-zero for some points, while the upper limits simultaneously converge against the true leakage. If even closer estimates are required, more traces can be recorded. Figure 3.6c shows the tight leakage estimation using 150 million traces.

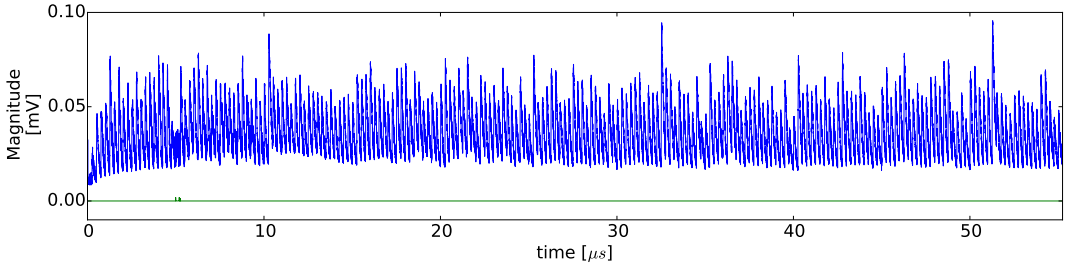
Specific sample points can also be analyzed independently. Figure 3.7 depicts the development of the interval at sample point 8715, corresponding to a peak in leakage at 6.97 µs, over the number of traces. If statements about (single) arbitrarily chosen points are made, as in the multi-point case, it is important to use the FWER-corrected confidence level as described in Sect. 3.2.2. If the point was chosen at random or using prior knowledge (such as prior measurements) and only a single point is considered, the unadjusted confidence level can be used.

As discussed in Sect. 3.2.3, the framework can also be used to assess higher-order leakage. Figure 3.8 shows the obtained intervals for second order. As the difference in the second moment is much higher than in the first, considerably less traces are required to tightly bound it.

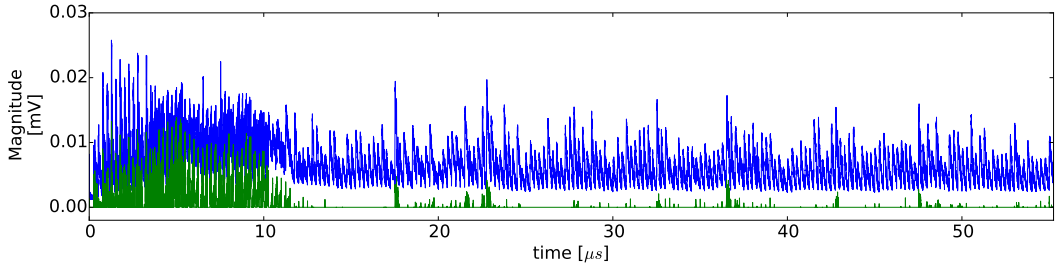
It is important to note that our results do not necessarily support the conclusion that the countermeasures in the analyzed implementations fail to provide protection against side-channel attacks. We can merely state that the *concrete instantiation* using the bitstream created by us and programmed into our target device exhibits the leakage described above.



(a) 1st-order confidence intervals using 2M traces.



(b) 1st-order confidence intervals using 3M traces.



(c) 1st-order confidence intervals using 150M traces.

Figure 3.6: Confidence intervals depicting lower (green) and upper (blue) bounds for the absolute difference in means. Please note different scales.

3.5 The Influence of Noise

The noise level in TVLA plays a significant role for assessment accuracy and the security of Side-Channel Analysis (SCA) countermeasures. Hiding countermeasures directly rely on a low SNR to prevent attacks and masking schemes are only secure if the noise in the system is sufficient. From an evaluator's standpoint the noise should be minimal in order to correctly and efficiently access the security of an implementation. In classical t-test-based TVLA noise can even lead to wrong assertions of security. Figure 3.9 shows an example for the effect of a high noise floor during a security evaluation. An evaluator with a noisy measurement system collecting the power traces resulting in the evaluation shown in Fig. 3.9b might wrongly conclude that no leakage is detectable while a cleaner measurement system will detect leakage as shown in Fig. 3.9a. The reason for this susceptibility to noise is that $t \propto \sqrt{n}/s$, therefore the required

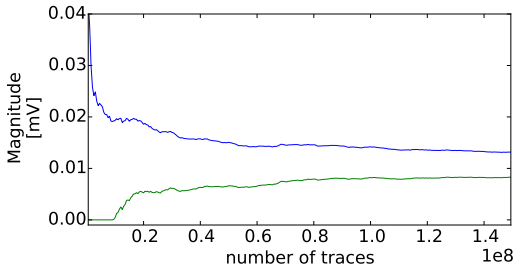


Figure 3.7: First-order confidence intervals at sample point 8715 over the number of measured traces.

number of traces increases r^2-fold if the SNR is lowered by a factor of r, in order to reach the same detection power.

When using the confidence-interval-based approach described in this work such ambiguity can be prevented. Figure 3.10 depicts an evaluation using our approach under the same parameters. While the evaluation in a noisy environment can not reliably detect any leakage – reflected by the lower bound being constant zero – the uncertainty of the evaluation is made explicitly visible in the correspondingly higher upper bound for the leakage. Therefore an evaluator with a noisier acquisition system needs to collect more traces in order to achieve a sufficiently low upper bound of the intervals and thus assert the absence of leakage above a certain threshold.

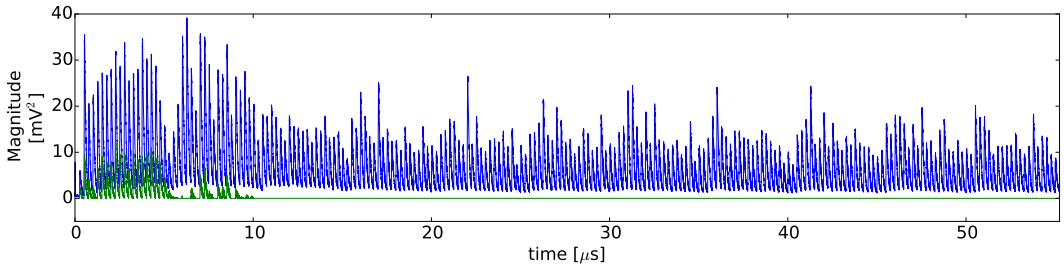
If an evaluator can control the masks and the Random Number Generator (RNG) on the DUT it is possible to reduce the measurement complexity for a higher-order evaluation by averaging over several traces with the same masks and RNG seed before preprocessing the data for evaluation. This is possible because the noise in the traces is reduced before the amplification through preprocessing [Sta18].

3.6 Signal to Noise Ratio

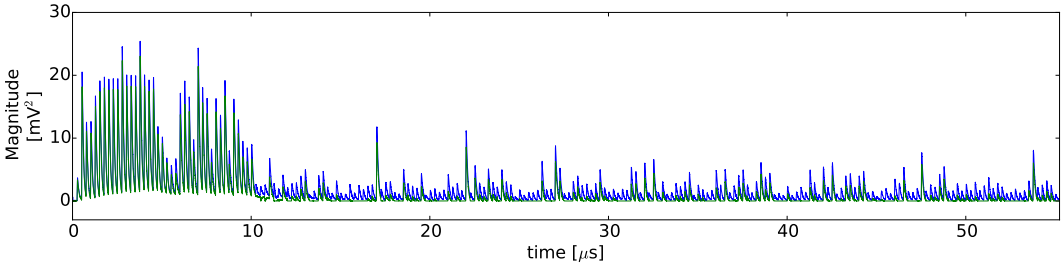
In order to assess the resistance of an implementation against SCA the absolute difference between the statistical moments for different inputs is not the only relevant metric. As shown in [MOP07] the measurement complexity of an attack increases with more noisy signals because the statistical distinction becomes more complex. A standard way of capturing the relation between the leakage and the noise, that also provides a scale independent metric for a devices security, is the application of an SNR. The confidence-interval-based evaluation allows a sound estimation of the SNR in the leakage if three properties hold:

- The second order leakage needs to be small, i.e., $\sigma_X \approx \sigma_Y$. This implies similar noise levels for all input classes.
- There must be sufficient noise in the system: $\sqrt{\sigma_X} \gg |\mu_X - \mu_Y|$.
- The number of traces must be high enough for a close estimation of the noise: $s_X \approx \sigma_X$.

If these requirements are fulfilled, the the traces t acquired by an attacker will be distributed closely to $t \sim \mathcal{N}(\mu_X, \sigma_X)$ allowing the construction of a confidence interval for the SNR as



(a) 2nd-order confidence intervals using 5k traces.



(b) 2nd-order confidence intervals using 500k traces.

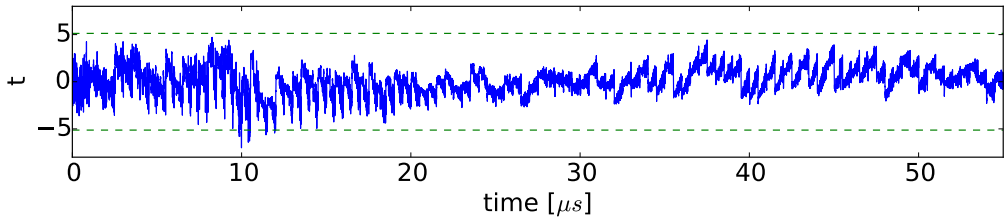
Figure 3.8: Confidence intervals for the absolute difference in variance (2nd-order analysis).

$I_{SNR} = \left[\frac{\Delta_{min}}{s_X}; \frac{\Delta_{max}}{s_X}\right]$ Note that, as this interval depends on the estimated variance s_X^2, the result is no longer valid for arbitrary measurement systems. An exemplary SNR-interval for is depicted in Fig. 3.11.

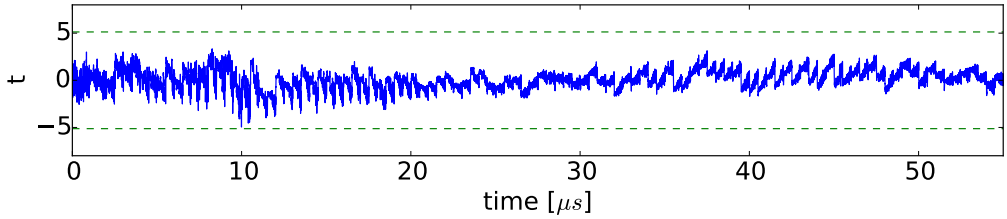
If an implementation with combined countermeasures, e.g., masking and a noise generator, is assessed the evaluator may choose to disable the hiding countermeasure for the confidence interval calculation. The decreased noise will yield a close leakage estimation while requiring less measurements. The actual noise with enabled hiding countermeasure can then be assessed subsequently and used as reference for an SNR interval. This noise estimation has much lower time complexity than a TVLA measurement because only an estimation for the variance itself is required, not for the – possibly very small – difference in variance between classes.

3.7 Pre-Evaluation Checks

Before a new design is evaluated using TVLA the correctness of the assessment system should be established: It should successfully detect actual leakage while not introducing spurious leakage through the measurement and evaluation setup that is not present in the DUT. The first property can be asserted by measuring the DUT in fixed-vs-random mode with constant masks, e.g., setting all masks to zero. In this mode, the device should show leakage even at orders that are protected by the masking scheme because the prerequisite of random uniform sharing is not fulfilled. If this test can not be conducted, e.g., because the DUT generates or refreshes the masks itself before executing the cryptographic algorithm, an unprotected dummy implementation with identical input-output behavior to the DUT should be synthesized and tested. Figure 3.12 shows the expected result of a fixed-vs-random evaluation with disabled masking

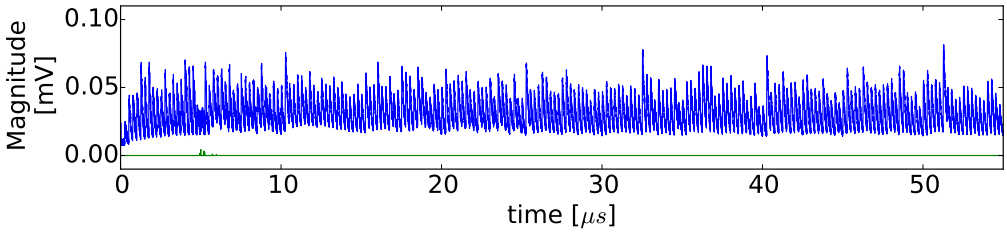


(a) Baseline measurement

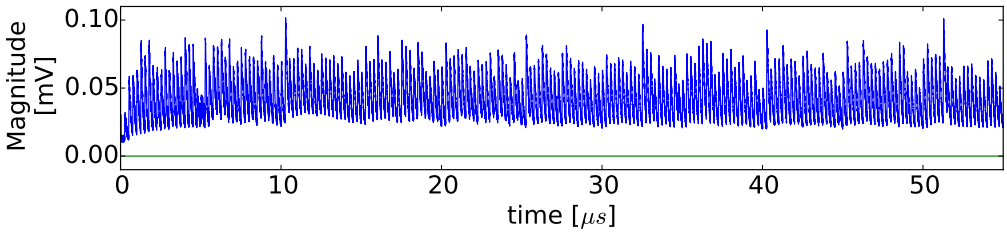


(b) Noise doubled (simulated)

Figure 3.9: Influence of noise on a t-test-based TVLA after 4 M measurements. Dotted lines mark the critical value corresponding to a significance of 0.01.



(a) Baseline measurement



(b) Noise doubled (simulated)

Figure 3.10: Influence of noise on a confidence-interval-based TVLA after 4 M measurements.

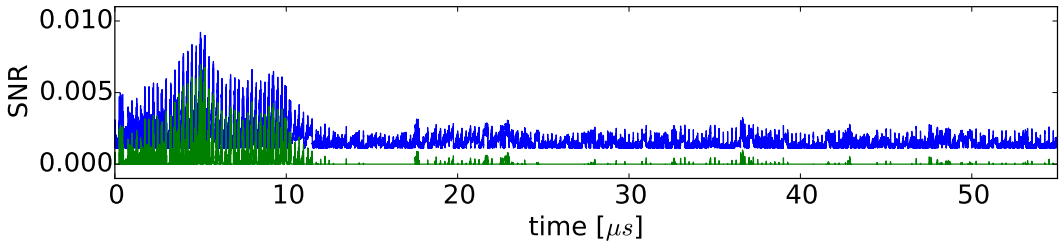
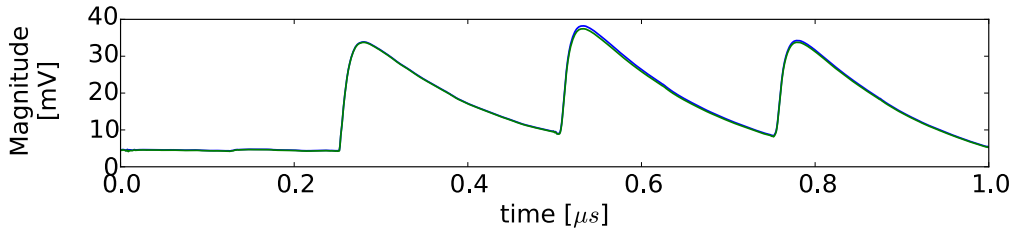
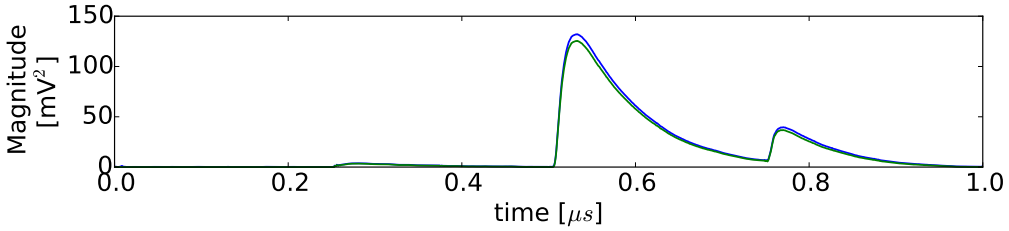


Figure 3.11: A SNR-interval for first-order leakage using 150 M traces.



(a) First order interval

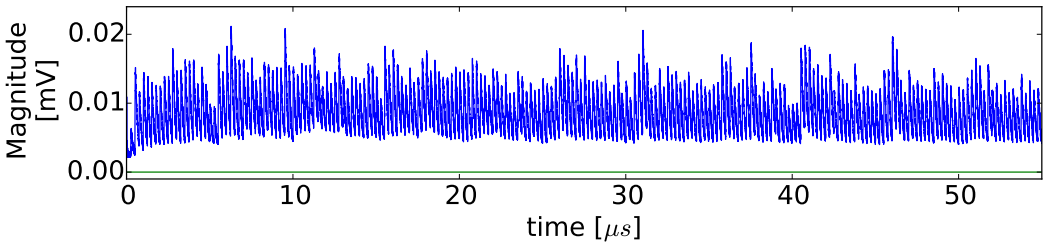


(b) Second order interval

Figure 3.12: The first 1 μs of a system test for fixed-vs-random input with constant masks using 100 k traces.

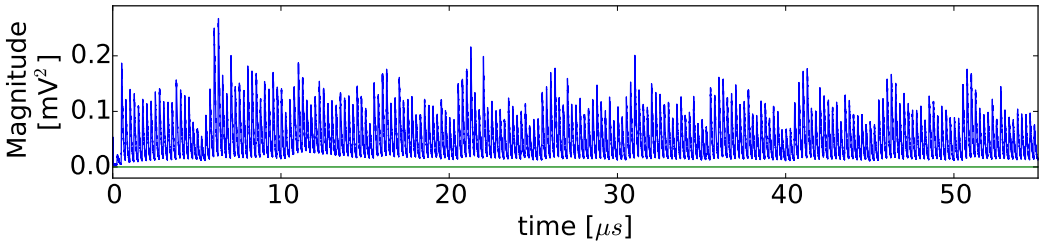
for a first-order secure design. Note the very high and tight estimation of the difference in both moments.

In order to ensure that detected leakage actually originates from the DUT an evaluation can be performed in which all input classes for the DUT are indistinguishable, e.g., by choosing them all uniformly at random. In all regards other than input generation, such as randomly interleaving the input class selection, the measurement system should operate as in the real assessment. Even with a high number of collected traces the lower limit of the calculated confidence interval should be zero for all moments. If the correctness of the mask-generating RNG can be asserted otherwise, a fixed-vs-fixed test with the same input for all classes is preferred as less traces will be required to show problems in the evaluation system, because of the reduced algorithmic noise. An example for the expected evaluation result is depicted in Fig. 3.13. Note that even after 45 M measurements the lower leakage bound is zero for all sample points while the upper bound is very low for both orders.



(a)

First order interval



(b)

Second order interval

Figure 3.13: System test for fixed-vs-fixed input with the same input for both classes after 45 M measurements.

3.8 Conclusion

We describe a new confidence interval-based framework for TVLA and demonstrate its advantages in comparison to established methods. We suggest our new framework to be applied as a drop-in replacement for t-test TVLA-methods currently in use in the industry and the scientific community to provide a clearer picture of the side-channel resistance of protected cryptographic implementations. To this end, instead of calculating a maximum t-value that is reached after a certain amount of measurements, maximum and minimum bounds for the leakage should be measured. We hope that our suggestions for a time efficient acquisition framework and the overview of our measurement parameter choices in conjunction with the interval-based analysis metric help evaluators and researchers in producing sound assessments of cryptographic devices and SCA countermeasures.

Chapter 4

Multivariate Leakage Assessment

In masked implementations, sensitive information is hidden in higher statistical moments of the leakage if processed at the same time (univariate) or in the combination of side-channel information from different points in time (multivariate) if processed sequentially. Test Vector Leakage Assessment (TVLA) is a common evaluation technique to address the growing number of specific attacks. However, the assessment of multivariate leakage requires the evaluation of all possible combinations of sample points, massively slowing down the evaluation and in turn the development of countermeasures due to computational complexity. In this chapter, we develop and compare different techniques to determine clock cycle combinations that leak information in a multivariate setting but allow to reduce the dimensional complexity of the measured data.

For this we develop an efficient multivariate assessment framework and show how this approach can be used to generate evaluation results that satisfy a desired confidence level, removing the uncertainty introduced by a measurement system. Eventually, we demonstrate the practical relevance of our approach by applying it to a masked sequential implementation of the PRESENT block cipher and a publicly available implementation of AES using a state-of-the-art masking scheme.

The work described in this chapter was accepted for publishing at IEEE Transactions of Computers [BWSG].

Contents of this Chapter

4.1 Introduction **39**
4.2 Preliminaries **41**
4.3 Multivariate Leakage Assessment **41**
4.4 Case Study: Sequential PRESENT **47**
4.5 Case Study: AES-DOM **51**
4.6 Performance Evaluation. **54**
4.7 Conclusion **55**

4.1 Introduction

In the previous chapter, we introduced a novel framework based on confidence intervals which addresses known shortcomings of t-test-based leakage assessment and evaluation. However, this

approach is still limited to univariate analysis but does not cover the combination of multiple measurements and samples for multivariate leakage evaluation. Unfortunately, in a black-box setting, without any information on POIs, multivariate analysis rapidly becomes infeasible as it requires exhaustive evaluation of all possible sample point combinations. Hence, given such an exponential complexity, multivariate leakage assessment is strongly limited in the number of sample points and massively increases development and evaluation time of protected cryptographic implementations.

4.1.1 Contribution

Given the previously mentioned limitations, this work presents an efficient assessment framework for multivariate leakage assessment based on confidence intervals.

More precisely, we first discuss and compare different pre-processing techniques for POI identification mainly with respect to efficiency and accuracy. Efficient and accurate identification of POIs and application of appropriate pre-processing and compression techniques eventually allows to reduce the evaluation complexity dramatically, again enabling continuous evaluation during development of countermeasures even when considering multivariate leakages. In a second step, we apply the concept of confidence intervals to multivariate leakage, hence, removing the uncertainty introduced by the measurement setup and assisting designers in achieving the desired level of security with high confidence. Ultimately, to demonstrate the application of our approach, we apply our concepts using a sequential implementation of a masked PRESENT as case study. In particular, this implementation has been constructed such that it does not exhibit any univariate leakage, but has some (expected) multivariate leakage due to sequential processing of masked shares. Finally, to emphasize the practical relevance of our framework, we conduct the evaluation of a state-of-the-art masked AES implementation, demonstrating the real-world applicability of our approach.

4.1.2 Related Work

Leakage Assessment Methodologies.

In order to improve the performance of TVLA evaluations, Reparaz et al. [RGV17] proposed an alternative computation of the statistical moments using histograms. In chapter 3, we proposed the concept of *confidence intervals* for the univariate case and demonstrated the benefits by overcoming known shortcomings in hypothesis testing due to noisy evaluation systems and sample point dependencies for larger number of sample points. Recently, Althoff et al. [ABK19] proposed a multivariate leakage assessment approach metric using a k-nearest neighbor criterion. Unfortunately, assessment of the effectiveness of this approach in the context of masked implementations is challenging, as results on practical experiments are not provided. Finally, Wegener et al. [WMM19] presented a leakage assessment concept that uses neural networks to build a distinguisher for leakage traces. Based on the results, this approach can detect leakage in masked implementations and also has a limited ability to detect POIs by using sensitivity analysis but can not provide assurance about the *absence* of leakage.

Pre-processing and POIs Identification.

As multivariate testing and evaluation is known to suffer from exponential complexity, different pre-processing techniques for POIs identification and complexity reduction have been presented and proposed in literature. In the course of research in this field, different approaches mainly based on Principal Component Analysis (PCA) or Linear Discriminant Analysis (LDA) for dimensionality reduction have been presented [BHvW12, BGH+15, SA08]. All these techniques have been shown to successfully reduce the complexity, but with varying quality in evaluation and analysis results. In a different attempt, the authors of [DSV+15] proposed a heuristic approach to POI detection based on projections pursuits. Here, a two-step optimization algorithm is used on masked implementations in order to find a projection of complete measurement traces to a smaller subspace. A downside of this approach is the number of parameters that must be selected a priori.

4.2 Preliminaries

In this section, we define the basic mathematical notations used in the remainder of the chapter and introduce the relevant statistical and mathematical tools.

4.2.1 Notation

In this chapter the same notation as in chapter 3 is used, with the following additions: The sample mean and sample variance of a random sample x drawn from a distribution are $\bar{x}$ and s_x, respectively. The number of samples in a measured trace, the number of traces measured, and the order of an analysis are denoted as m, n and d, respectively. The probability that statement X is true is denoted as $\mathbb{P}(X)$, whereas the conditional probability of X being true given Y is denoted as $\mathbb{P}(X|Y)$.

4.3 Multivariate Leakage Assessment

This sections details our evaluation framework for multivariate leakage, including the concepts for multivariate confidence intervals, and introduces complexity reduction approaches to make the evaluation feasible a the black-box setting (i.e., without prior knowledge of POIs).

Generation of Multivariate Traces.

In order to provide metrics for multivariate leakage we eventually construct confidence intervals for the difference of means of multivariate traces. In this step every d-tuple of samples is combined using a combination function for a d-variate analysis. If the original leakage traces consist of m samples each, this step results in traces of size m^d. Following [SM15], we use the centered product as the optimal (in case of Hamming-weight leakage) combination function. However, note that this combination function is commutative, hence, the number of distinct points in the multivariate trace is $m' = \binom{m+d-1}{d} < m^d$.

Construction of Confidence Intervals.

After the generation of multivariate traces, we then construct a confidence interval for the leakage contained at every index of the multivariate traces. As we require a (pre-selected) confidence α for *each* interval, a FWER controlling procedure for the confidence error resulting from multiple simultaneous interval constructions needs to be applied as in Sect. 3.2.2. This process ensures that the following analysis is valid for all samples when they are assessed in conjunction.

Multivariate Evaluation.

While our metric can be applied to higher variates, we focus on the bivariate case in this section as this is the most common case for practical black-box evaluation. Given a leakage trace L consisting of m samples $L_1, \ldots, L_m$, the bivariate leakage trace L' at sample points i and j is constructed using the centered product as:

$$L'_{i,j} = \left(L_i - \bar{L}_i\right) \cdot \left(L_j - \bar{L}_j\right).$$

This results in a preprocessed trace containing m^2 samples. As the centered product is symmetric only $m' = \frac{m \cdot (m+1)}{2}$ points need to be calculated. The confidence interval is then created for the absolute difference between the sets of collected traces as in chapter 3. Note that we compute intervals for the absolute leakage instead of the direct value in order to be able to construct a meaningful global interval, i.e., a single set of bounds which is followed by all combinations of samples. For this, the statistical moments of the preprocessed trace required for the construction of the confidence intervals are calculated using the one-pass algorithm described in [SM15].

As a consequence, the confidence intervals $[\gamma_{\min,i,j}, \gamma_{\max,i,j}]$ provide lower and upper bounds for the combined leakage at all sample points i, j for the chosen confidence level $1 - \alpha$. The Bonferroni correction assures that the probability of *all of the intervals* containing the true leakage is $\geqslant 1 - \alpha$.

In a final step the individual intervals can be combined in a worst-case analysis similar to the min-p approach used in traditional test-based leakage assessment [SM15]. By construction, $\exists(i,j) : \text{leakage}(i,j) \geqslant \max(\gamma_{\min,i,j})$ and $\forall(i,j) : \text{leakage}(i,j) \leqslant \max(\gamma_{\max,i,j})$, i.e., at least one combination of points exhibits at least leakage $\max(\gamma_{\min,i,j})$ and no combination exhibits more than $\max(\gamma_{\max,i,j})$. Therefore, $\gamma_{\min} = \max(\gamma_{\min,i,j})$ and $\gamma_{\max} = \max(\gamma_{\max,i,j})$ are a useful global metric to describe the evaluation result. Intuitively, $\gamma_{\min}$ holds the same information as the test-based assessment while $\gamma_{\max}$ can additionally certify[1] absence of leakage above a threshold.

Evaluation Procedure.

The overall procedure can be described as follows:

(1) Collect n side-channel traces, each containing m samples, using a fixed-versus-random or a fixed-versus-fixed measurement procedure.

[1] As for all statistical evaluation methods, this holds with the chosen confidence.

(2) Optionally apply complexity reduction algorithms described in subsection 4.3.1 resulting in m'/R samples per trace, with a reduction factor R. The choice of R primarily depends on the available computational resources.

(3) Choose the desired overall confidence level $1 - \alpha$ and calculate the per-point confidence $1 - \alpha_{\text{pp}}$ using the Bonferroni correction based on m'/R intervals.

(4) Generate the multivariate traces by combining the leakage samples as explained above.

(5) For each point in the preprocessed trace, calculate the confidence interval for the difference in means according to chapter 3.

(6) Calculate the overall leakage bounds using the global maxima of the lower and upper bounds.

Steps 2-6 can be repeated for different complexity reduction algorithms and parameters in order to obtain a more complete view of the possible leakage of the device.

4.3.1 Efficient Computation

The assessment of multivariate leakage is inherently computationally expensive if performed in a black-box setting, i.e., without the assumptions about POIs. This stems from the fact that an exhaustive d-th order multivariate leakage evaluation of n traces consisting of m samples each requires asymptotically $\mathcal{O}(n \cdot m^d)$ operations in order to calculate the leakage for all d-tuples in the trace. As SCA traces can often contain more than 10^5 samples and more than 10^6 traces are typically collected, exhaustive evaluation becomes very resource intensive even for a bivariate analysis. Therefore, the computational complexity can easily outweigh the measurement complexity which is proportional to $\mathcal{O}(n \cdot m)$. In order to address these issues we study several techniques that can reduce the constants driving the computational load. Another approach to speed up the computation of the multivariate statistical moments is depicted in Appendix A.1.

Averaging over Traces (AoT).

When observing and processing side-channel information of cryptographic implementations, evaluators and attackers alike have to cope with measurement noise that affect evaluation or attack quality.

We can model the leakage observed by an attacker or evaluator at each sample point as $L = L_{\text{data}} + L_{\text{noise}}$, where L_{data} depends on the input data of the target device and L_{noise} is independent of it. Hence, in order to understand how averaging over repeated measurements can improve the evaluation result, we may decompose the measured noise into algorithmic noise $L_{\text{noise-a}}$ and non-algorithmic noise $L_{\text{noise-na}}$. Algorithmic noise captures all dependence of the leakage on data processed by the target that is independent of the input varied by the evaluator during TVLA measurements. When evaluation masked implementations, different sharings of the same unmasked input data are not treated as different inputs as an attacker usually does not have access to them. Therefore, $L_{\text{noise-a}}$ mainly includes noise resulting from different masks and other random values needed by the target implementation. Consequently, $L_{\text{noise-na}}$ models all noise which is not captured by $L_{\text{noise-a}}$. This includes electrical noise in the target device

and measurement system, quantization noise due to digitization and any environmental noise sources, e.g. resulting from temperature changes or radiation.

In order to reduce the non-algorithmic noise, it is possible to record multiple power traces for every computation of the target device with identical inputs, masks and randomness. These can then be averaged before any further analysis to reduce the number of (averaged) traces n_{avg} needed for a meaningful analysis. As described in [Sta18], this preprocessing leads to a quantitative disconnection between the security perceived by an attacker (without access to masks) and the measured security when t-test-based TVLA is used. The disconnection stems from the fact that averaging the signal and (uncorrelated) noise using N measurements increases the SNR by a factor[2] of $\sqrt{N}$, reducing number of traces required for leakage detection, while an attacker does not have this advantage. However, confidence intervals do not suffer from this problem as they do not use the t-statistic (or associated p-value) directly as a metric but rather the upper and lower bounds of a confidence interval. While the size of the confidence interval decreases with averaging and therefore the accuracy increases, this does not influence the evaluation result. Thus, averaging by combining N traces can decrease the number of traces n_{avg} needed for the multivariate evaluation, and therefore the computational complexity while increasing the measurement complexity resulting in a trade-off between those two. However, note that the increase in SNR from averaging usually can not be determined before the measurement unless $L_{\text{noise-na}}$ and $L_{\text{noise-a}}$ can be characterized. When assuming that all $L_{\text{noise-na}}$ are i.i.d., independent of $L_{\text{noise-a}}$, and following normal distributions, their distributions can be estimated by collecting several traces while keeping the algorithmic noise at zero, i.e., with constant masks and randomness. Unfortunately these assumptions do not tend to hold in practice.

The choice of the averaging factor therefore depends on the concrete implementation (total clock cycles, clock frequency), the measurement system (sample rate, acquisition speed) and computational resources available to the evaluator and needs to be adjusted accordingly for the fastest evaluation. As a rule of thumb, faster measurement systems, higher evaluation orders and lower computational resources justify higher averaging factors.
In order to improve the readability of this document we will use the full term *averaging over traces* or the abbreviation AoT in the following sections to discriminate averaging over traces from averaging over samples described below.

Dimensionality Reduction.

Besides reducing the number of traces n, another approach to reduce the computational complexity of multivariate analysis lies in the reduction of the number of samples m per trace. As shown above, a reduction of m by a factor of R will reduce the computational complexity by a factor of $\mathcal{O}(R^d)$. To this end, we evaluate several methods to reduce the number of samples in each clock cycle of the target device. In order to retain the ability to locate POIs when reducing m, which in practice often corresponds to detecting potentially leaking clock cycles, our dimensionality reduction approaches try to preserve as much information about clock cycles as possible. Therefore, we focus on methods that reduce the number of samples per clock cycle while keeping the relations between cycles. This is assured by partitioning the unprocessed trace into segments corresponding to clock cycles and performing the dimensionality reduction on these segments. Each segment starts with the sample point at which the first transistors in

[2] This is only valid when ignoring $L_{\text{noise-a}}$.

the respective clock cycle start switching, i.e., with the clock edge usually visible in the trace. We compare the algorithms described in the following sections by keeping the number of resulting points constant for each of them. This results in a fair comparison in terms of computational complexity.

While the primary objective of the dimensionality reduction is the management of the computational complexity of the multivariate evaluation, the algorithms described below can also change the sensitivity of the analysis depending on parameter choices. As there is no a-priori indication of what information of the leakage traces can help in a multivariate analysis, a reduction of sample points could lead to loss of such information and thereby reduce the detection performance. On the other hand, a reduction of noise and (linear) combinations of leaking samples in the dimensionality reduction could lead to increased leakage detection sensitivity.

These effects are certainly not desirable when trying to quantify the leakage of a device. However, note that all of the preprocessing algorithms described below only use information which is available to any attacker as well. An attacker therefore can also apply these techniques to increase the success rate and/or reduce the complexity of a multivariate attack. We therefore argue that increased detection sensitivity through dimensionality reduction is not a downside of our approach but rather a potential attack vector that should be considered in any leakage assessment procedure.

On the other hand, the possible reduction of assessment power through preprocessing is a more serious problem for an evaluator. When selecting a reduction heuristic and its parameters there is no guarantee that the resulting evaluation will find all leaking tuples of clock cycles. While there is an optimal dimensionality reduction given a perfect leakage model [BGH+15] an evaluator does typically not have access to this model. Still, an attacker does face the same problem. A computationally constrained evaluator should therefore act economically by choosing the parameters resulting in the lowest complexity first. Due to the d-th order polynomial computational complexity of multivariate evaluations, one evaluation with a reduction factor of R needs approximately the same recourses as x^d evaluations with a reduction factor of $x \cdot R$. The overhead of the reduction approaches is small compared to the calculation of the confidence intervals as all reduction algorithms described below have approximately linear complexity in the number of samples and traces. We therefore encourage the application of multiple preprocessing algorithms with high R instead of performing an evaluation using a single algorithm with lower R. If the resulting assessment bounds the leakage to sufficiently small values and additional resources are available to the evaluator, subsequent evaluation runs can be performed with lower reduction factors on the same measurements. If, on the other hand, the relevant leakage threshold is exceeded, the evaluation can be stopped and the design must be rejected as insecure.

Downsampling. A trivial, simple, and computationally inexpensive approach to reduce the number of samples m can be achieved by dropping all but every R-th sample in each cycle. The result is similar to reducing the sample rate of the measurement system by a factor of R. However, we would like to note that even this simple dimensionality reduction algorithm may decrease or increase the detection sensitivity of the evaluation depending on the measured signal. In fact, a decrease in detection sensitivity can be obviously attributed to the (accidental) dismissal of leakage samples. On the other hand, a reduction of the size of the preprocessed trace m' leads to a higher corrected per-point confidence level $1 - \alpha_{\text{pp}}$ resulting in tighter confidence

intervals and therefore better leakage detection compared to a full evaluation. Typically, the latter effect is less pronounced in the similar case of a reduced sample rate of the acquisition system instead of post-measurement downsampling. This is because the jitter in the trigger response of some oscilloscopes depends on the sample rate[3], leading to increased jitter with lower sampling rates. Therefore, downsampling should be preferred to reducing the sample rate even if this is the only dimensionality reduction method used if the additional measurement complexity can be tolerated.

Averaging over Samples. This paragraph studies oversampling, the averaging over adjacent samples, in contrast to the averaging described in above. The reduced trace is computed by partitioning the samples corresponding to a clock cycle into sets and computing the average of each set, effectively creating a low-pass filtered and downsampled version of the leakage trace:

$$\bar{L}_i = \frac{1}{R} \sum_{j=i\cdot R}^{(i+1)\cdot R} L_j. \tag{4.1}$$

In contrast to using pure downsampling, an analysis using averaging can result in more leakage being detected compared to a full-trace analysis even when disregarding the improvement in per-point confidence. Averaging over samples increases the SNR by averaging out uncorrelated noise similar to averaging over traces as described above. However, this preprocessing technique can also reduce the leakage signal if the leakage contains frequency components that are higher than the resulting sample rate of the processed traces.

1-Norm. The 1-Norm calculates the sum of absolute values as:

$$\|L_i\|_1 = \sum_{j=i\cdot R}^{(i+1)\cdot R} |L_j|. \tag{4.2}$$

By summing over absolute values the potential signal reduction described in above can be avoided. However, for the same reason there is no increase in SNR. Note that the proportionality of the result to R prevents direct comparisons of results with different reduction factors.

2-Norm. The 2-Norm, or euclidean norm, is calculated as:

$$\|L_i\|_2 = \sqrt{\sum_{j=i\cdot R}^{(i+1)\cdot R} L_j^2}. \tag{4.3}$$

The resulting trace $\|L\|_2$ is related to the energy of the part of the measured signal between time points $i \cdot R$ and $(i+1) \cdot R$. Using the 2-Norm instead of the energy results in the same scale in confidence intervals computed from reduced traces when compared to raw traces.

[3] If no countermeasures such as trigger interpolation are employed.

Principal Component Analysis. PCA is a statistical tool commonly used to reduce the dimensionality of a data set. This is achieved by projecting the input data to space spanned by an orthonormal basis with linearly uncorrelated dimensions. The data reduction is then performed by dropping all dimensions but the once containing the maximal variances of the data. In the context of cycle-wise reduction of the number of samples, a PCA is be performed on every cycle separately and only the $m/R \cdot c$ most important dimensions are retained, where c is the number of cycles in a trace. Note that these dimensions can differ for different clock cycles, requiring separate PCAs for each cycle. As shown in [BHvW12], the dimensions containing the most variance do not necessarily contain the most side-channel leakage, but rather the most noise. This can be especially pronounced when evaluation protected implementations where the SNR can be very low. Therefore, noise reduction as described above will often benefit this preprocessing technique. In general, the optimal reduction factor is hard to estimate before an analysis has been performed.

4.4 Case Study: Sequential PRESENT

This section provides practical insights on how our assessment framework performs with different algorithms and parameter choices and on how to choose these parameters. We limit our analysis to the bivariate case in order to keep the computational requirements reasonable and to aid the visualization of our results.

4.4.1 Target Architecture

In order to evaluate our proposed multivariate leakage assessment scheme, we chose a hardware implementation of the PRESENT block cipher on an FPGA. The implementation was protected against first order side-channel attacks using a three-share threshold implementation [NRR06] with a quadratic decomposition of the cubic S-boxes similar to [PMK+11]. The evaluation of the resulting six component functions of the S-box was stretched into twelve clock cycles in order to avoid univariate second order leakage. This was achieved by placing a register stage before and after each layer of component functions. Each register was then sampled in a separate clock cycle and after the evaluation of each stage the other was reset (sequentially) in order to avoid transitional leakage between rounds. Note that this implementation was not designed to be efficient but rather to limit and prevent univariate leakage while having detectable multivariate leakage.

4.4.2 Measurement Setup

To measure the power consumption of the target design we used a dedicated Sakura-G SCA assessment board equipped with two Spartan-6 FPGAs - one FPGA generating (masked) input, key and randomness material, the other executing the PRESENT cipher on this data. The system randomly interleaves fixed and random plaintexts following the usual TVLA procedure from [SM15].

The design was clocked at 4 MHz. To capture the power consumption we used the internal 14 dB gain amplifier of the Sakura-G board, and additionally amplified the signal with an external zfl-1000+ [Min] low-noise amplifier with a gain of 20 dB to use the full amplitude of

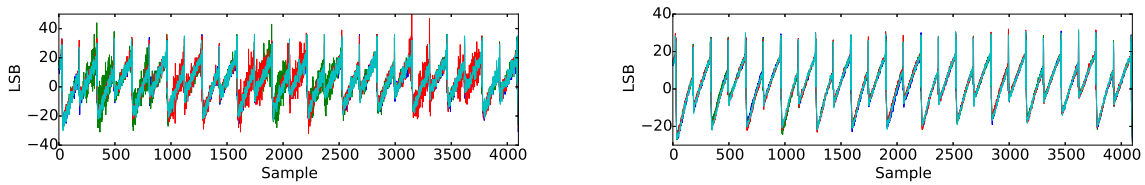


Figure 4.1: Overlay of four sample traces of the sequential PRESENT implementation without (left) and with 16-fold AoT (right).

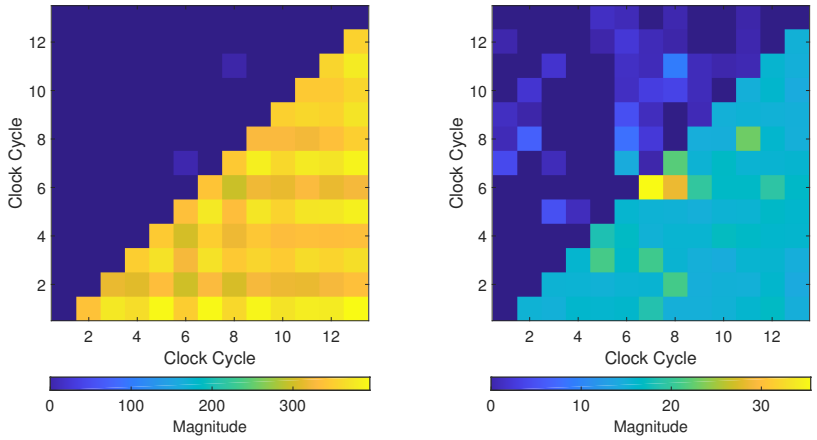


Figure 4.2: Per-cycle confidence interval maps for sequential PRESENT without (left) and with 16-fold (right) AoT. Lower bounds are in upper left half including the diagonal, upper bounds in the lower right half.

the digitizer. The traces were collected with a Spectrum Instrumentation M4 PCIe digitizer and a 400 MHz analog bandwidth limiter was enabled in order to reduce noise.

For this first case study we capture 1M traces at a sample rate of $1.25\,\mathrm{GS\,s^{-1}}$, each with a duration of 12 µs, yielding 15 040 samples per trace. As a result, one clock cycle of the target design covers 312.5 samples. We then analyzed one round of the cipher, taking 13 clock cycles, thus limiting our subsequent analysis to a window of 4062 samples.

4.4.3 Parameter Space Exploration and Results

This section shows the influence of the parameters of the preprocessing algorithms and displays the advantages and disadvantages of each algorithm.

Averaging over Traces (AoT).

The measurements are performed with averaging factors of 1 and 16. When comparing both cases, we keep the number of traces *after averaging* constant, e.g., 16-fold AoT requires 16 times the number of measurements when compared to an evaluation without AoT. However, this method results in the same computational complexity for the evaluation after the first AoT step. Figure 4.1 depicts four sample traces of the measured power consumption for both AoT factors. As expected the trace acquired with 16-fold AoT is notably cleaner.

The confidence interval maps with and without AoT are shown in Figure 4.2.[4] The upper left half of the plot shows the lower bounds while the upper bounds are displayed in the lower right half. The diagonal (corresponding to a univariate analysis) contains the lower bounds as well. For improved readability, the plots only show the maximum values corresponding to each clock cycle.

Note that the analysis of the traces that were not averaged can only identify two combination of cycles ($(6, 7)$ and $(8, 11)$) where leakage can be detected. Furthermore, the upper bounds of the intervals are very large compared to the detected leakage. The highest detected leakage is $\gamma_{\min} = 7.15\,\mathrm{mV}^2$ and the highest upper bound is $\gamma_{\max} = 1638.5\,\mathrm{mV}^2$ producing a ratio of $\frac{\gamma_{\min}}{\gamma_{\max}} = 1 \cdot 10^{-5}$. This result signifies low statistical power of the evaluation and would call for more measurements in an assessment of a target device.

As expected, AoT improves the evaluation performance significantly. The result of the averaged traces show a closer estimation of the leakage present in the target resulting in 37 leaking cycle combinations being identified. In this scenario, the leakage is assessed to be between $\gamma_{\min} = 35.28\,\mathrm{mV}^2$ and $\gamma_{\min} = 87.26\,\mathrm{mV}^2$, resulting in a relative interval ratio of $\frac{\gamma_{\min}}{\gamma_{\max}} = 0.40$. These metrics and the metrics for the following cases are depicted in Table 4.1.

Dimensionality Reduction.

We perform all reduction algorithms described in section 4.3.1 with different numbers of resulting samples per cycle and therefore different reduction factors R. We chose 1, 4, and 16 target dimensions per cycle, as starting with a minimal target dimension count is the economic choice when having limited computational power.

Downsampling. Figure 4.3 shows the assessment using the downsampling reduction approach. Note that the analysis without averaging over traces can only identify one tuple of points as leaking when keeping 16 samples per clock cycle. In the other two cases, no leakage is detected. When AoT is performed first, 11 to 31 tuples are detected, depending on the number of kept samples. An interesting observation can be made when comparing the result from the non-reduced and downsampling reduced evaluation in the AoT case: the latter finds some tuples to be sensitive which the former does not. This can be attributed to the resulting different per-tuple confidence level as explained in section 4.3.1.

Averaging over Samples. The result using the averaging reduction is depicted in Figure 4.4. Note that the difference between the cases with and without averaging over traces is comparatively small. This can be explained by the partially redundancy of the SNR improvement through averaging over traces which is achieved in a similar way by averaging points within a clock cycle. In both cases the minimal detected leakage increases with increasing dimensions but the relative size of the global interval remains approximately constant.

1-Norm. Figure 4.5 shows the results when applying the 1-norm reduction. In comparison to the other parameter choices, reducing each cycle to one point yields very poor results. In the non-AoT case, no tuple can be detected at all. Note that increasing the target dimension from 4

[4] All remaining figures depicting confidence interval maps are constructed by combining the lower and upper bound of the intervals into a single figure.

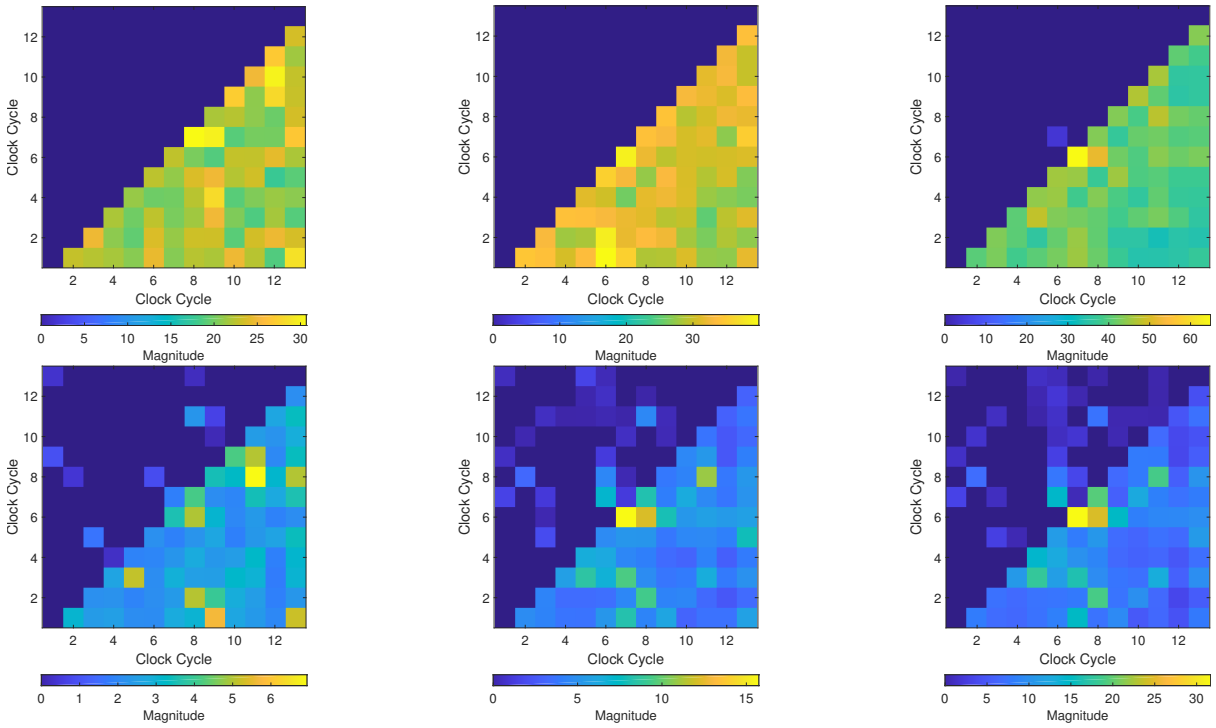


Figure 4.3: Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with downsampling reduction to 1, 4 and 16 (left to right) samples per cycle.

to 16 does not improve the assessment quality but reduces the number of detected tuples (only without AoT) and the interval ratio (both cases).

2-Norm. The results for the 2-norm reduction are depicted in Figure 4.6. As the difference between disabled and enabled AoT is similar to the 1-norm case, only the plot with 16-fold AoT is shown. Similarly, an increase of dimensions from 4 to 16 does not improve assessment quality.

Principal Component Analysis. Figure 4.7 shows the evaluation using the PCA dimensionality reduction algorithm. In contrast to all other algorithms, the number of detected tuples and the interval ratio decrease when increasing the number of dimensions even when starting from one when applied without prior AoT. This can be attributed to the most informative principal components being the first ones. Therefore, adding more components primarily decreases statistical power.

4.4.4 Discussion

As expected, all algorithms benefit from averaging over traces before performing a dimensionality reduction and the subsequent multivariate analysis. Therefore, if control of the masks and randomness used in the target device is possible and the trade-off between measurement and

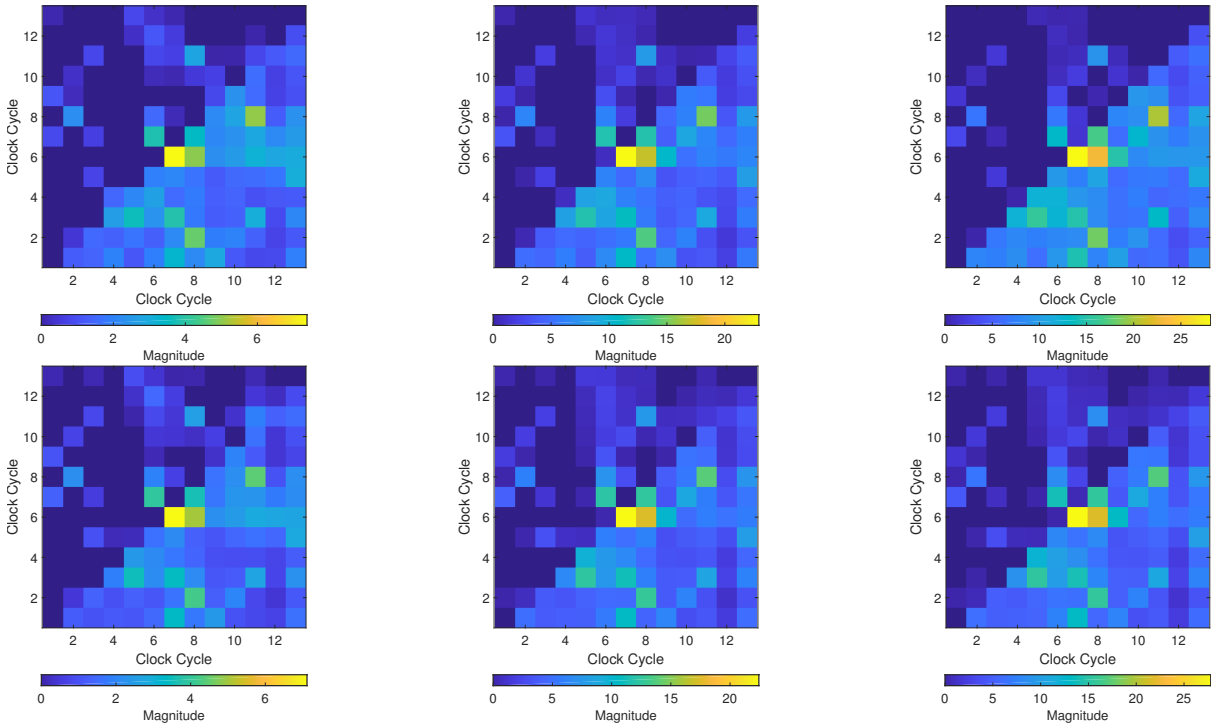


Figure 4.4: Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) AoT with averaging reduction to 1, 4 and 16 (left to right) samples per cycle.

computational complexity is favorable, averaging over traces should be used.
In the AoT case, not all tuples that are detected as leaking by the full analysis are also recognized with the reduction algorithms (e.g., tuple $(1, 12)$). However, with or without AoT, for all reduction approaches (except for downsampling) the number of detected tuples is higher than with a full evaluation for at least some parameters. In particular, downsampling performs the worst in terms of detected tuples and interval ratio in comparison to all other approaches when the same reduction factor is considered. Still, downsampling and averaging benefit from a lower reduction factor and therefore a higher number of resulting samples per clock cycles. The PCA, 1- and 2-norm reductions show diminishing returns or even decrease in performance when the dimensionality is increased.
It is important to note that all reduction approaches find leaking clock cycle tuples that are not detected by an evaluation without reduction and the detected tuples do not completely overlap for all parameters signifying the importance of a battery of reduction algorithms. As the complexity of the evaluation grows exponential in the dimensionality using higher reduction factors first when allocating computational resources is justified.

4.5 Case Study: AES-DOM

In the second case study we chose a more practical implementation in order to show the real-world applicability of our framework. The evaluation is restricted to the bivariate case for the

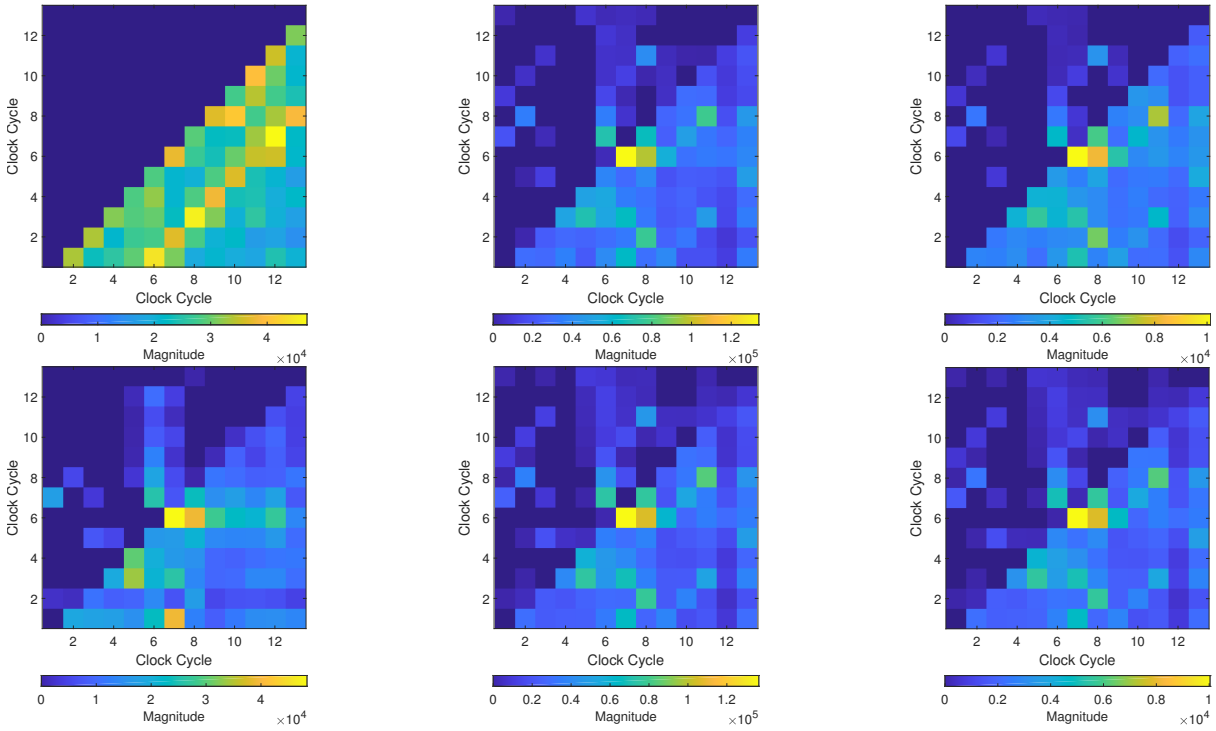


Figure 4.5: Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with 1-norm reduction to 1, 4 and 16 (left to right) samples per cycle.

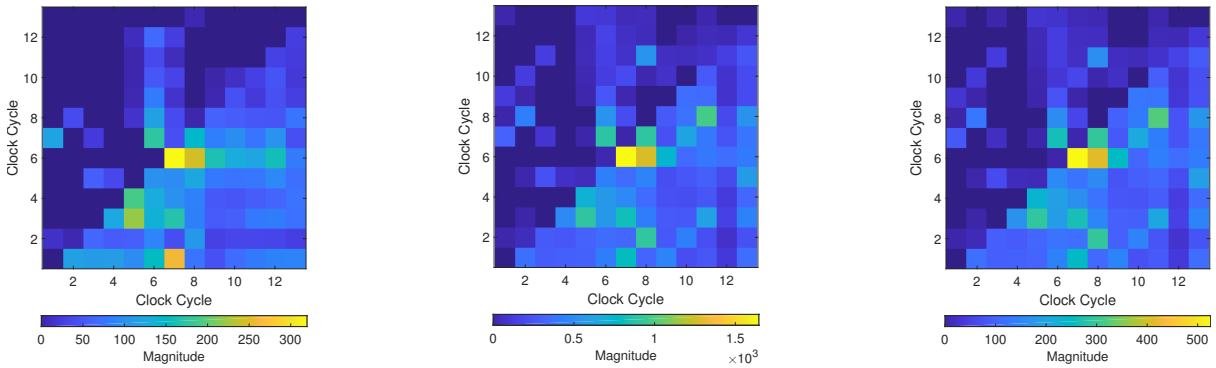


Figure 4.6: Per-cycle confidence interval maps for sequential PRESENT with 16-fold averaging over traces with 2-norm reduction to 1, 4 and 16 (left to right) samples per cycle.

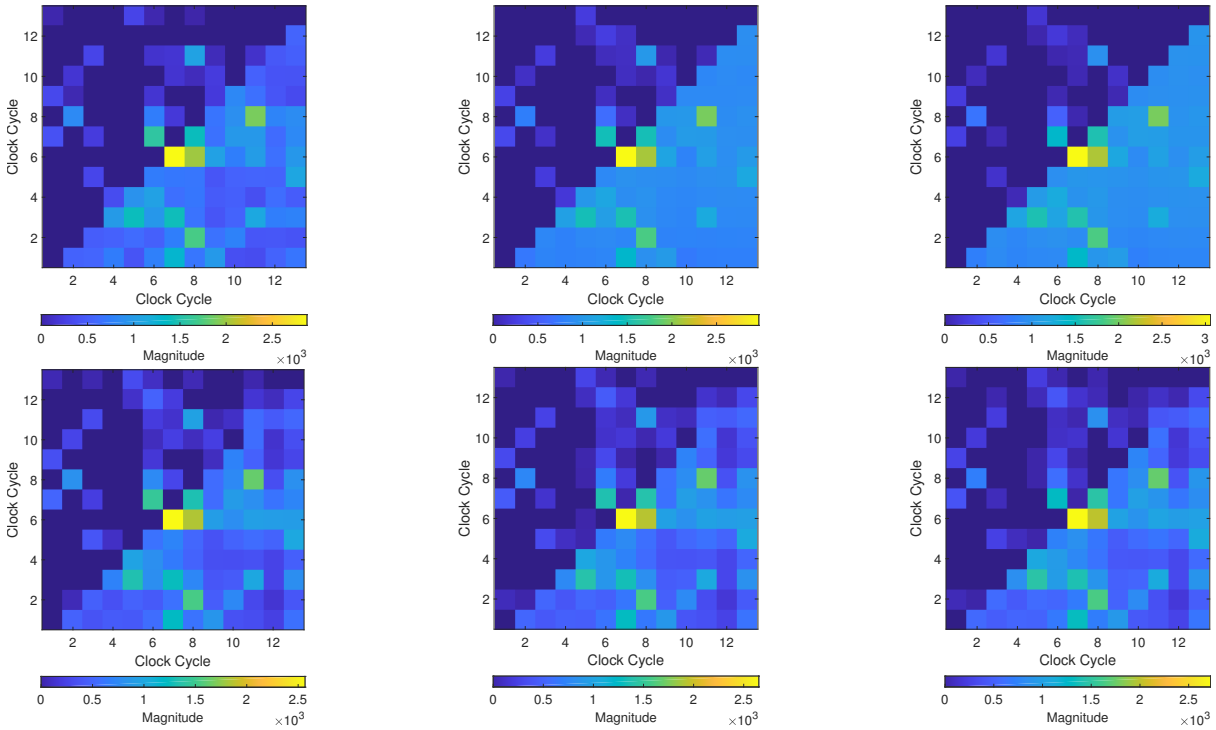


Figure 4.7: Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with PCA reduction to 1, 4 and 16 (left to right) samples per cycle.

same reasons as in the first case. The parameters were discovered analogous to the PRESENT case study, therefore we omit the detailed derivation of the parameters. While we still did perform an analysis without averaging over measurements, we omit the results here because they (unsurprisingly) confirm that AoT improves evaluation performance.
The measurement setup for this case study was the same as for the PRESENT target. The only changes lie in adjusted gain of the amplifier and oscilloscope and a different traces length. As above, 1 M measurements where collected. We evaluated 6250 samples corresponding to the first AES round.

4.5.1 Target Architecture

We use the configurable and publicly available DOM implementation of the AES [Gro16b] as a second benchmark for our proposed SCA evaluation framework. In general, the DOM scheme [GMK16b] provides provable d-th order security using $d+1$ shares by splitting computations into $d+1$ domains. When information from different domains is combined in non-linear functions, randomness and register stages are added to ensure uniformity and prevent glitches respectively. We chose the 2-share variant providing resistance against first-order attacks. The implementation takes 20 cycles per round and requires additional randomness which we created using LFSRs on the target device.

Table 4.1: Evaluation results for the masked sequential PRESENT implementation. Confidence interval bounds are in mV^2.

Algorithm	Dim.	*1-fold averaging over traces* Confidence Intervals			Tupl.	*16-fold averaging over traces* Confidence Intervals			Tupl.
		$[\gamma_{min}]$	$[\gamma_{max}]$	$[\frac{\gamma_{min}}{\gamma_{max}}]$		$[\gamma_{min}]$	$[\gamma_{max}]$	$[\frac{\gamma_{min}}{\gamma_{max}}]$	
full	312.5	7.15	1638.5	0.00	2	35.28	87.26	0.40	37
down-	1	0.00	178.42	0.00	0	5.62	24.71	0.23	11
sampling	4	0.00	236.74	0.00	0	17.29	38.43	0.45	27
	16	9.72	283.86	0.03	1	35.69	77.32	0.46	31
averaging	1	9.33	18.02	0.52	32	9.62	17.36	0.56	37
	4	29.25	55.98	0.52	38	29.79	54.74	0.54	49
	16	32.05	70.92	0.45	28	34.23	67.82	0.50	50
1-Norm	1	0.00	$2.49 \cdot 10^5$	0.00	0	$6.27 \cdot 10^4$	$1.20 \cdot 10^5$	0.52	28
	4	$7.02 \cdot 10^4$	$1.34 \cdot 10^5$	0.52	33	$1.71 \cdot 10^5$	$3.32 \cdot 10^5$	0.54	49
	16	$1.15 \cdot 10^4$	$2.51 \cdot 10^4$	0.46	25	$1.24 \cdot 10^4$	$2.44 \cdot 10^4$	0.50	49
2-Norm	1	0.00	1158.98	0.00	0	430.44	780.08	0.55	29
	4	1980.62	3899.00	0.51	27	2186.89	4016.97	0.54	50
	16	563.89	1296.39	0.43	22	647.66	1283.40	0.50	50
PCA	1	3873.02	7016.89	0.55	26	3564.50	6264.84	0.57	35
	4	3633.47	7256.47	0.50	25	3358.74	6470.63	0.52	40
	16	3420.09	7469.82	0.46	22	3175.46	6653.88	0.48	40

4.5.2 Results and Discussion

In this section, the evaluation results from the application of our framework to the AES-DOM target are discussed. Again, we used all dimensionality reduction approaches and parameters as in section 4.4. The figures and a table depicting the evaluation results are provided in Appendix A.2 in Figure A.1 and Table A.1.

Interestingly, there are some notable differences between this evaluation and the previous case study. First, no reduction approach detects as many leaking tuples as the full analysis. However, the reduction algorithms outperformed the full analysis in terms of interval ratio for some parameter choices. Additionally, the battery of reduction algorithms detects leaking tuples that are not identified by the full analysis as in the previous case study. Secondly, for all algorithms except for PCA, more dimensions improve the result. In the PCA-case the increase in number of detected tuples when changing the number of dimensions per cycle from one to four is comparatively large but the leakage estimation performance is reduced when using 16 dimensions. Thirdly, the strongest detected leakage is univariate but many multivariate tuples leak as well. Finally, downsampling is not the worst algorithm in general anymore.

4.6 Performance Evaluation.

The measurement speeds on our Spectrum Instrumentation M4 PCIe digitizer for both case studies are provided in Table 4.2. In the cases where averaging over traces was enabled it was performed simultaneously and the figures refer to the number of traces after averaging. The PRESENT and AES traces consist of 15 040 samples and 19 552 samples respectively and where cropped to one round after the measurement. The measurement speed is faster for the AES

although its trace contains more samples because the assessed implementation takes less cycles. In the 16-fold AoT case there is reduced disk load on the measurement system and no inter-FPGA communication overhead between acquisitions with the same sharing and randomness resulting a slowdown factor of only 5.8 and 5.6 respectively when compared to the baseline.

Table 4.2: Measurement performance for the PRESENT and DOM-AES case studies.

	PRESENT		**DOM-AES**	
	no AoT	*16-fold AoT*	*no AoT*	*16-fold AoT*
Traces/second	3095	530	5000	895
Megasamples/second	46.5	8.0	97.8	17.5

Table 4.3 provides the performance figures for the generation of the statistical moments of the bivariate traces for the PRESENT and DOM-AES case studies. The PRESENT traces capture 13 cycles and the DOM-AES traces 20 cycles consisting of 312.5 samples each. An Nvidia Tesla P100 GPU was used to perform the calculations on multiple points in parallel. For the chosen parameters, bottlenecks of the evaluation platform (e.g. disk access times and throughput) as well as better utilization of the GPU by larger traces overlay the quadratic complexity in the effective number of samples. Still, an evaluation without dimensionality reduction takes longer than four evaluations with all three reduction factors in the PRESENT case. For the longer AES, the full evaluation takes 7.4 times as long as an evaluation with all exemplary reduction factors. Therefore, it is faster to perform all five reduction approaches with one, four and 16 samples per cycles than one full evaluation.

Performance figures for the pre-processing steps are omitted because of their linear complexity[5] in the number of samples per trace. Even an unoptimized Python implementation outperforms the moment generation by at least an order of magnitude. Similarly, the final calculation of the confidence intervals from the statistical moments, while quadratic in the number of samples, is constant in the number of traces and can therefore be neglected.

4.7 Conclusion

In this work we proposed a novel side-channel leakage assessment framework based on confidence intervals that can be used to analyze multivariate leakage of protected implementations. The evaluation results can attest the presence as well as the absence of leakage and are robust

[5]Finding PCA projections is cubic in the number of samples per cycle and linear in the number of cycles but only needs a constant number of traces. The impact is therefore negligible.

Table 4.3: Evaluation performance for the PRESENT and DOM-AES case studies in traces per second.

Dim./cycle	1	4	16	312.5
PRESENT	2210	1955	1546	147
DOM-AES	2233	1915	1352	79

against noise in the measurement system. The feasibility of our approach is supported by a set of pre-processing algorithms that reduce the number of traces required in the evaluation and the number of tuples that need to be analyzed. We accomplish the former by averaging traces which provides strong benefits in the multivariate setting while tying in naturally with a confidence-interval-based assessment. The latter is achieved by reducing the effective dimensionality of each clock cycle allowing to retain cycle accurate POI information. While the parameter choice for the reduction algorithms is heuristic we provide a clear guideline for this choice when challenged by finite computational resources.

Part III

Side-Channel Resistance for Cryptographic Primitives

Chapter 5

High-Speed Masking for Polynomial Comparison in Lattice-based KEMs

With the National Institute of Standards and Technology (NIST) post-quantum standardization competition entering the third round, the interest in practical implementation results of the remaining NIST candidates is steadily growing. Especially implementations on embedded devices are often not protected against side-channel attacks, such as differential power analysis. In this regard, the application of countermeasures against side-channel attacks to candidates of the NIST standardization process is still an understudied topic. Our work aims to contribute to the NIST competition by enabling a more realistic judgment of the overhead cost introduced by side-channel countermeasures that are applied to lattice-based Key Encapsulation Mechanisms (KEMs) that achieve CCA-security based on the Fujisaki-Okamoto transform. We present a novel higher-order masking scheme that enables an efficient comparison of polynomials as previous techniques based on arithmetic-to-Boolean conversions renders this (generally inexpensive) component extremely expensive in the masked case. Our approach has linear complexity in the number of shares compared to quadratic complexity of previous contributions and it applies to lattice-based schemes with prime modulus. It comes with a proof in the probing model and an efficient implementation on an ARM Cortex-M4F microcontroller which was defined as a preferred evaluation platform for embedded implementations by NIST. Our algorithm can be executed in only 1.5-2.2 milliseconds on the target platform (depending on the masking order) and is therefore well suited even for lightweight applications. While in previous work, practical side-channel experiments were conducted using only 5,000 - 100,000 power traces, we confirm the absence of first-order leakage in this work by collecting 1 million power traces and applying the t-test methodology. The work described in this chapter was published at CHES 2020 [BPO+20].

Contents of this Chapter

5.1 Introduction **60**
5.2 Preliminaries **61**
5.3 Higher-Order Masking of Comparison of Polynomials **65**
5.4 Microcontroller Implementation **72**
5.5 Results **75**
5.6 Conclusions and Future Work **79**

5.1 Introduction

As shown in previous chapters, cryptographic implementations on embedded devices often have to consider countermeasures against side-channel attacks such as timing attacks, power analysis, or fault injections. Remedies against timing attacks are usually simpler to realize, however, there are exceptions, like Gaussian samplers [AJS16, SBG+18, KMRV18, KRVV19].

In response to the looming threat of quantum computers rendering most contemporary public key crypto obsolete, NIST started a standardization competition for post-quantum cryptography that has entered the second round in early 2019. NIST explicitly mentions the side-channel security of schemes as one of their evaluation criteria [NIS16]. Among the remaining 17 encryption schemes and Key Encapsulation Mechanisms (KEMs) in round two, there were 9 lattice-based KEMs, which constitute the largest group of KEMs. Many of these schemes use the Fujisaki-Okamoto transform [FO99] or a variant [TU16] to achieve CCA-security. One of those schemes – KYBER – was selected as a third round NIST finalist in July 2020. A crucial component of the transform is the comparison of outputs. The comparison component has been widely disregarded in side-channel analysis of post-quantum cryptography so far as the performance cost for this component is negligible in the unmasked case. In this work, we show that when masking is applied using a conventional approach, the comparison step actually amounts for a sizable number of clock cycles in the execution of the KEM. We therefore propose a novel and highly efficient higher-order masked algorithm for the comparison component that solves the aforementioned problem and works on lattice based schemes with prime modulus. Our solution significantly outperforms the previous approach by one to two orders of magnitude, depending on the masking order. Our masking scheme is applicable for the NIST post-quantum standardization candidates `NewHope` [ADPS16], `Kyber` [BDK+18], and `LAC` [LLZ+18].

5.1.1 Related Work

Applying masking to schemes based on the Ring-Learning with Errors (Ring-LWE) problem as countermeasure to power analysis has been first analyzed by Reparaz et al. in [RRVV15, RdCR+16, RRdC+16]. Their masking proposals however only protect schemes providing security against chosen-plaintext attackers (CPA). In a CPA-secure Ring-LWE-based scheme, no comparison is required. Previous approaches for masked polynomial comparison for lattice-based cryptography have been proposed by Oder et al. [OSPG18], Barthe et al. [BBE+18], and Migliore et al. [MGTF19]. The approach from [OSPG18] also targets Ring-LWE-based schemes and explicitly takes CCA-security into account, but is optimized for first-order masking only and cannot be adapted to higher orders. The approaches from [BBE+18] and [MGTF19] are masking schemes for lattice-based digital signatures. The comparison algorithms in both works are alike and both are based on conversion algorithms that transform arithmetic shares to Boolean shares. These transformations are computationally extremely expensive and have a quadratic complexity regarding the number of shares [SPOG19]. This makes the polynomial comparison an unnecessary overhead in masked implementations of lattice-based cryptography. Several contributions performed practical experiments to verify the side-channel security of their proposals. However, the extensiveness of these experiments is lower compared to this work with only 5k traces in [RdCR+16], 10k traces in [MGTF19], and 100k in [OSPG18].

Our approach relies on reducing the comparison of multiple values to one comparison of a function these values. Similar approaches have been applied in other contexts than mask-

ing, e.g., batch verification of the equality of logarithms [APB+04]. However, to the best of our knowledge, our approach is the first to utilize this basic principle to speed up the secure comparison of masked values.

5.1.2 Contribution

In this work, we present the first higher-order masking algorithm for polynomial comparison that is specifically optimized for usage in lattice-based KEMs. The contributions in this chapter are as follows:

- We show that the adaption of other approaches (from lattice-based signatures) introduces a significant computational overhead. That is because the A2B conversions used in this naive approach are expensive and have a quadratic complexity in the masking order.
- We present a new algorithm that has a simpler structure, better asymptotic run time, less constant overhead, and is still provable secure for higher masking orders. Our approach exploits that the comparison steps reduces the entire information contained in a polynomial to just a single bit.
- A theoretical proof in the $t-$probing model of the new scheme is provided. In particular we show that it satisfies the stronger property of NI.
- Our developed ARM Cortex-M4F assembly implementation shows the superior practical performance of our approach as it is able to reduce the computational cost of the comparison step by a factor of 16 for first-order masking security and even more for higher orders.
- The side-channel security of our approach is evaluated in practical experiments with 1 million power traces that is significantly more than the number of power traces used in previous work on power analysis attacks on lattice-based cryptography.

Our algorithm can be applied to multiple lattice-based KEMs that are submitted to the NIST post-quantum standardization competition and with the publication of this work, we will make the source code of our microcontroller implementation publicly available.

5.1.3 Outline

This chapter is organized as follows: In Section 5.2, the necessary theoretical background that is crucial for the understanding of this chapter is introduced. After that, we present our novel approach for higher-order masked polynomial comparison in Section 5.3. The implementation of this algorithm is discussed in Section 5.4. In Section 5.5, we present the results of our evaluation of the algorithm. Finally, we draw a conclusion in Section 5.6.

5.2 Preliminaries

In this section, we introduce the necessary theoretical background. This includes a description of the `NewHope` KEM, a definition of side-channel security, and the masking countermeasure.

5.2.1 Notation

In the rest of the chapter, we denote with q the modulus in the lattice-based scheme, which in this work is always a prime number. We will indicate with k the number of coefficients in a polynomial. Moreover, the lower case will be used for Boolean encoding and the upper one for arithmetic encoding. To refer to the i-th coefficient of a polynomial A, we will write A_i, while we write A_i^j to refer to the j-th share of the i-th coefficient of the polynomial.

5.2.2 The Basic `NewHope` Scheme

We chose `NewHope` as case study to evaluate our comparison algorithm. There are two variants of `NewHope`, one that is secure against chosen-plaintext attackers (CPA-secure) and one that is secure against chosen-ciphertext attackers (CCA-secure). Even though the comparison is only necessary in the CCA-secure scheme, we still review the basic construction of the CPA-secure `NewHope` as it is necessary for the understanding of how our comparison algorithm works. For the sake of simplicity, we omit a lot of details concerning efficiency, like the application of the number-theoretic transform or compression of polynomials, in the description of the key generation, encryption, and decryption in Algorithms 1,2, and 3. The most important parameters of the scheme are the modulus q, the lattice dimension k, and the sampling parameter κ.

Algorithm 1 `NewHope CPA.Keygen`

Input: Public constant polynomial a
Output: Public key pk and secret key sk
1: $seed \xleftarrow{\$} \{0,\dots,255\}^{32}$
2: $r_1, r_2 \leftarrow \texttt{SampleBinomial}(seed)$
3: $p \leftarrow r_1 + ar_2$
4: **return** $pk = (a, p), sk = r_2$

Algorithm 2 `NewHope CPA.Encryption`

Input: Public key pk, message $\mu \in \{0,\dots,255\}^{32}, coin \in \{0,\dots,255\}^{32}$
Output: Ciphertext $A = (c_1, c_2)$
1: $e_1, e_2, e_3 \leftarrow \texttt{SampleBinomial}(coin)$
2: $c_1 \leftarrow ae_1 + e_2$
3: $c_2 \leftarrow pe_1 + e_3 + \texttt{Encode}(\mu)$
4: **return** (c_1, c_2)

Algorithm 3 `NewHope CPA.Decryption`

Input: Secret key $sk = r_2$, ciphertext $A = (c_1, c_2)$
Output: Message μ
1: **return** $\texttt{Decode}(c_2 - c_1 r_2)$

The algorithm `SampleBinomial`() uses a PRNG that is seeded by a random bit string. The PRNG sends two random bit strings of length κ bits to the sampling algorithm. The sampler

then calculates the Hamming weight of both bit strings and subtracts the Hamming weights. The result is a binomial distributed random number. `SampleBinomial`() outputs an entire polynomial with binomially distributed coefficients. The algorithm `Encode`() transforms the input message into a polynomial. Each bit of the message is encoded into four coefficients. This is done by setting these four coefficients to $\{0, 0, 0, 0\}$ if the message bit is 0. If the message bit is 1, the respective coefficients are set to $\{\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor\}$.

Many lattice-based KEMs use a very similar base construction. The main difference to `Kyber` is that the security of `Kyber` is based on the Module-LWE problem while the security of `NewHope` is based on the Ring-LWE problem. The security of `Frodo` is based on the plain LWE problem. This implies that the parameters of `Frodo` are much bigger, but the underlying lattice is actually random and not structured as in the case of the Ring-LWE and the Module-LWE problem. `Saber` and `Round5` are based on the the (Ring-) Learning with Rounding (LWR) problem that is related to LWE.

5.2.3 Fujisaki-Okamoto Transform as Applied to `NewHope`

As many lattice-based KEMs that base their security on the LWE or LWR problems, the original `NewHope` proposal [ADPS16] is only secure against chosen-plaintext attackers. The Fujisaki-Okamoto transform [FO99] is a standard conversion algorithm that many designers of post-quantum cryptography use to turn a CPA-secure scheme into a CCA-secure one. The idea behind the Fujisaki-Okamoto transform is that a re-encryption in the decryption detects whether the input ciphertext was valid or not. Therefore the re-encrypted ciphertext will be compared to the original input and if this comparison fails, a random value will be output by the algorithm. Figure 5.1 depicts how the Fujisaki-Okamoto transform is applied to `NewHope` in the specification of the NIST post-quantum standardization [AAB$^+$]. The ciphertext $\tilde{A}$ is initially decrypted to μ' using the secret key sk and then used as input to the random oracle G. The random oracle outputs a) the seed for the PRNG of the re-encryption coin", b) the (symmetric) key material k', and c) and additional 256-bit value d'. After the re-encryption of μ' using the public key and the PRNG seed, the resulting ciphertext A is compared to the input $\tilde{A}$. Only if this comparison and the comparison of the bit string d' with the input d is true, the key material k' is hashed together with d and $\tilde{A}$ to receive the final symmetric key. Otherwise, k' is replaced by a random value.

As noted in [OSPG18], all intermediate values depending on the output of the initial decryption are sensitive and need to be masked as indicated in Figure 5.1 by bold lines. In particular, this also includes the output of the re-encryption A which is compared to the initial ciphertext $\tilde{A}$. This comparison step is linear in the number of coefficients k and therefore negligible regarding the performance for the unmasked case. When masking is applied to the comparison of these polynomials, the performance cost gets significantly larger. Therefore, this work analyzes how an efficient masking scheme can be applied to this comparison step.

5.2.4 Security against Side Channel Attacks

In order to secure our scheme against side-channel attacks, we adopt the countermeasure of masking, which consists in randomizing the computation of the targeted circuit such that if a bounded amount of information is leaked during the execution, this is statistically independent of the sensitive variable. To this scope, every sensitive variable S is encoded into n shares S^j,

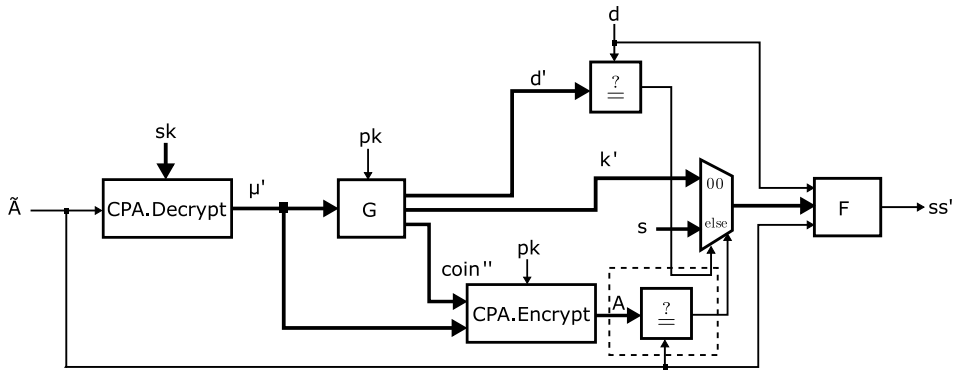


Figure 5.1: IND-CCA-secure variant of the `NewHope` KEM. The dashed line highlights the comparison component that is subject of this work. Bold lines indicate masked data.

such that the sum of all of them gives the original masked variable S but a collection of less than n shares does not allow to reconstruct S. Ishai et al. in [ISW03] formalized for the first time the concept of masking and introduced the so-called *t-probing model*, where an adversary is allowed to access up to t intermediate values in a circuit.

The definition of $t-$probing security requires the existence of a simulator, which can simulate the adversary's view without having access to the sensitive variables, but using only a subset of cardinality at most t of their shares. The first security definition has been enriched later on in [BBD+15] in order to additionally guarantee security also when an algorithm is part of a bigger circuit and ensuing that using the output of a gadget as input to another one does not add sensitive information to the adversary's knowledge. The new security definitions are called $t-$non interference (abbreviated with t–NI) and $t-$strong non interference (abbreviated with t–SNI). Under some circumstances, t–NIgadgets and t–SNIones can be securely composed. Instead, t–SNIgadgets can always be securely composed with each other. The formal definitions are given below.

Definition 1 (t–NI)**.** A given gadget $\mathcal{G}$ is t-Non-Interfering (t–NI), if every set of t probes on the internal and output values can be simulated by using at most t shares of each input.

Definition 2 (t–SNI)**.** A given gadget $\mathcal{G}$ is t-Strong-Non-Interfering (t–SNI), if every set of t_1 probes on the internal values and t_2 probes on the output values, with $t_1 + t_2 \leqslant t$, can be simulated by using at most t_1 shares of each input.

In particular, the last definition requires that the number of probes on the outputs are independent from the number of the shares needed by the simulation.

In the rest of the chapter, n will always refer to the number of shares and t to the security order.

5.2.5 Arithmetic and Boolean Masking

Masking schemes for cryptographic algorithms can apply different forms of masking. When Boolean masking is applied, the shares have to be combined via the XOR operation to reconstruct the secret value.

$$A = \bigoplus_j A^j = A^1 \oplus A^2 \oplus \ldots \oplus A^n$$

In arithmetic masking, this operation is replaced by modular addition

$$A = \sum_j A^j = A^1 + A^2 + \ldots + A^n \bmod q$$

Some components of lattice-based KEMs are more efficient when masked arithmetically, like polynomial multiplication. Other parts, especially symmetric components as used for PRNGs and XOFs, are more efficient when using Boolean masking. Therefore, we need to be able to switch between both forms of masking. Special conversion algorithms have been developed for Arithmetic-to-boolean (A2B) and Boolean-to-arithmetic (B2A) conversion. For this work, A2B conversions are especially relevant. We therefore review in Algorithm 4 the A2B conversion algorithm from [SPOG19] that is specifically designed for arbitrary moduli. The subroutines `Expand`() and `SecAdd`() are given in Appendix A.3.

Algorithm 4 `ConvertA2B` [CGV14]

Input: $(A^i)_{1\leqslant i\leqslant n} \in \mathbb{F}_q$ such that $\sum_i A^i = A \mod q$
Output: $(z^i)_{1\leqslant i\leqslant n} \in \mathbb{F}_2$ such that $\bigoplus_i z^i = A$

1: **if** n=1 **then return** A_1
2: **end if**
3: $(x^i)_{1\leqslant i\leqslant n/2} \leftarrow$ `ConvertA2B`$((A^i)_{1\leqslant i\leqslant n/2})$
4: $(x'^i)_{1\leqslant i\leqslant n/2} \leftarrow$ `Expand`$((x_i)_{1\leqslant i\leqslant n/2})$
5: $(y^i)_{1\leqslant i\leqslant n/2} \leftarrow$ `ConvertA2B`$((A^i)_{n/2+1\leqslant i\leqslant n})$
6: $(y'^i)_{1\leqslant i\leqslant n/2} \leftarrow$ `Expand`$((y^i)_{1\leqslant i\leqslant n/2})$
7: $(z^i)_{1\leqslant i\leqslant n} \leftarrow$ `SecAdd`$((x'^i)_{1\leqslant i\leqslant n}, (y'^i)_{1\leqslant i\leqslant n})$
8: **return** $(z^i)_{1\leqslant i\leqslant n}$

5.3 Higher-Order Masking of Comparison of Polynomials

In this section, we first explain why previous approaches from lattice-based signature schemes are not suitable for application in lattice-based KEMs. Then, we present a more efficient solution and provide a security proof for the side-channel security of our algorithm.

5.3.1 Evaluation of Previous Approaches

In [OSPG18], Oder et al. explain why it is necessary to apply masking to the comparison step. In short, a CCA-attacker is able to make predictions about the input to the comparison and by applying masking to the comparison we prevent the attacker from verifying these predictions. The approach for the comparison of arithmetically masked polynomials from [OSPG18] however only works for first-order masking schemes and is not adaptable to higher-order masking. In [OSPG18] the reported cycle counts also do not explicitly mention the cost of the comparison. The approaches from [BBE+18] and [MGTF19] rely on A2B conversions. They have been proposed for application in lattice-based signature schemes, but an adaptation to lattice-based KEMs is possible and trivial to realize. Due to costly conversions, these comparisons can become a major slow-down factor in implementations of higher-order masked lattice-based schemes [SPOG19]. Therefore, we first analyze the impact that a comparison algorithm based on A2B conversions on the overall performance would have. We give a lower bound for the cost in terms of cycle counts on Cortex-M4 by measuring the necessary cycle counts of an A2B conversion as the most expensive component of the algorithm. Due to the lack of code availability for the implementation described in [BBE+18] and [MGTF19], we developed an A2B implementation ourselves. For our assembly-optimized implementation, we follow the quadratic A2B approach from [SPOG19] that improves upon the cubic A2B algorithm from [BBE+18]. Adjusting the algorithms from [BBE+18] and [MGTF19] to the specific `NewHope` parameters ($n = 1024$ and $q = 12289$), we measure 4,031 cycles for one first-order masked A2B conversion. In one execution of the CCA-secure `NewHope` decapsulation, 2048 A2B conversions are necessary for the comparison of both ciphertext polynomials. These 2048 conversions therefore take 8.3 million cycles and would introduce a significant performance overhead. For reference, in [OSPG18] a first-order-only masked implementation of `NewHope` takes 25.3 million cycles in total. This means that using A2B conversions for a higher-order masked comparison would introduce an unexpected performance overhead. In the following section, we describe a more efficient way that significantly reduces this overhead.

5.3.2 Our Proposal

The inputs to the new comparison algorithm are one unshared polynomial $\tilde{A} \in \mathbb{F}_q[X]$, that is also the input to the `CCA.Decryption` algorithm, and one shared polynomial $\mathbf{A} \in \mathbb{F}_q[X]$ that is the result from the re-encryption in `CPA.Encrypt`. For k coefficients and n shares we have

$$0 \leqslant i < k : A_i = \sum_{j=1}^{n} A_i^j \mod q.$$

The core idea of our efficient algorithm is to perform the comparison directly on a large set of the coefficients of the two polynomials, instead on each of them individually as in previous schemes. The set of coefficients is chosen in such a way that an attacker could pass the comparison step using a malformed ciphertext only with negligible probability. The k coefficients are distributed into x sets of cardinality $l = \frac{k}{x}$ using index sets I_m (for the sake of simplicity, we assume that $x|k$). The sets I_m are constructed in the following way:

$$0 \leqslant m < x : I_m := \{m \cdot \frac{k}{x}, \ldots, (m+1) \cdot \frac{k}{x} - 1\}.$$

After the construction of the sets I_m the following x shared sums B_m^j over subsets of coefficients are calculated (all operations are *mod* q and performed share-wise):

$$0 \leqslant j < n, 0 \leqslant m < x : B_m^j = \sum_{i \in I_m} \left(A_i^j + r_{1,i} \right) r_{2,i}$$

with $r_{1,i}$ and $r_{2,i}$ being uniformly random numbers $\in_r \mathbb{F}_q$. Then each of the shares is summed up:

$$B_m = \sum_{j=0}^{n-1} B_m^j$$

By computing the sum we get

$$\begin{aligned} B_m &= \sum_{j=0}^{n-1} \left(\sum_{i \in I_m} \left((A_i^j + r_{1,i}) r_{2,i} \right) \right) \\ &= \sum_{i \in I_m} \left(\left(\sum_{j=0}^{n-1} A_i^j + n \cdot r_{1,i} \right) r_{2,i} \right) \\ &= \sum_{i \in I_m} \left(\left(A_i + n \cdot r_{1,i} \right) r_{2,i} \right). \end{aligned}$$

Therefore we can now calculate the sums $\tilde{B}_m$ of the coefficients of the same index sets of the unshared polynomial $\tilde{A}$

$$\forall m \in [0, x-1] : \tilde{B}_m = \sum_{i \in I_m} \left(\left(\tilde{A}_i + n \cdot r_{1,i} \right) r_{2,i} \right)$$

and compare it to the sums B_m. This comparison is performed unmasked using any method, e.g., subtracting the B_m from $\tilde{B}_m$ and checking the zero-Bit.

The comparison returns success if $\tilde{B}_m = B_m$ for every $m \in [0, x-1]$ and fails otherwise. This approach needs $2k$ q-sized words randomness and the performance is linear in k and n. Algorithm 5 shows how to compute the masked sums B_m and $\tilde{B}_m$. In the following, our analysis will focus on this algorithm.

Correctness

The correctness of the algorithm is given if a valid ciphertext input successfully passes the comparison and the probability that a malformed ciphertext successfully passes the comparison is negligible. It is trivial to see that the first condition is fulfilled. For the second condition, we analyze the upper bound Y for the probability that an attacker can create a collision, indicated with P_{coll}, i.e., that all m masked sums B_m and $\tilde{B}_m$ are equal even though $A \neq \tilde{A}$.

$$P_{coll} = P(\text{compare}(\mathbf{A}, \tilde{A}) = \text{ true } | \exists i : A_i \neq \tilde{A}_i) \leqslant Y$$

Algorithm 5 `MaskedSum` of m-th set

Input: $\mathbf{A_1}, \ldots, \mathbf{A_l} \in \mathbb{F}_q^n$ such that $\sum_j A_i^j = A_i \mod q$, $\tilde{A}_1, \ldots, \tilde{A}_l \in \mathbb{F}_q$
Output: $B_m, \tilde{B}_m \in \mathbb{F}_q$

1: $(B_m^i)_{1 \leqslant i \leqslant n} \leftarrow \mathbf{0}$
2: $B_m \leftarrow 0$
3: $\tilde{B}_m \leftarrow 0$
4: **for** $i = 1$ to l **do**
5: $\quad R_1 \xleftarrow{\$} \mathbb{F}_q$
6: $\quad R_2 \xleftarrow{\$} \mathbb{F}_q$
7: $\quad$ **for** $j = 1$ to n **do**
8: $\quad\quad B_m^j \leftarrow B_m^j + (A_i^j + R_1) \cdot R_2 \mod q$
9: $\quad$ **end for**
10: $\quad \tilde{B}_m \leftarrow \tilde{B}_m + (\tilde{A}_i + n \cdot R_1) \cdot R_2 \mod q$
11: **end for**
12: **for** $j = 1$ to n **do**
13: $\quad B_m \leftarrow B_m + B_m^j \mod q$
14: **end for**

In order to determine P_{coll}, we point out that

$$P_{coll} \quad = \quad (P_{single-coll})^x$$

where $P_{single-coll}$ is the probability that one pair of masked sums B_m and $\tilde{B}_m$ is equal when $A \neq \tilde{A}$. Assuming random input A, $P_{single-coll}$ is given by:

$$\begin{aligned}
P_{single-coll} &= P(B_m = \tilde{B}_m) - P(\forall i \in I_m : A_i = \tilde{A}_i) \\
&= P\left(\sum_{i=1}^{l} \left(A_i - \tilde{A}_i\right) \cdot r_{2,i} = 0\right) - q^{-l} \\
&= P\left((A_l - \tilde{A}_l) \cdot r_{2,l} = 0\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = 0\right) \\
&+ \sum_{c=1}^{q-1} \left(P((A_l - \tilde{A}_l) \cdot r_{2,l} = c) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = q - c\right)\right) - q^{-l} \\
&= \left(1 - \left(\frac{q-1}{q}\right)^2\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = 0\right) \\
&+ \sum_{c=1}^{q-1} \left(\left(\frac{q-1}{q^2}\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = q - c\right)\right) - q^{-l} \\
&= \frac{1 - q^{-l}}{q} \approx \frac{1}{q}.
\end{aligned}$$

If this assumption holds, the overall collision probability is then

$$P_{coll} \approx \frac{1}{q^x}.$$

As shown in Fig 5.1, the attacker has full control over $\tilde{A}$, but cannot directly influence A. In the following, we want to show that for some input $\tilde{A}$ the resulting A is indistinguishable from a deterministic but random vector and therefore the equation above holds. As Figure 5.1 shows, A is the output of the `CPA.Encryption` and as such also depends on the output of the `CPA.Decryption` that in turn depends on the secret key. With his choice of the $\tilde{A}$, the attacker can single out single coefficients of the secret key and therefore minimize the influence of the secret key. By doing so, the number of possible outputs of the `CPA.Decryption` is q. The output of the `CPA.Decryption` is also the input to the random oracle G, which outputs the seed for the PRNG that is used in the `CPA.Encryption`. The PRNG is then used to generate the noise polynomials in the `CPA.Encryption`. This means that, while the attacker can reduce his uncertainty about the output of the `CPA.Decryption` to a single coefficient, this uncertainty will spread through the PRNG and influence all other coefficients in the output of the `CPA.Encryption`. Therefore A will look random and our equation for P_{coll} holds.

The number of sets is therefore calculated as $x = \lceil -\log_q Y \rceil$. For $q = 12289$ and $Y < 2^{-128}$, we have $x = 10$. In our implementation, we will set $x = 16$ since $16|(k = 1024)$ and there is no difference in performance. The only downside of increasing the number of sets is that the maximum masking order is reduced. But for $x = 16$ sets and $k = 1024$ coefficients, the possible number of shares is still $n \leqslant l = 64$, which should be much more than needed in practice. For $x = 16$, the collision probability is $P_{coll} \approx 2^{-217}$.

Probing Security Proof

In this section we provide the formal proof that our comparison algorithm is t probing secure, with $t \leqslant \min(n - 1, l)$, with probability $1 - q^{-l}$. We proceed by first showing, in the following proposition, that the algorithm `MaskedSum` is t–NI.

Proposition 1. *Algorithm 5 is* t–NI *at any order* $t \leqslant \min(n-1, l)$*, unless* $\forall i \in I_m : A_i = \tilde{A}_i$.

Proof. Let $\mathcal{P} = (\mathcal{I}, \mathcal{O})$ be the set of t adversarial probes on Algorithm 5, with $\mathcal{I}$ the ones on the internals and $\mathcal{O}$ on the output. The elements of $\mathcal{I}$ belong to the following possible groups:

(0) A_i^j

(1) $R_1^{(i)}, R_2^{(i)}$, the values of R_1 and R_2 at the i^{th} iteration of the loop at line 4

(2) $A_i^j + R_1^{(i)}$

(3) $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$

(4) $(A_1^j + R_1^{(1)}) \cdot R_2^{(1)} + \cdots + (A_k^j + R_1^{(k)}) \cdot R_2^{(k)}$ with $k \leqslant l$, for $j = 1, \ldots, n$

(5) $B_m^1 + \cdots + B_m^h$ with $h < n$

(6) $\tilde{A}_i$

(7) $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$

(8) $\sum_{i=1}^{k} (\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $k < l$

with, where not differently stated, $i = 1, \ldots, l$ and $j = 1, \ldots, n$. The set $\mathcal{O}$, instead, is constituted by B_m and $\tilde{B}_m$.

We start by constructing l sets of indexes $I^{(1)}, \ldots, I^{(l)}$ in the following way: for each probe in group (0), (2) or (3) add the index j to the set $I^{(i)}$ and for each probe in group (4) add the index j to each $I^{(i)}$ with $i \leqslant k$. Since for each adversarial observation at most one index is added to the $I^{(1)}, \ldots, I^{(l)}$ and no index is added when there is a probe on the output, then each of the sets has cardinality at most $|\mathcal{P}|$.

We now simulate the set of probes $\mathcal{P}$ by using only the A_i^j with $j \in I^{(i)}$. Since the $\tilde{A}_i$ are already public, group (6) does not need to be simulated.

Step 1 For each element in group (0), by construction $j \in I^{(i)}$ and therefore the element can be simulated as in the real algorithm.

Step 2 Each element in group (1) is simulated uniformly at random, i.e., by picking $R_1^{(i)} \in \mathbb{F}_q$ and $R_2^{(i)} \in \mathbb{F}_q$.

Step 3 For each element in group (2), we distinguish two cases. If $R_1^{(i)}$ was already observed, we take the value simulated in the previous step. Otherwise we pick $R_1^{(i)}$ uniformly at random from $\mathbb{F}_q$. In both cases, since $j \in I^{(i)}$ we can simulate A_i^j as in the real algorithm.

Step 4 For each element in group (3), we distinguish the following cases. If $(A_i^j + R_1^{(i)})$ was probed, we take the already simulated value, otherwise we simulate it according to the previous step. Additionally, we simulate the value $R_2^{(i)}$ with his value from Step 2, if it was previously observed, and by picking it uniformly at random from $\mathbb{F}_q$ otherwise. We finally calculate the product $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$.

Step 5 For each element in group (4), we distinguish the following cases. For any sum $\sum_{i=1}^{h} (A_i^j + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $h < k$ that has already been probed, use the probed value for its simulation. For the rest of the addends, if one of the sums $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$ was probed, we take the simulated value, otherwise we simulate it as in the previous step. We finally calculate the sum of the values as in the real algorithm. Note that by construction, even if all the random bits are probed, the simulation needs at most one share A_i^j of each input.

Step 6 If the probe is in group (5), we point out that, due to the common random bits among the addends, the sum can be rewritten as $\sum_{i=1}^{l} (A_i^1 + \cdots + A_i^h + hR_1^{(i)}) R_2^{(i)}$. Despite the simplification, since $t \leqslant l$, then there exists at least one pair $R_1^{(\hat{i})}$ and $R_2^{(\hat{i})}$ that has not been probed. We pick such $R_1^{(\hat{i})}$ and $R_2^{(\hat{i})}$ uniformly at random from $\mathbb{F}_q$, and therefore there exists at least one of the sums that will be simulated at random, and as a consequence the entire sum is simulated randomly and independently from each A_i.

Step 7 In the case the probe is in group (7), if $R_1^{(i)}$ (resp. $R_2^{(i)}$) has not already been simulated during one of the steps above, pick uniformly at random $R_1^{(i)} \in \mathbb{F}_q$ (resp. $R_2^{(i)} \in \mathbb{F}_q$), otherwise take the value already assigned and in both cases compute the probe as in the algorithm, using the public value $\tilde{A}_i$.

Step 8 In the case the probe is in group (8), for any sum $\sum_{i=1}^{h}(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $h < k$ that has already been probed, use the probed value for its simulation. For the rest of the sums, for each $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ already observed, take the value already simulated. For the remaining addends, simulate them as in Step 7. Finally sum up the $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ as in the real algorithm.

Step 9 For the output $B_m = B_m^1 + \cdots + B_m^n$, since this corresponds to an element in group (5) with $h = n$, the simulations follows the same procedure as Step 6. This time the sum reduces to $\sum_{i=1}^{l}(A_i + nR_1^{(i)})R_2^{(i)}$. We point out again that, since $t \leqslant l$, then there exists at least one pair of elements $R_1^{(i)}$ and $R_2^{(i)}$ that has not been probed.

Step 10 Finally, for the simulation of the output $\tilde{B}_m$, i.e., $\sum_{i=1}^{l}(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ we notice that since $t \leqslant l$, there exists at least one pair $R_1^{(i)}$ and $R_2^{(i)}$ that has not been probed, and therefore there exists at least one of the sums that will be simulated at random, and as a consequence the entire sum is simulated randomly and independently from $\tilde{A}$. If $\forall i \in I_m : A_i = \tilde{A}_i$, this simulation is not consistent because $\tilde{B}_m$ must be equal to B_m in this case and it therefore depends on the A_i.

From the simulation above and Definition 1 we can conclude that Algorithm 5 is t–NI, unless $\forall i \in I_m : A_i = \tilde{A}_i$. □

In the following we show, that the probabilistic nature of our security proof is of no consequence and the comparison is secure when using practical parameters.
Note that Proposition 1 implies that the outputs B_m and $\tilde{B}_m$ of Alg. 5 and, by extension, the result of the comparison of these values can be simulated without knowledge of any coefficient A_i unless $\forall i \in I_m : A_i = \tilde{A}_i$. In this case the proof fails which results in possible leakage of the A_i. This type of collision can happen in one of the following two cases:

(1) $\forall m, \forall i \in I_m : A_i = \tilde{A}_i$, i.e., all coefficients are equal.

(2) $\exists m, \forall i \in I_m : A_i = \tilde{A}_i$ and $\exists m, \exists i \in I_m : A_i \neq \tilde{A}_i$, i.e, only the coefficients used in some sums are equal.

In case (1) the A_i are not sensitive as they are already known by the attacker.
As noted in Sect. 5.3.2 any change in a coefficient of $\tilde{A}$ is propagated to all coefficients of A through the random oracle G. Therefore, if the output $coin''$ of `G` is collision free, the probability of case (2) is $P_{B-coll} = q^{-l}$. When using practical values for q and l, P_{B-coll} is always smaller than the collision probability of `G`[1] and the algorithm is secure against t-order attackers. For example, with our parameter choice for `NewHope` ($q = 12289$, $l = 64$) $P_{B-coll} < 2^{-869}$.

[1]More precisely, the relevant metric is second-preimage resistance. In the case of `NewHope` $coin''$ is $\in \{0, ..., 255\}^{32}$.

5.3.3 Application to NewHope and Other Schemes

While our masking scheme is relevant for a large variety of schemes, we specifically pick NewHope [ADPS16] as case study to be consistent with previous work [RRVV15, RdCR+16, RRdC+16, OSPG18, SPOG19]. The relevant parameters of the scheme are the lattice dimension (i.e., the number of coefficients in the polynomials) of $k = 1024$ and the modulus $q = 12289$. We expect similar results when our algorithm is applied to other schemes. Generally, higher-order masked lattice-based KEM implementations that rely on the Fujisaki-Okamoto transform to achieve CCA security can benefit from our comparison approach as long as the parameters are compatible to the requirements described in Sect. 5.3.2. For example, in Kyber [BDK+18] the lattice dimension is only $k = 256$, but depending on the parameters set there are more (up to four) polynomials in the ciphertext. As the CCA security of Kyber depends on the Fujisaki-Okamoto transform and as shown in Alg. 9 in the Kyber specification [SAB+19], the comparison is the same as in NewHope. Consequentially, the input to the comparison in line 6 also depends on the output of a random oracle G, therefore our proof in Sect. 5.3.2 holds. The modulus of LAC [LLZ+18] is only $q = 251$. This modulus would lead to reduced memory requirements as each coefficient can be stored in a single byte.

5.4 Microcontroller Implementation

In this section, we discuss our microcontroller implementation in detail and the methodology for our side-channel measurements.

5.4.1 Microcontroller Implementation

Our evaluation platform is the STM32F4-DISCOVERY board that is based on the STM32F407VGT6 ARM Cortex-M4F microcontroller. NIST recommends to use the Cortex-M4F as target platform for microcontroller evaluations of post-quantum standardization candidates [Moo19]. Furthermore, the concrete processor and board we used in our performance and side-channel evaluation was suggested as a reference platform for PQC algorithms in [KRSS19]. It runs with a clock frequency of up to 168 MHz. The board offers 192 kB of RAM as well as 1 MB of flash memory. Furthermore, it features a true random number generator (TRNG) based on analog circuitry and a floating-point unit (FPU). The Cortex-M4F has 13 general purpose registers and $(R_0 - R_{12})$, one register reserved for the stack pointer, a link register, one register reserved for the program counter, and special-purpose program status registers. When mixing C with assembly it is important to note that the calling convention requires parameters to be in $R_0 - R_3$ and the result to be in $R_0 - R_1$. The link register can be used as general purpose register as well, if the assembly function does not call any other function and its original value is restored before leaving the function.

To prevent timing leakages, implementations of cryptographic schemes are usually expected to be protected against timing attacks, this is usually referred to as *constant-time* property of implementations. However, we need to distinguish between two different notions of *constant-time*. In the following, we will use the expression *constant-time* in case the execution time of an implementation is actually constant. Furthermore, we will refer to an implementation as *timing-independent* if the implementation has a non-constant execution time but is still

protected against timing side channels because the execution is independent from the input data.

The implementation of our comparison algorithm requires the generation of random numbers in $[0, q-1]$, where q is an arbitrary integer and in many cases (like `NewHope`) a prime. To ensure a uniform distribution of these numbers, we apply the rejection sampling method from [SPOG19] that takes as input random bit vectors from the on-board TRNG and only accepts the input if the value is in $[0, q-1]$ and rejects otherwise. As this method works with rejections, it is *timing-independent*, but not *constant-time*. We therefore implemented two variants of the algorithm - one *constant-time* implementation that is suitable for side-channel evaluation and one performance-optimized variant for practical use. The main difference between these two implementations is that the side-channel measurement-friendly variant first fills a buffer with random values in $[0, q-1]$. The implementation of the actual comparison then just accesses this buffer to load the necessary random numbers. By doing so, side-channel measurements can be triggered after the (*non-constant-time*) generation of random numbers is completed avoiding the necessity of trace alignment in the side-channel experiments. However, as the on-board TRNG needs to sample sufficient thermal noise in the background, requesting random numbers from the TRNG in rapid succession is quite slow as the program will be halted until the TRNG is ready. In our second performance-optimized approach, we therefore spread out the TRNG calls throughout the algorithm to minimize the TRNG waiting time.

```
MOV TMP,#0x3001
ADD INPUT_A, INPUT_B
SUB INPUT_A, TMP
SXTB TMP2, INPUT_A, ROR #24
AND TMP, TMP2
ADD INPUT_A, TMP
```

Listing 5.1: Combined modular addition and subtraction in assembly.

For the modular reduction, we use the constant-time Barrett reduction from [OSPG18] that uses the floating point unit of the Cortex-M4F. We furthermore use a special method to combine the addition and modular reduction of two values *mod q* as shown in Listing 5.1. The idea is to perform a conditional subtraction of the modulus in constant-time. We first load the modulus into a temporary registers tmp_1. Then we add the two inputs and subtract the modulus from the sum. With the help of the `SXTB` instruction and the internal barrel shifter of the Cortex-M4F we create a bit mask that is either $1...1_2$ in case the result of the subtraction was negative or $0...0_2$ if the result was positive and store the bit mask in tmp_2. We then compute the `AND` of tmp_1 and tmp_2. The register tmp_1 now contains either the modulus if the result of the subtraction was negative or $0...0_2$ if the result was positive. Finally, tmp_1 is added to the result of the subtraction to receive the output. This approach takes only six cycles for combined addition and modular reduction and needs two temporary registers for intermediate results.

We try to minimize the load and store memory accesses by efficiently using the available registers of the Cortex-M4F. While doing so, it is important to keep in mind that we first sum up all coefficients within a set of a share and then sum up the sums of each share. It might be tempting to switch the order of these summation because it would save a big number of loads

and stores of the random r values. However, this would also introduce a side-channel leakage as the secret shares would be combined without sufficient randomness. Apart from the r values, no value is loaded twice and no store instruction is used. We therefore argue that our memory access scheme is optimal.

5.4.2 Side-channel Measurements

In order to practically evaluate the resistance of the proposed comparison algorithm against side-channel attacks, we performed a Test Vector Leakage Assessment [GJJR11] of the *constant-time* implementation described in Section 5.4.1. With the goal of reducing the computational complexity of the evaluation, we set the number of coefficients in the measured implementation to four and only considered $k = 1$ sets. Note, that this does not weaken the evaluation results due to the construction of our algorithm. By using the non-specific fixed-versus-random t-test, the analysis can show possible leakage points independent of specific sensitive variables. In this evaluation methodology, the target device is supplied with either a fixed or random input in a random order. During the computation of the target, the side channel (e.g. power consumption or EM emanation) is measured on which the test metric is applied to decide if the consumption is distinguishable depending on the input. For the first-order univariate case, the test statistic to evaluate if the mean of a sample point of the two sets of traces F and R is different can be computed as

$$t_{F,R} = \frac{\bar{F} - \bar{R}}{s_n}$$

with

$$s_n = \sqrt{\frac{s_F^2}{n_F} + \frac{s_R^2}{n_R}}$$

where n_X, $\bar{X}$, and s_X^2 are the number of collected samples, the estimated means and the estimated variance of the respective point. The magnitude of this test-statistic can then be compared to a threshold which is required to be reached to confirm an input-dependent mean of the analyzed sample point. For the evaluation of complete power- or EM-traces, the statistic can be computed point-wise. As pointed out in [DZD+17], this simultaneous application of multiple tests can artificially skew the outcome towards a positive result. In order to obtain a result with a given confidence, the detection threshold must therefore be adjusted depending on the number of samples in a trace. We use the Šidák Correction as suggested in [BPG18] to calculate the threshold for a confidence level of α given a trace length L and n measurements:

$$t_{th} = Q_t(1 - \sqrt[L]{1-\alpha}, v),$$

where $Q_t(\cdot)$ is the quantile-function of the t-distribution and $v \approx n/2$ the degree of freedom. Therefore a threshold for multivariate leakage (with $L{*}L$ points) is always higher when compared to the univariate case. As our experiment targets a software implementation where different shares of a sensitive variable are manipulated at different points in time, a multivariate analysis is necessary for achieving a meaningful evaluation of higher-order leakage. We restricted the experiments to first- and second-order analysis of the two- and three-share variants of the comparison algorithm, as higher-order multivariate leakage assessment is computationally prohibitive. This is because the effort for the required sample-point combination is proportional

to $n \cdot L^E$ where E is the evaluation order. The evaluation of second-order multivariate leakage was performed following [SM15] by combining all pairs of points in a trace using the optimal centered product according to [SVO+10].

We sent a fixed or random challenge to the target microcontroller in a random order. The power consumption was measured while the microcontroller was evaluating the comparison algorithm on either a fixed or a random input. The device generated a trigger pulse for optimal trace alignment and performed the comparison algorithm on the provided input and a fixed set of coefficients stored at compile-time. When generating the random input, we rejected values that were equal to the reference coefficients to avoid producing high false-positive (non-exploitable) t-test results in the evaluation.

The measurements were conducted on the same ARM Cortex-M4F board that was used for the performance evaluation.

In order to show the side-channel resistance of our implementation in a worst-case scenario, we increased the signal-to-noise ratio as much as possible by modifying the PCB as well as adapting the firmware accordingly. For data transfer between a host computer and the target board, we made use of a UART-core of the microcontroller which eliminates noise introduced by the on-board USB interface. In addition to disabling data and instruction caches available in the STM32F407VG, the SysTick-timer and all interrupt source in the controller were disabled to ensure *constant-time* behavior of the measured code. For further noise reduction in the measurement system, all non-essential clock paths in the target controller were disabled. In an effort to reduce other noise sources as much as possible, all non-essential peripheral devices on the board, such as the MEMS accelerometer and the audio DAC, were de-soldered.

As power measurements appeared to not contain a sufficient signal-to-noise ratio for a successful evaluation on our microcontroller board, we collected EM traces with a near-field probe and amplified them before feeding them into an oscilloscope. The EM traces were acquired with a sample rate of 156.25 MS/s at 8 bit resolution using a 50 MHz-bandwidth H-field probe. We kept the position of the probe relative to the microcontroller fixed between measurements in order to produce comparable results.

The findings of the practical SCA-evaluation are provided in Sect. 5.5.3.

5.5 Results

In this section, we present the results of our practical evaluations. This includes performances evaluations as well as side-channel evaluations.

5.5.1 Performance Evaluation

We evaluated our work by using the *OpenSTM32 System Workbench* (version 2.6), which is based on the development environment `Eclipse` and has specifically been designed to support the development for ARM-based STM32 boards. The IDE uses the GNU ARM Embedded Toolchain (version 7.2) and we set the optimization level to `-O3`. Determining the performance of our implementation was done by using the cycle count register `DWT_CYCCNT` of the *Data Watchpoint and Trace* unit that the Cortex-M4F offers. We set the clock frequency of the microcontroller to 24 MHz to avoid any wait cycles at the memory. All cycles counts were obtained by measuring our performance-optimized implementation. With the publication of

this work, we will make our implementation publicly available to allow independent verification of our results.

In Table 5.1, we show the cycle counts for the comparison algorithm of one polynomial with $k = 1024$ coefficients. We compare our approach with the cost of 1024 A2B conversions as explained in Section 5.3.1. Both implementations benefit from assembly optimization. The direct comparison of both approaches shows that for two shares already, our approach is at least 14 times faster than an A2B-based approach. It is also obvious from the table that our algorithm has a better asymptotic complexity as it is only linear in the number of shares while A2B conversions are at best quadratic. Therefore the speed-up factor gets even bigger when the number of shares is increased which makes our algorithm two orders of magnitude faster for five shares already. Extrapolating these numbers we expect the cycle counts for higher orders to be around $165,000 + 41,000 \cdot n$, where n is the number of shares.

Table 5.1: Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for $k = 1024$ including randomness generation. All results are averaged over 100 runs.

Shares	2	3	4	5
Comparison in [OSPG18]	480,227	-	-	-
`NewHope` CCA-DEC [OSPG18]	25,334,493	-	-	-
1024 A2B conversions	4,127,744	11,875,328	21,027,840	35,353,600
Our comparison algorithm	250,991	284,989	329,053	373,860
Speed-up factor	**x16**	**x42**	**x64**	**x95**

At our measurement frequency of 24 MHz, 250,991 cycles are executed in 10 milliseconds. However, the maximum clock frequency of the microcontroller is 168 MHz. For reference, we also measured our implementation at 168 MHz, to get a realistic impression of the time needed to execute the algorithm. We observe only a minor increase in the number of cycles at 168 MHz, namely to 258,695 cycles. Since the algorithm does not load any constants values from flash memory, we assume that this difference is mainly caused by the variable timing behavior of our implementation (see Section 5.4.1 for a discussion about why our implementation is secure against timing attacks). We therefore conclude that the cycle counts obtained at 24 MHz also give a realistic impression of the performance of the implementation at 168 MHz. Therefore, at 168 MHz, the masked comparison of one polynomial takes 1.5 ms for two shares, 1.7 ms for three shares, 2.0 ms for four shares, and 2.2 ms for five shares. This makes our approach a low-overhead component for the construction of higher-order masking schemes lattice-based cryptography.

For polynomials with $k = 1024$, the maximum masking order that our implementation supports is 64 as we decided to implement 16 subsets of coefficients. We expect this to be sufficient for practical uses in the foreseeable future. One downside of our approach is the relatively high dynamic memory consumption as we need to store $2k$ random values *mod q*. This is equal to the memory consumption of two polynomials. However, embedded into a lattice-based KEM, we do not expect our algorithm to increase the dynamic memory consumption at all as the peak memory usage is expected to be in the `CPA.Encryption` or `CPA.Decryption` and the compari-

son algorithm is executed after these two components. The static Flash memory consumption of our algorithm is also very low as it needs only about 200 lines of assembly code.

To our knowledge, currently there exists no higher-order CCA-secure implementation of any lattice-based KEM. However, we also implemented the approach from [OSPG18], even though this algorithm is only first-order secure and cannot be extended to higher orders. Table 5.1 includes the cycle count for the complete CCA2-secure decryption as well as for only the comparison step. The board used to measure these performance values was identical to the one we used in our evaluation. The idea from [OSPG18] is to subtract $\tilde{A}$ from one share of A, individually hash the result of this subtraction and the other share and compare the outputs of the hash calls. Our implementation achieves similar performance even though our source code can generically support higher orders too and the implementation from [OSPG18] is optimized for the first-order case.

Table 5.2: Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for different parameter sets including randomness generation. All results are averaged over 100 runs.

Shares		2	3	4	5
KYBER-768	A2B conversions	3,095,040	8,906,496	15,770,880	26,515,968
	Our algorithm	185,338	216,945	248,455	279,973
	Speed-up factor	**x17**	**x41**	**x63**	**x95**
LAC-192	A2B conversions	2,267,136	6,360,064	11,131,904	18,551,808
	Our algorithm	230,432	272,489	314,558	356,621
	Speed-up factor	**x10**	**x23**	**x35**	**x52**

In Table 5.2 we also exemplarily evaluated the performance of our implementation for the parameter sets of KYBER-768 and LAC-192 to show the impact of the choice of n on the performance of the comparison. KYBER-768 uses $k = 768$ coefficients and a modulus $q = 3329$. In this case, we observe very similar speed-up factors in comparison with an A2B approach. This is expected since both approaches are linear in k. Therefore the cycle counts for $k = 768$ are also roughly equal to three quarters of the cycle counts for $k = 1024$. In this scenario the cycle count for higher orders can be expected to be around $122,000 + 32,000 \cdot n$ with n shares.
While LAC-192 operates on the same number of $k = 1024$ coefficients as NewHope the modulus is $q = 251$ in this case. This results in lower randomness requirements of our algorithm and therefore slightly increased performance when compared to NewHope. As the A2B-based approach benefits even more from this, the speed-up gained by our approach is lower. For LAC-192, the cycle count for higher orders can be expected to be around $145,000 + 42,000 \cdot n$ with n shares.

As stated before, complete higher-order masked lattice-based KEMs are currently not available for comparison. However, we can roughly estimate the impact of our comparison algorithm at higher orders. For obvious reasons, the cycle count of any masked implementation must be at least linear in the number of shares n.[2] As our algorithm has linear complexity in n the relative overhead of the algorithm does not (asymptotically) grow with increasing n.

[2]Otherwise it could just use less shares.

5.5.2 Randomness Consumption

In this section, we analyze how much randomness our implementation needs for the masking scheme. For our specific choice of $q = 12289$, the rejection-based sampling of uniform random numbers *mod* q has an acceptance rate of 75% since the acceptance rate a_r is calculated as $a_r = \frac{12289}{2^{14}}$. For efficiency reasons, we use chunks of 16 bits of randomness for one sampling attempt of which two bits are simply ignored. On our evaluation platform the randomness can be efficiently generated by the on-board TRNG. On platforms with lower randomness generation capabilities the randomness requirements can be reduced to 87.5% of the values below by using 14- instead of 16-bit chunks. This approach requires more memory to store the randomness and more processor cycles to extract it. As we need $2k$ random numbers *mod* q, the expected number of required random bits for our approach is

$$r_{bits} = 2k\frac{16 \text{ bits}}{a_r} = 2 \cdot 1024 \cdot \frac{16 \text{ bits}}{0.75} = 43,688 \text{ bits}$$

This calculation is valid for any number of shares $n < 64$. This theoretical amount of randomness is confirmed by our practical experiments. We compare the randomness consumption of our approach to the A2B-based approach in Table 5.3 as measured by our implementation. The first-order only approach from [OSPG18] does not need any additional randomness. While our algorithm needs less randomness than the A2B-based approach even for low masking orders this advantage increases for higher orders because the randomness requirement is independent of the masking order.

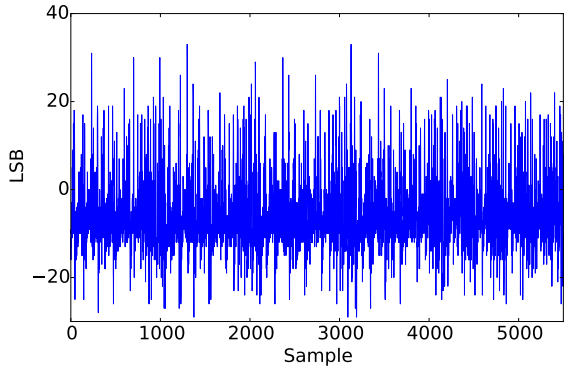
Table 5.3: Random bit consumption for our ARM implementations of the masked comparison for different parameter sets. All results are averaged over 100 runs.

Shares		2	3	4	5
`NewHope`	A2B conversions	655,360	2,392,064	4,653,056	8,519,680
	Our algorithm	43,648	43,680	43,584	43,712
	Improvement factor	**x15**	**x55**	**x107**	**x195**
`KYBER-768`	A2B conversions	491,520	1,794,048	3,489,792	6,389,760
	Our algorithm	30,240	30,208	30,176	30,272
	Improvement factor	**x16**	**x59**	**x116**	**x211**
`LAC-192`	A2B conversions	393,216	1,343,488	2,555,904	4,587,520
	Our algorithm	16,704	16,711	16,706	16,705
	Improvement factor	**x24**	**x80**	**x153**	**x275**

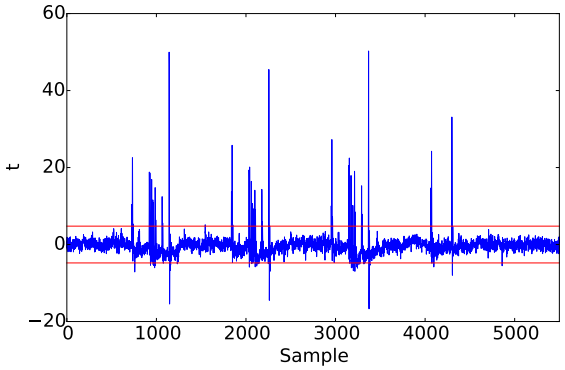
5.5.3 Leakage Evaluation

This section details the results of our experimental side-channel evaluation. We used a significance level of $\alpha = 0.01$ for all assessments in this section. For reference, Figure 5.2a shows an example trace of the measured EM-emanation. When the masking countermeasure is deactivated by setting the masks to zero, large first-order leakage can be observed as shown in

Figure 5.2b even using only 5000 traces. This behavior is expected as the algorithm operates on unmasked values in this case.



(a) Example trace of two-share version with four coefficients.



(b) First-order leakage for two-share version with four coefficients (masks disabled, 5 k measurements, $t_{th} = 4.77$).

Figure 5.2: Sample trace and reference measurement.

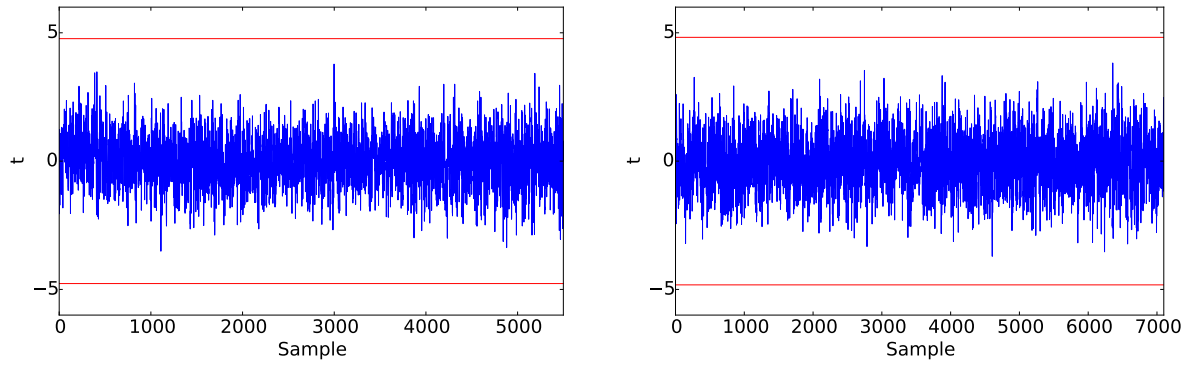
When random masks are used, an evaluation using 1 million traces does not show detectable first-order leakage (Fig. 5.3a). The two-dimensional plot resulting from the bivariate TVLA is shown in Figure 5.4. Each pixel is colored according to the absolute t-value present in the respective combinations of points in time. In order to ensure readability of Figures 5.4 and 5.5 we binned the sample points into a 100 times 100 pixel-grid and plotted the maximum t-value of each bin. The multivariate analysis of the second-order leakage allows to clearly identify points at which different shares of coefficients are handled, as the 2-share implementation only protects against first-order attacks.

Figure 5.3b shows the result of a first-order evaluation on traces collected from the three-share implementation with four coefficients and activated masking after 1 million measurements. As expected, no first-order leakage can be detected. The results of the second-order multivariate t-test is shown in Fig. 5.5. The leakage detection threshold of $t_{th} = 6.36$ is not reached at any point in time.

In summary, we were not able to detect first-order leakage in the two-share *constant-time* implementation or second-order multivariate leakage in the three-share implementation even using 1 million measurements.

5.6 Conclusions and Future Work

In this work, we identify the comparison step of the Fujisaki-Okamoto transform as, a so far overlooked, bottleneck in higher-order masking of lattice-based cryptography. We present a novel higher-order masking scheme for the comparison, that outperforms the naive approach by at least one order of magnitude and it is applicable to constructions with prime modulus. The naive approach based on A2B conversions has a complexity of $O(n^2k)$, while the asymptotic complexity of our algorithm is only $O(nk)$, i.e., it is linear in the number of shares and in the number of coefficients of the polynomial. Furthermore, the probability for an attacker



(a) First-order leakage for two-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 4.77$).

(b) First-order leakage for three-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 4.82$).

Figure 5.3: First-order SCA analysis of 2- and 3-share implementation.

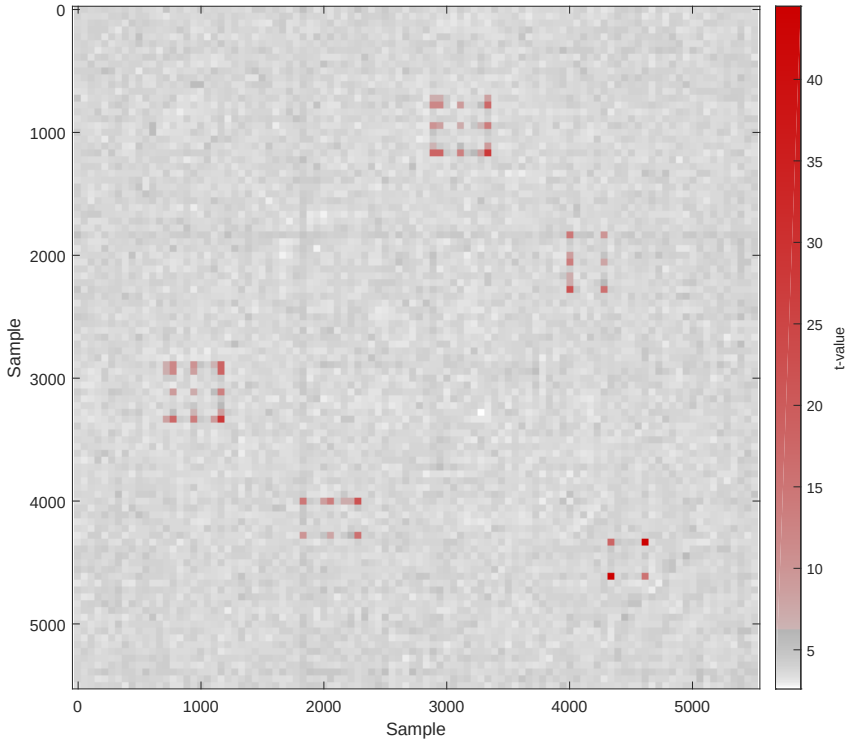


Figure 5.4: Second-order leakage for two-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.28$). Points with t-values above the threshold are highlighted red.

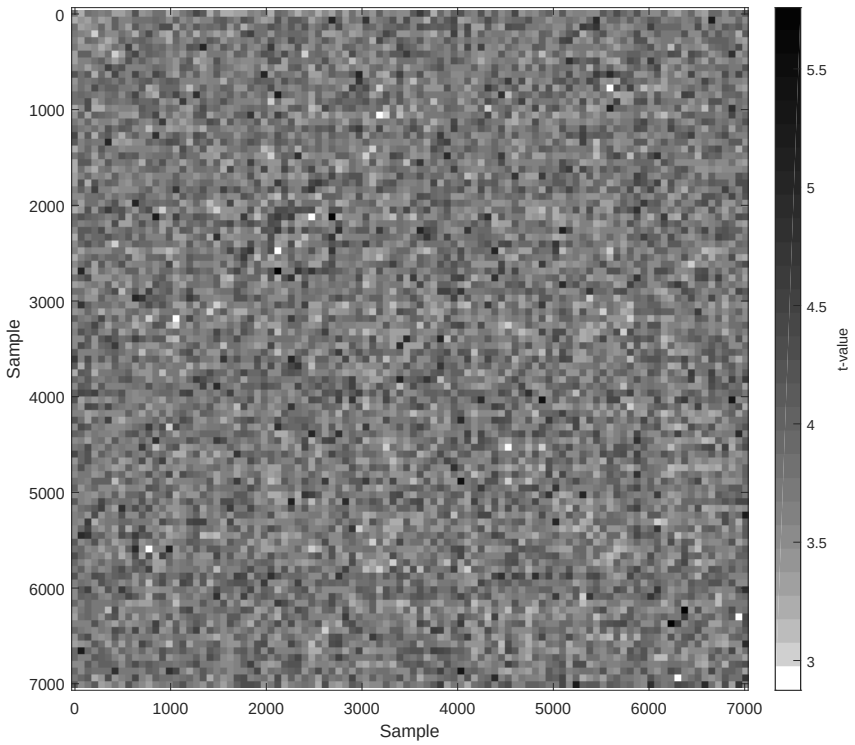


Figure 5.5: Second-order leakage for three-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.36$). Points with t-values above the threshold are highlighted red (none present).

to forge an invalid ciphertext that is still accepted by our comparison is negligible (2^{-217}). We give a theoretical proof of the side-channel security of our algorithm and confirm with practical measurements that our highly efficient microcontroller implementation does not show side-channel leakage, even for significantly more power traces than in previous work on masking for lattice-based cryptography. In the ongoing NIST post-quantum standardization, our work is an important step towards understanding the overhead cost of side-channel countermeasures applied to the NIST candidates.

After the publication of our work at CHES 2020 [BPO+20] the authors of [BDH+21] discovered two possible attacks against the masking scheme proposed in this chapter. The first attack relates to the fact that, while the final result of the comparison is not sensitive, the comparison over a subset of coefficients is, if a slightly modified ciphertext is generated by an attacker. A similar vulnerability was also found to apply to [OSPG18]. The second attack exploits a similar weakness by which the assumed collision probability of q^{-x} can be increased to $\frac{1}{q}$. This effectively breaks the CCA security of the scheme without requiring side-channel information. In order to prevent these vulnerabilities the authors of [BDH+21] propose to generate a masked sum over all coefficients instead of a subset of coefficients. This sum generation is then to be repeated x times, such that the collision probability q^{-x} is acceptably low.

As future work, we advise extending the comparison algorithm to work for power of two moduli, which at the moment are not considered in our scheme. Furthermore it would be interesting to see how our masking countermeasure can be combined with countermeasures against other attacks, like fault injection attacks, since it was already analyzed in [OSPG18] that the comparison step could be a primary target for fault attacks.

Chapter 6

Masking Addition with Boolean Shares

Modular addition is an important component of many cryptographic algorithms such as ARX-ciphers and lattice-based post-quantum secure schemes. In order to protect devices that execute these algorithms against side-channel attacks, countermeasures such as masking must be applied. However, if an implementation needs to be secured against multivariate attacks, univariately secure masking schemes do not suffice.

In this chapter, we focus on hardware architectures for higher-order masked addition circuits. We present and discuss three adder designs that are protected with a provably secure masking scheme. Concretely, we discuss Kogge-Stone, Sklansky and Brent-Kung adders regarding their suitability for high-order masking and their performance in this setting. All architectures are fully pipelined and achieve a throughput of one addition per cycle. In order to achieve multivariate security at arbitrary orders, we use HPC2 Gadgets that satisfy the PINI security notion. Additionally, we apply a first-order secure threshold implementation scheme to the adder variants and compare their performance in the univariate case. The work described in this chapter was published at MDPI Applied Sciences [BG22].

Contents of this Chapter

6.1 Introduction **83**
6.2 Preliminaries **84**
6.3 Boolean Masking for Addition Circuits **85**
6.4 Implementation Results and Discussion **91**
6.5 Conclusion **92**

6.1 Introduction

Modular addition in the ring $\mathbb{Z}_{2^n}$ is a core part of several cryptographic schemes. For example, in ARX ciphers they perform similar function to S-boxes in classical block ciphers, being the only non-linear part of the algorithm. ARX constructions, such as SPECK or SALSA20, combine this arithmetic function with the boolean exclusive or operation (XOR) and a bit-wise rotation to produce a secure algorithm. When hardware implementations of such algorithms can potentially be targets of side-channel attacks, masking can be used as an effective countermeasure. However, while the separate protection of the arithmetic and boolean parts of these algorithms is possible

using an arithmetic and a boolean masking scheme, the conversion between these representations poses a significant problems, especially in hardware.

6.1.1 Contribution

In this work we study how addition operations can be protected against side-channel attacks in the hardware context using boolean masking. We propose three different designs for parallel-prefix addition circuits that can generically be masked at arbitrary order using gadgets that follow the PINI security notion. To this end, we study their suitability for masking and compare them regarding their area and randomness requirements as well as their latency. In order to achieve higher-order multivariate security, we employ the HPC2-gadgets that were proposed in [CGLS20]. We used the source code for the gadgets from the library provided by the authors [Cas] in our implementations. Concretely, the proposed adder structures are:

(1) A Kogge-Stone adder with a latency of $\log n$-cycles.

(2) A Sklansky adder with the same latency but reduced area and randomness requirements.

(3) A Brent-Kung adder which trades of higher latency for an even lower area requirement and requires less randomness.

After performing a detailed analysis of their structure, we implement these adder types as 32-bit variants on an FPGA. All implemented variants are fully pipelined and therefore achieve a throughput of one addition per cycle. To our knowledge, the proposed adders are the first arbitrary-order masked hardware designs that are secure against multivariate attacks.

Additionally, we discuss the application of the TI masking scheme to the Sklansky and Brent-Kung structures, where we use the same sharing that the authors of [SMG15a] applied to ripple-carry and Kogge-Stone adders. Only the first order variant is considered, as higher-order TI can not generically provide protection against multivariate attacks [Rep15]. In this case, randomness and area can be saved in comparison to the variants that are secure at arbitrary order.

6.2 Preliminaries

6.2.1 Notation

The binary operations `and`, `or` and `xor` are denoted by the symbols $\wedge$, $\vee$ and $\oplus$, while the $+$ sign is used for the addition over integers or rings. All logarithms are in base two. The least significant and most significant bits of the n-bit variable a are a_0 and a_{n-1} , respectively. Single subscripts indicate the index of multi-bit variable (e.g. a_i) and consecutive groups of bits between index i and j are indicated as $a_{\{i,j\}}$. In order to simplify equations, indices are treated as zero if they become negative. Superscripts note the respective share of a shared variable. Logarithms are always in base two.

6.2.2 Parallel Prefix Adders

In order to compute the sum s of two n-bit inputs over $\mathbb{Z}_n$, an addition circuit needs to compute a sum bit s_i for every pair of input bits (a_i, b_i) using carry bits c_i as:

Table 6.1: Number of elementary XOR and AND operations per basic function.

Function	$g(i)$	$p(i)$	$g(\{i,j\})$	$p(\{i,j\})$	$s(i)$
AND	1	0	1	1	0
XOR	0	1	1	0	1

$$s_i = a_i \oplus b_i \oplus c_i \quad \text{with}$$

$$c_i = c_{i-1} \wedge (a_{i-1} \oplus b_{i-1}) \vee (a_{i-1} \wedge b_{i-1}) \quad \forall i > 0 \text{ , else } 0$$

A direct realization of these equations leads to a carry-ripple adder with a circuit depth of n due to the dependency of c_i from c_{i-1}. When masking this architecture in hardware where glitches occur, the required registers in each stage lead to a latency of n cycles.

Parallel-prefix adders can reduce this latency by restating the computation of the carry bits using (group-) generate and propagate terms and computing them in parallel. Intuitively, the generate term $p_{\{i,j\}}$ determines if the groups of input bits $a_{\{i,j\}}$ and $b_{\{i,j\}}$ will generate a carry output c_{j+1}, independently of the carry input c_i. The propagate term determines if an input carry c_i will affect the output carry c_{j+1}.

For single-bit inputs (a_i, b_i), the generate and propagate terms are computed as $g_i = a_i \wedge b_i$ and $p_i = a_i \oplus b_i$, respectively. We call this initial step, which is the same for all parallel-prefix adders, the preprocessing step. Given $i > k \geqslant j$, the group-generate term is calculated as

$$g_{\{i,j\}} = g_{\{k,j\}} \oplus (p_{\{k,j\}} \wedge g_{\{i,k+1\}}) \tag{6.1}$$

and the group-propagate term as

$$p_{\{i,j\}} = p_{\{i,k+1\}} \wedge p_{\{k,j\}} \, . \tag{6.2}$$

The final sum bits can then be computed as $s_i = g_{i-1:0} \oplus p_i$, where $g_{-1:0} = 0$. The costs in XOR and AND operations for implementing each of the functions mentioned above are itemized in Table 6.1.

In the following we use $pg_{\{i,j\}}$ (or PG-term where the indices are not relevant) as a short form for the tuple $(p_{\{i,j\}}, g_{\{i,j\}})$. We define the function block that computes these terms by combining existing terms as $PG(\{i,j\},\{k,l\}) = (p(\{i,l\}), g(\{i,l\}))$ and call them PG blocks.

The different variants of parallel-prefix adders, such as the ones discussed here, all apply this same principle but differ in the way PG-terms of larger groups of bits are constructed from smaller ones.

6.3 Boolean Masking for Addition Circuits

In this section, we discuss the application of three different architectures for parallel addition circuits that can be effectively used for higher-order masked applications. In many cryptographic schemes, e.g., in ARX algorithms, the addition operation is performed in the ring $\mathbb{Z}_n$, where $n = 2^m, m \in \mathbb{Z}$. Therefore, we assume the width of the addition circuits to be 2^m and do not

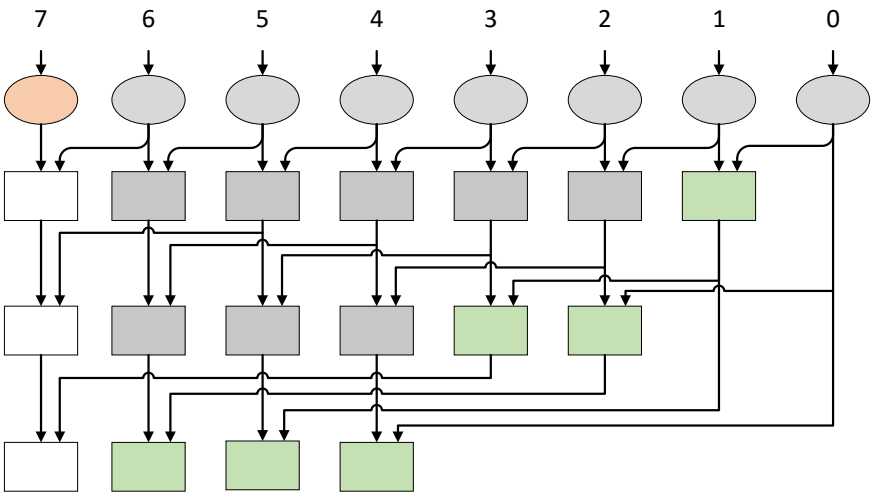


Figure 6.1: 8-bit Kogge-Stone Adder.

consider input and output carry bits. However, the proposed designs can be adapted to other bit widths and to the handling of carries, if required by an application.

Throughout this sections we use diagrams to illustrate the structures of the algorithms under discussion. In these diagrams, the preprocessing blocks are represented by oval shapes while all other blocks are rectangular. A block is shaded gray if both, the generate and propagate terms, are computed by it. If a block only needs to calculate the generate or propagate term, it is shaded green or orange, respectively. Blocks that are only needed if an output carry is required are not filled. All described adder designs require a final stage for the calculation of the sum bits, which is not shown in the diagrams.

6.3.1 Kogge-Stone Adder (KSA)

KSAs [KS73] are parallel-prefix adders that are similar to greedy algorithms in the sense that the maximal number of PG-terms are combined in each stage. A n-bit KSA requires $\log n$ stages to compute all g({i, 0}) terms and one additional stage to compute the final sum bits. In order to better illustrate the architecture, an 8-bit KSA is depicted in Fig. 6.1. When excluding the preprocessing step and the final sum calculation, it requires 3 stages and 14 PG-blocks.

In the first stage after the preprocessing step, every PG-term is combined with its neighbor, i.e., $PG(\{i,i\}\ ,\ \{i-1,i-1\})\ \forall 0 < i < n-1$ is computed. Therefore, $n-1$ PG function blocks need to be instantiated in that stage. In the second stage, $PG(\{i,i-1\}\ ,\ \{i-2,i-4\})\ \forall 1 < i < n-1$ is computed, requiring $n-2$ PG function blocks. Note that $p(\{1,0\})$ is not needed in this computation and the hardware for its generation can therefore be omitted in the first stage. In general, stage k computes $n-2^k$ PG-terms as[1]:

$$PG(\{i, i-2^k+1\}\ ,\ \{i-2^k, i-2^{k+1}+1\})\ \forall k < i < n-1.$$

[1]When numbering the stages, the stage number k of the first stage after preprocessing is zero.

Table 6.2: Number of required basic functions for different parallel-prefix adders.

	$g(i)$	$p(i)$	$g(\{i,j\})$	$p(\{i,j\})$	$s(i)$
Kogge-Stone	$n-1$	n	$(n-1)\cdot\log n - n + 1$	$(n-1)\cdot\log n - 2\cdot n + 3$	n
Sklansky	$n-1$	n	$(n/2-1)\cdot\log n$	$(n/2-1)\cdot\log n - n + 2$	n
Brent-Kung	$n-1$	n	$2\cdot n - 2\cdot\log n - 2$	$n - 2\cdot\log n$	n

Table 6.3: Number of elementary XOR and AND operations for different parallel-prefix adders including the preprocessing stage and final computation of the sum bits.

	Kogge-Stone	Sklansky	Brent-Kung
AND	$2\cdot(n-1)\cdot\log n - 2\cdot n + 3$	$(n-2)\cdot\log n - n + 1$	$3\cdot n - 4\cdot\log n - 1$
XOR	$(n-1)\cdot\log n - 3\cdot n$	$(n/2-1)\cdot\log n + 2\cdot n$	$4\cdot n - 2\cdot\log n - 2$

As we do not consider the output carry generation, the calculation of $pg(\{n-1,i\})\ \forall i \neq n-1$ can be skipped, saving $\log n$ PG blocks. This reasoning does not only apply to Kogge-Stone adders but also to the other structures discussed below. Over all stages excluding the preprocessing, $(n-1)\cdot\log n - n + 1$ PG blocks are required. Of these blocks, $n-2$ blocks do not need to compute the propagate term. In the preprocessing step, which is the same for all parallel-prefix adders, n generate functions and $n-1$ propagate functions are needed. A summary of the required numbers of all the basic blocks for all adder types is provided in table Table 6.2. By combining Table 6.1 and Table 6.2, we can summarize that an n-bit Kogge-Stone adder requires $(n-1)\cdot\log n - 3\cdot n + 1$ XOR gates and $2\cdot(n-1)\cdot\log n - 2\cdot n + 3$ AND gates.The number of elementary operations per adder is depicted in Table 6.3.

Masking KSAs When compared to the other designs discussed in this work, an implementation of a KSA requires the highest number of PG-blocks, resulting in the largest area requirement when implemented using masking. As dth-order secure HPC2 AND-Gadgets require $d\cdot(d+1)/2$ bits of fresh randomness, the total randomness requirement of a HPC2-masked n-bit KSA is the highest of the discussed variants at $d\cdot(d+1)\cdot((n-1)\cdot(\log n - 1)+1)$ bits. However, the required randomness can be drastically reduced if only univariate security is considered, as shown in [SMG15a]. Here, a first-order secure TI of a KSA using 3 shares that only needs n bits of fresh randomness is proposed. As noted by the authors, their second-order variant does not provide protection against multivariate attacks.

6.3.2 Sklansky Adder (SA)

The Sklansky Adder was introduced in 1960 [Skl60] as an efficient parallel adder with low area requirements. It has the same latency as a KSA at $\log n + 1$ cycles but requires a lower total number of PG blocks. In contrast to the previously described circuit, the number of PG blocks per stage is constant at $n/2$. Figure 6.2 depicts an 8-bit Sklansky adder, realized with 12 PG blocks in 3 stages, excluding preprocessing.

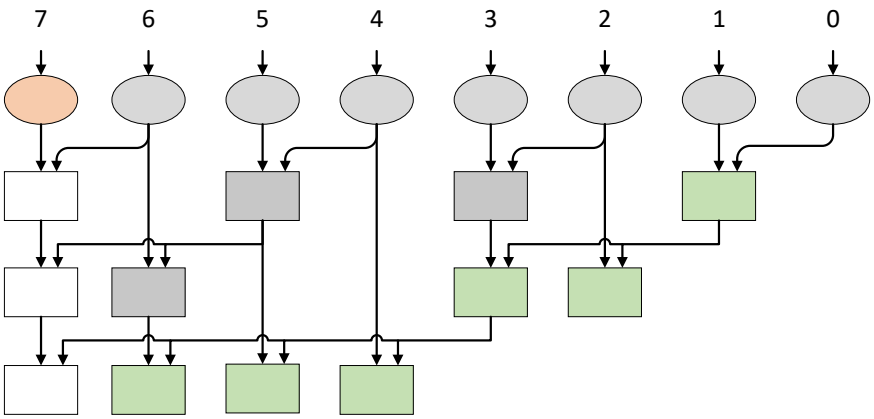


Figure 6.2: 8-bit Sklansky Adder.

In the first stage, every other PG-term is combined with its neighbor, i.e.,

$$PG(\{2 \cdot i + 1, 2 \cdot i + 1\}\ ,\ \{2 \cdot i, 2 \cdot i\})\ \forall 0 \leqslant i < n/2$$

is computed. The second stage combines these 2-bit PG-terms as

$$PG(\{\left\lfloor 4 \cdot \frac{i}{2} \right\rfloor + 2 + (i \bmod 2), 4 \cdot \left\lfloor \frac{i}{2} \right\rfloor + 2\}\ ,\ \{4 \cdot \left\lfloor \frac{i}{2} \right\rfloor + 1, 4 \cdot \left\lfloor \frac{i}{2} \right\rfloor\})\ \forall 0 \leqslant i < n/2.$$

In general, the k-th stage for $k > 0$ combines the previous PG terms using $n/2$ PG-blocks in the following way:

$$\begin{aligned} PG(\{&\left\lfloor \frac{i}{2^k} \right\rfloor \cdot 2^{k+1} + 2^k + (i \bmod 2^k), \left\lfloor \frac{i}{2^k} \right\rfloor \cdot 2^{k+1} + 2^k\}\ , \\ &\{\left\lfloor \frac{i}{2^k} \right\rfloor \cdot 2^{k+1} + 2^k - 1, \left\lfloor \frac{i}{2^k} \right\rfloor \cdot 2^{k+1} + 2^{k-1} - 1\})\ \forall 0 \leqslant i < n/2 \end{aligned}$$

Following the reasoning of section 6.3.1, $(n/2 - 1) \cdot \log n$ PG blocks are needed to build a SA when excluding the preprocessing stage. As in in the Kogge-Stone case, $n - 2$ of these do not need to compute the propagate term. The resulting number of required the basic blocks is provided in table Table 6.2.

Masking SAs When compared to a KSA, the main advantage of an SA in the context of masked implementations lies in the reduced number of required PG blocks, which directly results in a lower area requirement. If HPC2 gadgets are used this also leads to reduces randomness use of $d \cdot (d + 1) \cdot ((n - 2) \cdot \log n - n + 1)$ bit.

If only first-order security is desired, a TI similar to [SMG15a] can be used to reduce the randomness requirements in comparison to HPC2 gadgets. In this publication, the authors re-use shares of the generate terms to reduce the required fresh randomness in the computation of the

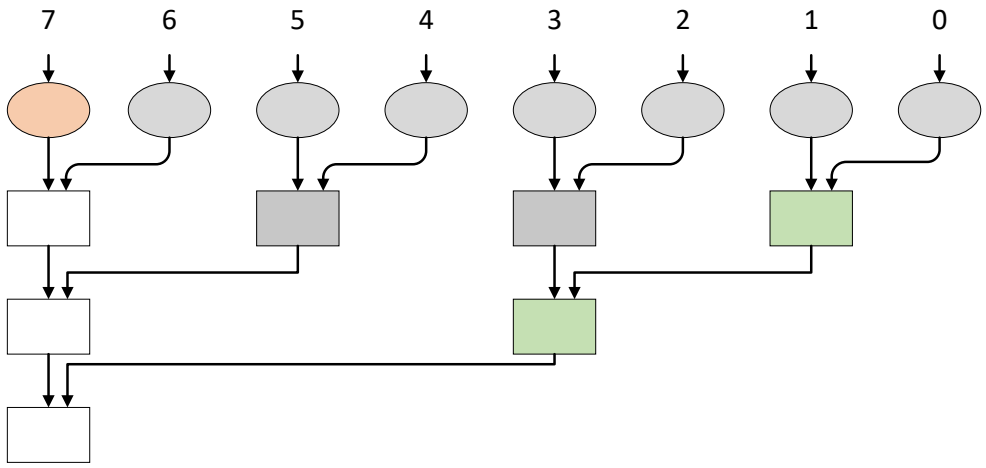


Figure 6.3: 8-bit Brent-Kung Adder, generation of the MSB carry bit.

propagate term. Specifically, when computing $p_{\{i,j\}}$ as a masked version of Equation 6.2, the first share of $g_{\{k,j\}}$ is used to achieve the uniformity of the tuple $(p_{\{i,j\}}, g_{\{i,j\}})$. In the KSA-case, each p-term is not used more than once per stage as the rightmost term in Equation 6.2. Therefore, each g-term is not used more than once to replace a bit of fresh randomness, preventing potential violations of the joint uniformity of the pg tuples in later stages. Unfortunately, this randomness reduction approach can not directly be applied to SAs, due to the higher fan-out of the PG-blocks of up to $n/2 - 1$ in this case. Therefore, SAs require additional randomness when masked with TI. However, the number of required fresh random bits can be reduced by following the the construction of the second-order secure KSA presented in [SMG15a]. Here, the authors take four shares from $g_{\{k,j\}}$ to replace fresh mask bits. Following this approach, instead of taking the same first share as mask replacement, we can use up to three different shares. In stages with a fan-out higher than three, additional fresh random bits need to be inserted. This results in 27 additional random bits required by a 32-bit first-order secure SA when masked with TI.

6.3.3 Brent-Kung Adder (BKA)

A BKA [BK82] allows a further reduction in the number of PG blocks in comparison to an SA, albeit with the cost of increased latency. The general structure of BKA is composed of two trees, where the first tree computes the group generate and propagate terms of increasingly larger groups of bits until the carry bit for the most significant output can be determined. The resulting structure can be viewed as a binary tree where the output carry for the most significant bit represents the root and the PG-terms generated by the preprocessing represent the leafs. An 8-bit version of this tree is shown in Figure 6.3. As PG-terms, which are not required to compute the most significant carry bit, are not considered in this process, a second reversed tree is needed in order to generate them. A full 8-bit BKA that can compute all bits of the final

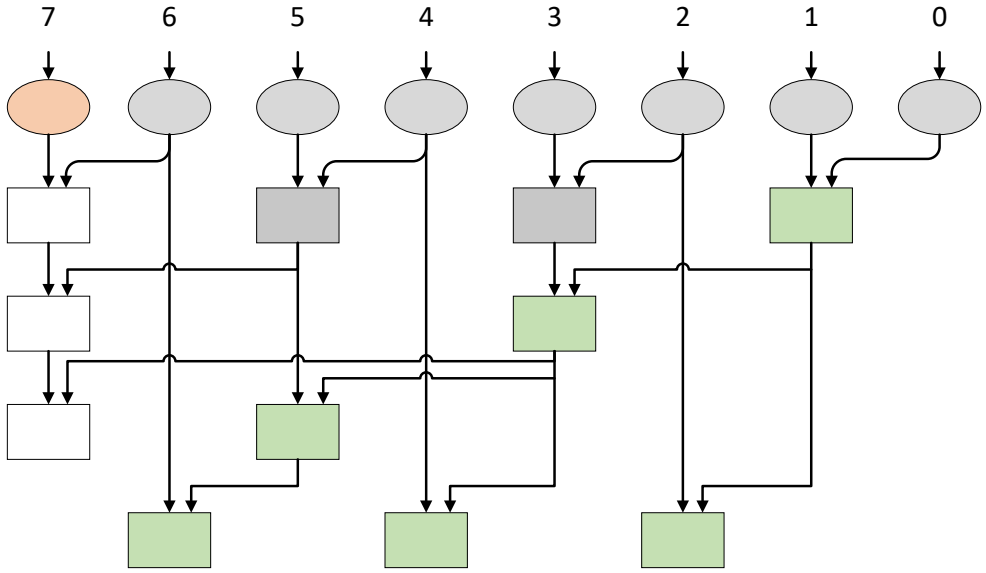


Figure 6.4: Complete 8-bit Brent-Kung Adder.

sum is depicted in Figure 6.4. Note that in this 8-bit case, the carry for the most significant bit is already available after $\log 8 = 3$ stages. However, an additional stage is required for the reverse tree, which computes the remaining carry bits. As a result, the 8-bit BKA requires a total of 4 stages and 11 PG blocks, excluding the preprocessing stage.

The first stage in a BKA is the same as for the SA, combining every other PG term with its neighbor and therefore generating $\frac{n}{2}$ 2-bit PG terms. Following the tree structure of BKAs, the second stage combines adjacent 2-bit PG terms to 4-bit terms as:

$$PG(\{4 \cdot i + 3, 4 \cdot i + 2\}\ ,\ \{4 \cdot i + 1, 4 \cdot i\})\ \forall 0 \leqslant i < n/4.$$

This pattern is repeated until the final carry can be computed, requiring $\log n$ stages. The k-th stage therefore computes PG-terms for increasingly larger groups of bits using $\frac{n}{2^{k+1}}$ PG-blocks according to:

$$PG(\{i \cdot 2^{k+1} - 1, (2 \cdot i - 1) \cdot 2^k\}\ ,\\ \{(2 \cdot i - 1) \cdot 2^k - 1, (i - 1) \cdot 2^{k+1}\})\ \forall 1 \leqslant i < \frac{n}{2^{k+1}}.$$

The resulting structure consists of $n - 1$ PG blocks, of which $\log n$ are only needed for the output carry generation. As explained above, this binary tree does only generate the group PG terms that are necessary to calculate carry for the MSB. The reversed tree is therefore inserted to the circuit below the initial binary tree, generating the remaining terms. In order to improve the readability of the equations describing the PG term generation, we count the stages of this subtree beginning with the output stage of the adder, i.e., the stage number l of the inverse tree

is related to the stage of the complete adder k through $l = 2 \cdot \log n - 2 - k$. On the lowest level ($l = 0$), the inverse tree generates all even PG terms, i.e., $pg(\{2 \cdot i, 0\})\ \forall 1 \leqslant i < n/2$ as:

$$PG(\{2 \cdot i, 2 \cdot i\}\ ,\ \{2 \cdot i - 1, 0\})\ \forall 1 \leqslant i < n/2.$$

In general, the inverse tree in a BKA computes $\frac{n}{2^{k+1}} - 1$ PG terms in $\log n - 1$ stages as:

$$PG(\{i \cdot 2^{l+1} + 2^l - 1, i \cdot 2^{l+1}\}\ ,\ \{i \cdot 2^{l+1} - 1, 0\})\ \forall 1 \leqslant i < \frac{n}{2^{l+1}} - 1.$$

The total number of PG terms in this lower tree is equal to $n - \log n - 1$.

When joining both tree structures, the last stage of the upper tree and the first stage of the lower tree can be combined in one stage, as there is no direct dependency between them. If no carry output is required, this step is trivial as the last stage of the upper tree can be omitted. As a result, a BKA can be realized in with a total of $2 \cdot (n - \log n - 1)$ PG blocks in $2 \cdot \log n - 2$ stages, when no carry output is computed.

Masking BKAs The asymptotic complexity of the number of PG blocks in a BKA is $\mathcal{O}(n)$ in comparison to $\mathcal{O}(n \cdot logn)$ in the KSA and SA cases. This leads directly to smaller implementations if masking countermeasures are employed. Additionally, as the number of AND-gates has the same linear complexity, the required number of fresh random bits is reduced further to $d \cdot (d + 1) \cdot (3 \cdot n - 4 \cdot \log n - 1)$. These improvements are bought with a higher latency due to BKAs requiring $2 \cdot \log n - 2$ stages, while KSAs and SAs can be realized in only $\log n$ stages.

However, if a first-order TI is used as the masking scheme, the randomness requirements are higher than in the KSA case. As BKAs have a maximal fan-out of $\log n$, additional randomness is needed, similar to the SA case. Due to the tree-like structure of the BKA the randomness overhead is not as severe as for SA adders, amounting to only 3 additional bits in a 32-bit adder.

6.4 Implementation Results and Discussion

This section provides implementation details for the proposed addition structures. We implemented 32-bit versions of the three algorithms as this size is commonly needed in cryptographic algorithms such as SALSA20. Note that adaptions to other widths are trivial. All designs were specified in VHDL and VERILOG and were synthesized for a Xilinx Spartan 6 XC6SLX75 FPGA with speed grade -3 using Xilinx ISE 14.7. In order to assure the correct realization of masked gadgets the relevant signals were exclude form optimizations and the design hierarchy was preserved. The implementation results are presented in Table 6.4.

6.4.1 TI Implementations

If first-order security is sufficient in an application, TI is a valid choice for a masking scheme. Regarding area, our results for the KSA adder are similar to the figures from [SMG15a]. The clock frequency that was estimated by the synthesis tool differs significantly from our results, although the authors performed their benchmark on an FPGA from the same family as we did. Different speed grades of the devices and different constraining of the synthesis might explain this discrepancy. The SA design utilizes less logic than the KSA, which can be attributed to the

Table 6.4: Implementation results for different 32-bit adder designs.

Design	Flip-Flops	LUTs	Freq (MHz)	Latency	Rand.
TI KSA [SMG15a]	1416	1008	197	6	32
TI KSA	1330	937	62	6	32
1st-order HPC2-KSA	3869	3197	143	12	249
2nd-order HPC2-KSA	7646	4427	132	12	747
TI SA	1423	804	216	6	59
1st-order HPC2-SA	3093	2219	180	12	119
2nd-order HPC2-SA	5761	2959	123	12	357
TI BKA	1749	839	220	9	35
1st-order HPC2-BKA	3207	2045	216	18	74
2nd-order HPC2-BKA	5518	2677	174	18	222

lower number of PG blocks. The number of flip-flops is very similar, as it is dominated by the number of stages in a pipelined architecture. The clock frequency can be increased slightly, but the higher randomness requirement offsets the advantages in many applications. When masking using TI, the BKA does not pose any advantages when compared to the other designs. In spite of requiring the lowest number of PG blocks, the number of LUTs is higher than for the SA adder, due to the higher number of stages leading to higher routing overhead.

6.4.2 HPC2 Implementations

If resistance against multivariate attacks is demanded, the KSA shows the worst performance in all categories. It should therefore not be considered in this scenario. The SA can effectively be utilized if the latency of the adder is the most important factor. The logic and memory requirements are significantly lower and the randomness requirement is less than half when compared to the KSA. If latency is less important, the BKA should be preferred. While its area is only slightly lower, the randomness requirement can be further reduced by almost 40 % in comparison to the SA. Due to the differing asymptotic complexity this difference increases for larger widths.

6.5 Conclusion

In this chapter we study three adder designs regarding their suitability for boolean masking. The algorithms where masked with the TI scheme for first-order security and with HPC2 gadgets that provide resistance against multivariate attacks. After a detailed complexity analysis and practical realization on an FPGA we found different scopes of application for the algorithms. The KSA can effectively be used to achieve univariate security. If randomness requirements are less important than area, a SA can be considered. In the multivariate case the SA provides the lowest latency, while the BKA can reduce area and randomness requirements at the cost of latency.

Part IV

Side-Channel Resistance for Integrated Systems

Chapter 7

A Side-Channel Protected Processor for ARX-based Cryptography

ARX-based cryptographic algorithms are composed of only three elemental operations — addition, rotation and exclusive or — which are mixed to ensure adequate confusion and diffusion properties. While ARX-ciphers can easily be protected against timing attacks, special measures like masking have to be taken in order to prevent power and electromagnetic analysis. In this chapter we present a processor architecture for ARX-based cryptography, that intrinsically guarantees first-order SCA resistance of any implemented algorithm. This is achieved by protecting the complete data path using a Boolean masking scheme with three shares.
We evaluate our security claims by mapping an ARX-algorithm to the proposed architecture and using the common leakage detection methodology based on Student's t-test to certify the side-channel resistance of our processor.
The results presented in this chapter were published at DATE 2017 [BSMG17].

Contents of this Chapter

7.1 Introduction **95**
7.2 Preliminaries **96**
7.3 Design Considerations and Technical Description **97**
7.4 Implementation **101**
7.5 Evaluation **102**
7.6 Conclusion **104**

7.1 Introduction

When designing security-critical digital devices, the security system designers are often faced with two contradictory challenges. On the one hand, they are required to build and integrate a robust cryptographic subsystem that is efficient but resistant against any type of (physical) attack. On the other hand, they must create a lifetime-secure but agile system with included migration paths to allow updates of cryptographic components in the field, if necessary. While the first requirement indicates a hardware implementation that combines computational efficiency and physical protection, the second update criterion demands a software-like implementation. In this context FPGAs can be one viable solution for some situations, but in many lightweight

contexts (e.g., Smart Cards) they are not applicable for several reasons. Hence, for a conventionally combined setting (i.e, using hardware-software co-design) the implementation of a holistic security concept is far from trivial and requires particular care and security expertise from both hardware and software engineers.

7.1.1 Contribution

In this work, we present a novel co-processor subsystem that is designed as an Application-Specific Instruction-Set Processor (ASIP) for a specific class of cryptosystems with inherent hardware resistance against side-channel analysis. More precisely, our architecture and instruction set follows principles from the Threshold Implementation (TI) concept that is known to provide provable security against power side-channel analysis. As an ASIP it can be loaded with software implementations of different symmetric ARX-based cryptographic primitives, such as stream and block ciphers or hash-functions without the need for adaption of the hardware. We show that a prototype of our design including a software implementation of Speck is not only secure against first-order side-channel analysis and timing attacks but can be realized at moderate costs that are even comparable against pure (protected) hardware implementations. Note that the hardware is designed to completely counter the aforementioned side-channel attack which significantly relaxes the requirements for software engineers to handle complex constraints of physical side-channel security.

7.1.2 Related work

While we provide the first ASIP specifically built for ARX-based symmetric cryptosystems, the authors of [Gro15] had previously reported on an approach to protect an existing microcontroller architecture with a power side-channel countermeasure - yet without practical evaluation. The work [SKR+13] provides a dedicated accelerator for ARX-based cryptography, which does not include a consideration on physical attacks. Finally, we refer to the proposals for side-channel resistant (hardware) implementations of ARX-based constructions as reported in [STE15, CITE15].

7.2 Preliminaries

7.2.1 ARX Algorithms

ARX-based cryptography is denoting a set of symmetric constructions that are purely based on Additions (mod 2^n), Rotations and XOR (ARX). In [KN10] the set of ARX operations was proven to be functionally complete over $\mathbb{Z}_{2^n}$. The main advantage of ARX cryptosystems is their compact and fast instantiation on most instruction-set architectures, including resistance against timing attacks by design. Examples for ARX-based constructions are the block ciphers FEAL, Threefish or Speck; the stream ciphers Salsa20, ChaCha, HC-128 or the hash functions BLAKE and Skein. While there is a wealth of different ARX-constructions, we will focus in this work on the Speck block cipher and the Salsa20 stream cipher as case studies.

Speck

Speck is a recently proposed and lightweight ARX-based block cipher optimized for software implementations presented in [BSS+13]. It is specified for block sizes between 32 and 128 bit and uses key sizes between 64 and 256 bit. The operand length for the elemental ARX-operations is half of the block size. In each round, the state is updated using a very simple Feistel-like function, defined as

$$x_{i+1} = ((x_i \gg \alpha) + y_i) \oplus k_i,$$
$$y_{i+1} = x_{i+1} \oplus (y_i \ll \beta)$$

where (x_i, y_i) is the current state, k_i is the round key, and α and β are constants. The Speck key schedule is very similar to its round function and is computed as

$$l_{i+m-1} = (k_i + l_i \gg \alpha) \oplus i,$$
$$k_{i+1} = k_i \ll \beta \oplus l_{i+m-1}.$$

where m is the chosen number of key words, $(l_{m-2}, \ldots, l_0, k_0)$ is the key and k_i is the ith round key.

Salsa20

Salsa20 is a lightweight ARX-based stream cipher presented in [Ber08] and a final candidate in the eSTREAM hardware profile. The keystream is generated by calculating a hash of the 256-bit key, a 128-bit constant, a 64-bit nonce and a 64-bit block number using the Salsa20 hash function. The initial 512-bit state is composed by arranging these inputs in a 4-by-4-word matrix containing 32 bit in each entry. After 20 iterations of the round function the result can be used as a keystream. In each round every column is treated separately while each word is updated once, starting with the below-diagonal words. Let x_i be the word in column i that is to be updated, then the update process is defined as:

$$x_i \leftarrow \left(\left(x_{(i-1) \bmod 4} + x_{(i-2) \bmod 4}\right) \ll 7\right) \oplus x_i.$$

After each round the state is transposed. A study analyzing the susceptibility of the phase three eSTREAM candidates towards SCA attests Salsa20 exploitable SPA and DPA vulnerabilities (especially in its rotate function), while assessing the cost of countermeasures as high [GBC+08]. A successful (simulated) Correlation Power Analysis (CPA) attack on Salsa20 is presented in [MAS15].

7.3 Design Considerations and Technical Description

Our system is designed as a generic and intrinsically SCA-protected co-processor platform for ARX ciphers. It supports arbitrary ARX algorithms by updating the program code, without the need for adapting the hardware design. In particular, we merge the concept of the TI countermeasure into our SPARX microarchitecture instead of applying it to a dedicated cipher implementation. For this purpose, we designed an application-specific CPU with a TI-protected

ARX-ALU that is provably secure against first-order attacks (using $d = 3$ shares). Certainly, higher-order attacks can also be prevented by increasing d at higher hardware costs. Our system is designed to separate any (SCA-critical) data flow from the control flow.
The fourfold pipelined architecture is based on a RISC approach and incorporates two separate ALUs. The side-channel protected ALU performs all elementary ARX operations on a protected register file with direct access to a source of randomness that is required for the addition operation. We further identified that a dedicated unprotected ALU, which operates on a dedicated register file, is beneficial to increase the overall performance at reasonable costs. Load and store instructions are available for moving data between the RAM and the register files. The high-level structure of SPARX is provided in Fig. 7.1.

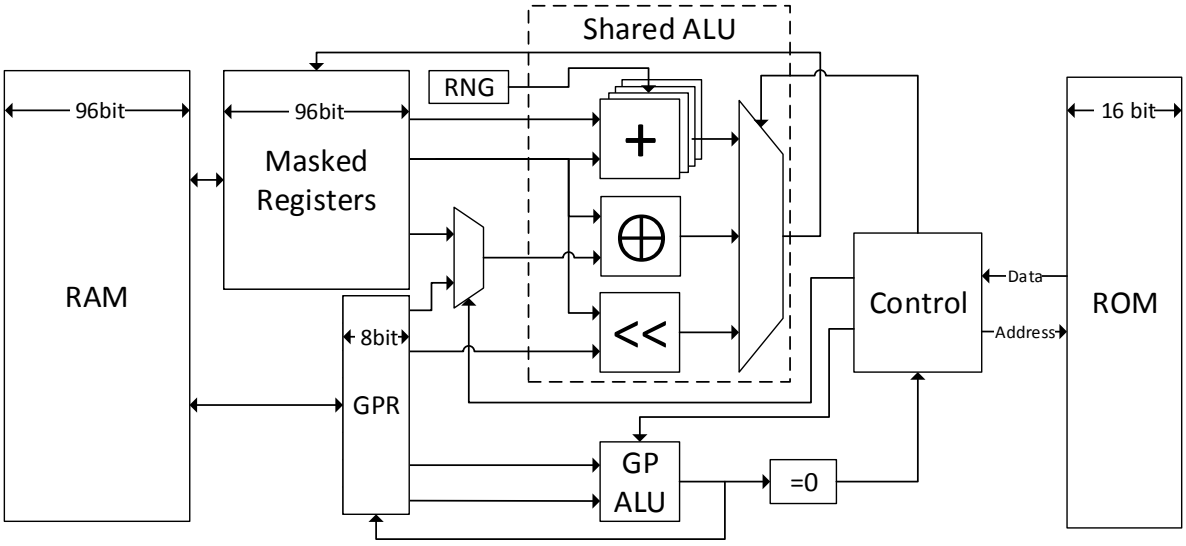


Figure 7.1: High-Level Block Diagram of SPARX

7.3.1 Protected ARX-Specific ALU

The main ALU operates on triple-shared 32-bit words and is used for all calculations on the sensitive state of the implemented ciphers. It consists of a TI-protected adder, an `xor` and a rotation unit and is connected to a dedicated register file. Because both, the `xor` and rotation operations are linear, ensuring that each share is processed independently is sufficient for the masked implementation. However, addition in $\mathbb{Z}_{2^{32}}$ is non-linear. The construction of a TI-conform shared representation of the addition is non-trivial.
We therefore refer to the discussion of this issue in [SMG15b]. Schneider *et al.* proposed two different types of addition circuits for Boolean-masked values which follow the TI principle. For our purpose, we use a similar variant of their proposal based on the ripple-carry adder as it is far more area-efficient for lightweight application than their presented Kogge-Stone adder, while only requiring four bits of fresh randomness per operation. In addition, we found that a slight modification of the original blinded addition circuit can enhance the versatility of our processor. More precisely, in [SMG15b] the additions only take the two summands as input and no initial carry. However, it is easily possible to tweak the original design to include this

capability without breaching the security assumptions. To this end, we require that the input carry is uniformly shared, which is implicitly given if it is the output carry of a previous addition. As the output carry shares are not independent of their related sum bits they must not be used as joint inputs to a shared function. Our design accounts for that by only using the carry bits for adding multiple 32-bit blocks. With this tweaked adder, our processor can support multi-precision addition of inputs larger than 32 bits (e.g., for Speck128/256, Blake2b or Threefish) without negatively affecting the performance of the single-limb 32-bit addition.
Note that the adder is the slowest element in the overall design, requiring 32 cycles to complete one 32-bit addition. In order to still allow for high throughput the module is provided with a separate clock running at double the frequency of the main clock. This reduces the latency of one addition to 16 cycles. In order to increase the CPU's resource utilization, the adder is connected asynchronously to the rest of the processor using two instructions: one for starting the addition and one for retrieving the result. The retrieve instruction returns the current state of the addition, regardless of whether the operation has finished or not. It is the responsibility of the programmer or compiler to ensure that a retrieve operation is only issued after at least 16 cycles have passed since the addition has started. This task is trivial and can easily be automated because in every cycle exactly one new instruction is fetched.
In order to further increase the throughput of SPARX four parallel addition units were instantiated. This reduces the average number of required clock cycles per addition from $16 + 2 = 18$ to $(16 + 8)/4 = 6$ including the time needed for the add- and retrieve-instructions. Incorporating more than four adders provides diminishing returns in performance and could not be efficiently exploited by most ARX algorithms.

7.3.2 Auxiliary General-Purpose ALU

For increased efficiency, SPARX contains an unprotected 8-bit auxiliary ALU that supports general purpose arithmetic and logical operations in particular for control flow operations. It provides single-cycle addition, subtraction, `and`, `or` and `xor` operations, each with either two register operands or one register and one immediate value. Furthermore, the auxiliary ALU can be used to compute counters and flags to control the key-independent program flow, without occupying the main TI-adders. It can also safely calculate round constants and other inputs to the cryptographic algorithm. Sensitive data cannot be leaked by the auxiliary ALU, because there is no connection between the protected register file and the unprotected ALU. Unmasked data can be loaded from and stored in the RAM to enable interaction with the control flow from outside of the CPU. This is useful to dynamically select a cipher algorithm or to generate an "encryption-done" flag for an external main CPU.

7.3.3 Data and Control Flow

SPARX is designed based on a standard RISC architecture. The pipeline is composed of four stages, namely *Fetch*, *Decode*, *Execute* and *Writeback*. The design relies on single-cycle read-latency memory and therefore uses neither data- nor instruction-caching. The architecture does not incorporate a stack or `call` and `return` operations to enable function calls, as cryptographic primitives do not benefit from this in general. It does, however, support branches and loops which can reduce the code size for round-based algorithms – like ARX ciphers – significantly. In order to control the program flow, two branching operations are implemented: an unconditional

`jump` and a branch-not-zero (`bnz`) instruction. The condition for the `bnz` instruction is generated by comparing the result of the general purpose ALU operations to zero. Masked data cannot be used as input to this process, which ensures that the execution times of all programs running on the proposed architecture are data independent, rendering the design resistant to timing-attacks. In conclusion, there are no operations that stall or flush the pipeline, which results in a throughput of one instruction per cycle.

7.3.4 Memory Configuration

Our SPARX processor is based on a Harvard architecture, i.e., it has as separate program and data memory. This enables increased instruction throughput without requiring another memory port and cleanly supports different widths for data and instruction words.
Each instruction word is encoded in 16 bits so a program memory of the same width can easily provide one instruction per cycle. The program memory size is limited to 4096 words. For comparison, our implementations of Speck and Salsa20 need 113 and 310 instruction words respectively.
The data port width is 96 bit to natively support access to 32-bit masked values. Only direct addressing of RAM data is supported. The amount of RAM is not fixed but the instruction-word width limits its size to 512 96-bit words. Besides for buffering data, the RAM is also used as IO interface.
In order to improve the throughput while keeping the number of pipeline stage reasonably low, the processor has access to dedicated registers. However, it is imperative to isolate the masked sensitive data from the unmasked general purpose data in order to prevent information leakage. To this end, two separate register files were implemented in the proposed architecture. The eight general purpose registers are 8 bit wide and can be used for storing rotation offsets, round constants, counters or flags. The second register file is used as a working memory for the sensitive data words SPARX is operating on such as the key, the plaintext and the ciphertext. The number of shared registers had to be considered carefully because each register is 96 bit wide and therefore costly in terms of hardware resources. In order to maximize the utilization of the four adders, up to eight operands / registers are optimal. While some ARX-algorithms could benefit from more than eight registers, the extra hardware cost does not pay off for most scenarios. As an example, Salsa20's throughput could be increased by 12% when 12 registers are available but, according to our benchmarks as described in Section 7.5, at an additional area cost of more than 16%

7.3.5 Data Separation

In order to ensure SCA-protection for arbitrary ARX-implementations, information flow from the protected data to the unprotected auxiliary data must be prohibited. Otherwise, information could be leaked although the ARX-primitives have been securely implemented. This is ensured by instantiating separate registers for the critical masked data and the auxiliary non-masked data.
The protected `xor` module operates on either two masked values or one masked and one non-masked input. In both cases masked output values are generated. This feature can significantly improve performance for algorithms relying on round constants: Because the constants are public and do not need to be protected they can be computed using the general purpose ALU

instead of the slower, more restricted ARX ALU. The `xor` of the non-masked value B and the shared value $A = A_1 \oplus A_2 \oplus A_3$, which is internally represented as the triple $\tilde{A} = (A_1, A_2, A_3)$, is computed as $\tilde{A} \oplus B = (A_1, A_2, A_3 \oplus B)$. The resistance of SPARX against SCA is not harmed during this operation because only one share of the masked value is modified by the linear and invertible `xor` function.
For the rotation, the value to rotate is always masked while the rotation offset is non-masked. This is not an issue as long as the offset does not depend on secret information. Preventing information flow from the masked to the non-masked data through main memory access is not enforced in hardware. Hence, the compiler must ensure that sensitive data is never loaded into non-masked registers.

7.4 Implementation

In this section we describe the technical hardware instantiation of the SPARX processor and a set of ARX-based ciphers in software.

7.4.1 Hardware Instantiation of the SPARX Processor

The Instruction Set Architecture of SPARX was specified using the Language for Instruction Set Architecture (LISA) in version 2.0. In order to develop a working prototype of the processor, Synopsis Processor Designer was used to generate HDL code, an assembler and a simulator for the proposed architecture. The rotate-module, the TI-adder and the required RNG-component were hand-coded in VHDL because LISA does not directly support these structures or the generated code was inefficient.

7.4.2 SPARX-Compliant Cipher Implementations

Two ciphers were implemented for SPARX as case studies.

Speck

The Speck variant with a 64 bit state and 128 bit key that uses 27 rounds when mapped to the platform. In a straight-forward implementation of Speck only two additions, one for the round function and one for the key schedule, are computed per round such that two of SPARX' four TI-adders remain unused. For improved resource utilization, block parallelism with Speck can be used that performs three encryptions and computes the key schedule at once. In our implementation, the rotation of x_i is performed in each round first, after which the additions are started. While the adders are operational, the y_i are bitwise rotated, the round counter is incremented, l_i is stored in the main memory and l_{i+1} is loaded into a register. While load/store-operations are necessary due to SPARX' limited register count, they do not reduce the throughput because the following computations have to wait for the result of the addition either way. After retrieving the addition result, the rest of each round is computed. Our Speck implementation consists of 113 16-bit instruction words and can encrypt three 64-bit words in 1057 cycles.

Salsa20

As the Salsa20 round function operates on four independent columns, its implementation can directly utilize the four addition units of SPARX. Except for the first four words updated in each round, every word depends on the previous computations, which limits the possibilities for reordering the operations with respect to latency of our TI-adder. The 16-word state of Salsa20 is too large to fit into the register file but virtually all necessary load/store-operations can be performed while waiting for the adders. Because every other round operates on rows instead of columns, two rounds of Salsa20 were unrolled in order to avoid the overhead associated with transposing the state. The resulting program contains 310 instructions and takes 2937 cycles to compute 512 bit of key-stream.

7.5 Evaluation

In this section we evaluate the SCA protection of the proposed architecture, provide size and performance figures and compare them to previous publications.

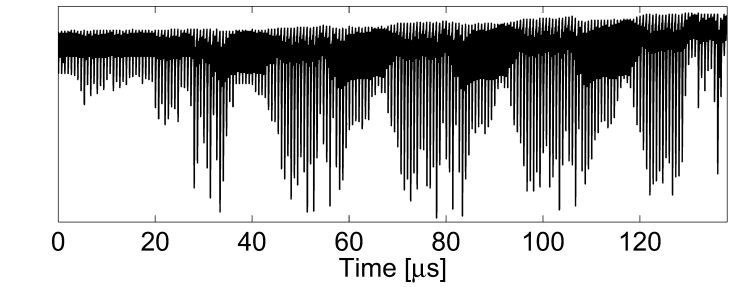
7.5.1 Leakage

As stated in the preliminaries, correctly implemented TI-based masking provides provable resistance against power- and EM-attacks. In practice however, ensuring that the TI requirements are actually met in the synthesized design can be difficult, especially for more complex architectures. In order to practically assess SPARX' resistance against SCA attacks, we made use of a SAKURA-G board [SAK] as an SCA evaluation platform. The design was mapped to a Xilinx Spartan6 XC6SLX75 FPGA, and a 4-round version of our Speck implementation was realized as the target ARX algorithm. Such a reduced-round implementation has been chosen to shorten the power traces in order to accelerate the measurement and evaluation processes, while still utilizing all the processor's components. The power traces have been collected by a digital oscilloscope at a sampling rate of 500 MS/s while the design was clocked at a frequency of 3 MHz thereby obtaining clear traces with minimal level of noise where the power peaks of adjacent clock cycles are not overlapping.

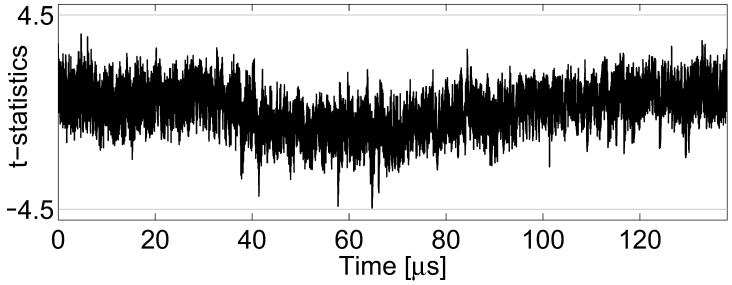
As the evaluation metric, we applied a leakage detection scheme instead of an arbitrary attack vector. The non-specific t-test – known as *fixed versus random* t-test – has been employed to examine the ability of the design to prevent SCA leakages. For more detailed information we refer to the original articles [GJJR11, SM15]. For such an evaluation, we collected 10 million power traces, one of which is shown in Figure 7.2a. The t-test results at first and second orders are shown by Figure 7.2, which confirm the robustness of our construction against first-order attacks. Since first-order TI is the underlying masking scheme, the design as expected exhibits (although small) higher-order leakages, which should in practice complicate the feasibility of higher-order SCA key-recovery attacks.

7.5.2 Performance and Size

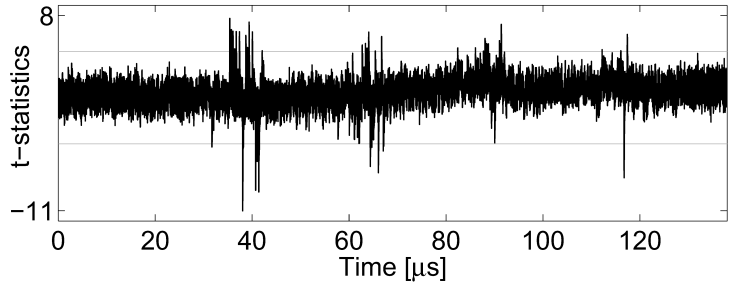
The performance of SPARX was evaluated for two different target architectures: a Xilinx Spartan-6 FPGA and the Faraday 180nm ASIC process. The bitstream for the FPGA was generated using Xilinx ISE 14.7 with the optimization goal speed. In order to prevent the



(a) sample trace



(b) 1st-order t-test



(c) 2nd-order t-test

Figure 7.2: A sample power trace, and the result of "fixed versus random" t-test using 10 million traces

mapping tool from using the same Look-Up Table (LUT) to compute functions on bits of independent shares, and thereby violating the TI requirements, the design hierarchy was maintained during synthesis. This resulted in a total number of 1519 used slices and a maximum clock-frequency of 112 MHz. The ASIC netlist was created with Synopsis Design Compiler 2010.12. To ensure that the security assumptions are not violated, ungrouping, flattening and register re-timing were disabled for the synthesis. The total number of cells required by the design amounts to 40358 NAND-gate equivalents. The sizes of SPARX' main modules for the ASIC as well as the FPGA process are provided in Table 7.1. In the FPGA results, slices that are used in more than one module, are counted multiple times.
The throughput of our Speck implementation is 20.3 Mbit/s on the FPGA and 75.7 Mbit/s on the ASIC, while the Salsa20 core can encrypt 19.5 Mbit/s or 69.7 Mbit/s, respectively.

Table 7.1: Relative Area Cost of SPARX' Modules

Module	Size on FPGA (Slices / %)	Size on ASIC (kGE / %)
Adders	471 / 31.0	18.1 / 44.8
Register File	424 / 27.9	9.5 / 23.7
Shifter	107 / 7.0	4.7 / 11.7
XOR	48 / 3.2	0.4 / 1.0
Total	1529	40.4

Comparison

While, to our knowledge, the proposed design is the first side-channel resistant, flexible ARX accelerator, several hardware implementations of ARX ciphers have been introduced in the literature. In [CITE15] a 99-slice SCA-resistant Speck implementation for FPGAs with a throughput of 9.7 Mbit/s is presented. The unprotected Speck implementation proposed in the same contribution achieves a performance of 10.1 Mbit/s using 42 slices.
The authors of [YH07] present several unprotected 180 nm-implementations of Salsa20 that are optimized towards either area or throughput. Their balanced iterative design can generate 255 Mbit/s at 23.4 kGE.
The unprotected high-performance ARX-accelerator CoARX[SKR+13] can generate a Salsa20-keystream at a rate of of 1.93 GBit/s running at 700 Mhz. CoARX achieves this speed when synthesized in a 90 nm process costing 95 kGE. While the algorithm running on CoARX can be selected via software, the supported algorithms must already be known at (hardware-) design time in contrast to our approach.

7.6 Conclusion

In this chapter we propose a flexible ARX-ASIP that intrinsically protects all implemented algorithms against timing and first-order side-channel attacks. The resistance of our implementation was verified practically by applying a well-established leakage detection scheme. The proposed architecture enables support for multiple ARX algorithms such as block cipher, stream ciphers and hash functions at the same time and permits updating of cryptographic algorithms in the field, while keeping the cost for securely adapting the software to changing requirements minimal.

Chapter 8

Automated Masking of Software Implementations on Industrial Microcontrollers

As shown in previous chapters, a gap between well-studied leakage models and observed leakage on real devices makes the application of these countermeasures non-trivial. This work provides a gadget-based concept to automated masking covering practically relevant leakage models to achieve security on real-world devices. We realize this concept with a fully automated compiler that transforms unprotected microcontroller-implementations of cryptographic primitives into masked executables, capable of being executed on the target device.
In a case study, we apply our approach to a bitsliced LED implementation and perform a TVLA-based security evaluation of its core component: the PRESENT s-box.
The work described in this chapter was published at DATE 2021 [ABB+21].

Contents of this Chapter

8.1 Introduction **105**
8.2 Concept **107**
8.3 Security in Practice **108**
8.4 Gadget Design **109**
8.5 Code Transformation **110**
8.6 Case Study **112**
8.7 Conclusion **115**

8.1 Introduction

Implementing countermeasures such as masking is still a tedious and error-prone task, especially for software countermeasures running on embedded devices. This is mainly due to the following characteristics:

The practical security of a countermeasure cannot be evaluated at a high abstraction level, such as C-code. This is due to the various translation steps to machine code that might add, remove, or reorder instructions along the way. Indeed, there are numerous examples of compilers breaking otherwise secure masking schemes [BCH+20, BGG+14].

Further, the device-under-test's internal architecture adds additional *device-specific leakage* way beyond a simple Hamming-weight or value-based leakage model.

For example, a register update will produce the Hamming-distance between the new and the old value. Correct first-order masking becomes insecure in practice if two shares k_0, k_1 of a secret value k are combined with such leakage behavior, and information on the secret k is leaked. Indeed, leakage across instructions invalidates not only the theoretical assumption of "independent leakage" but also prevents the re-use of security unaware compiler back-ends [PV17, BGG+14, MOW17, CGD18]. Most notable, this device-specific behavior also applies for different implementations of the same architecture (e.g., RISC-V), leading to broad diversity of physical side-channel behavior among devices even when executing the very same code.

8.1.1 Contribution

We tackle these issues by introducing a dependable compiler that automatically and effectively protects insecure code. The compiler provides automated side-channel resilience in practice by combining careful device-specific masking with secure code transformations. Our code transformations are independent of device-specific leakage behavior and allow users to adapt the approach to new devices. Informally, the approach consists of replacing vulnerable machine-code with "gadgets", which provide the same functionality in a power side-channel resilient manner. We formally verify our gadgets' security in precise device-specific leakage behavior models and describe the modeling process.

A proof-of-concept implementation of our approach for Arm Cortex M0+ microcontrollers is developed and applied to a bitsliced implementation of the LED block cipher. Finally, we demonstrate the effectiveness of our approach by a practical security evaluation of physical measurements.

8.1.2 Related work

Many papers try to automate certain aspects of the implementation of countermeasures. We focus on software countermeasures and briefly discuss the shortcomings of existing work in the following.

Insufficient leakage assumption for practical resilience et al. Moss provided one of the first compilers for unprotected C-code to masked assembly [MOPT12] but only rely on first-order masking and do not consider device-specific leakage. Eldib *et.* Wang [EW14] use synthesis to mask C-Code in LLVM Intermediate Representation (IR). Most notable, they indeed report that compiler transformations break the correctness of masking.

Dependence on security of off-the-shelf compilers et al. Barthe develop a fast compiler masking C-Code at higher-order in [BBD+16]. They depend on an off-the-shelf compiler to produce executable code. This likely breaks security as reported in [BCH+20, BGG+14]. Further, they do not consider device-specific leakage. Likewise, et al. Agosta [ABMP13] employ precise information flow analysis within LLVM but suffer from the same shortcomings, i.e., a potential break at LLVM's back-end and neglected device-specific leakage. Recently, et al. Belleville automate application of first-order masking, capable of masking tables [BCH+20]. It is implemented as middle-end compiler pass on LLVM IR and uses formal verification on machine code to analyze which back-end(s) potentially break the countermeasure. Indeed, they

report at least one harmful back-end pass for ARM architectures. However, their analysis is restricted to a basic value leakage model under the "independent leakage assumption" and is not backed by physical security assessment. However, this is crucial as multiple passes such as instruction scheduling and register allocation potentially break security in presence of leakage spanning across instructions. Instead, our approach consists of assembly transformation to prevent flaws from regular compilation.

et al. Wang [WSW19] repair register allocation of LLVM to prevent leakage arising from register re-use, but assume that the input code is already masked. Further, the modification is specific to the leakage model and cannot be easily extended to cover leakage across instructions. We note that fixing compiler passes according to model assumption most likely renders the entire compiler device-specific. Instead, our compiler passes are independent of any leakage model assumptions.

et al. Bayrak automatically insert first-order Boolean masking in machine-code [BRN+15]. While they do not suffer from potentially harmful compiler-backends, device-specific leakage is again out-of-scope. They also discuss pairing physical evaluation with compilation to focus solely on detectable leakage. Unfortunately, this mandates the presence of a measurement setup and respective expertise during compilation which somewhat contradicts the purpose of automated security.

Conceptually, practical resilience can also be achieved by combining automated masking with a subsequent automatic repair: Potentially insecure output of the first step could be secured by inserting more countermeasures, as e.g. discussed in [SSB+19]. One inherent issue here is that a repair based on inserting additional instruction is not always possible, especially if shares are combined during computation by compiler optimizations. Instead, our work provides systematic protection resulting in practically-resilient implementations while only relying on device-specific gadgets.

8.2 Concept

Our core concept consists of replacing vulnerable machine-code computing on sensitive data by invocations of gadgets that compute securely on masked data. In the end, a protected executable is returned, which can directly be run on the targeted platform. The overall compilation,

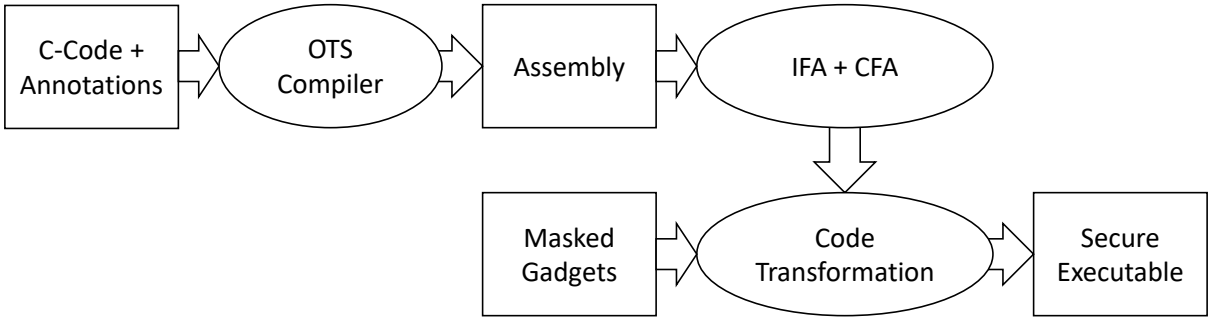


Figure 8.1: High-level structure of our proposed automated masking approach.

depicted in Fig.. 8.1, consists of three stages: (1) compilation of annotated but unprotected code with an optimizing off-the-shelf C-compiler (gcc), (2) information and control flow analysis

(IFA/CFA) to mark sensitive computations for transformation, and (3) the actual substitution of insecure instructions by secure gadgets.

The annotation for stage (1) is very simple and only consists of a specification of the sensitive inputs to the algorithm, e.g., the plaintext and key.

Stage (3) makes use of a library of gadgets specifically designed to achieve device-specific side-channel resilience. Pointers replace sensitive data in registers with masked data stored in memory. The compiler inserts code to handle memory, data, masking, and demasking of sensitive data, randomness, and the invocation of secure gadgets so that the same functionality is computed in a side-channel resilient manner.

The gadget-based approach succeeds whenever the sensitive part of the source program can be expressed by operations for which gadgets exist. We focus on bitsliced ciphers, which can be represented using Boolean masking. However, the concept could also secure other types of the input program, such as add–rotate–xor (ARX) ciphers when the required gadgets are provided (e.g., for addition).

The realization of such compilation results in two tasks: (1) construction of device-specific gadgets (Section 8.4) (2) vulnerability analysis and code rewriting (Section 8.5). Task (1) is performed by security experts prior to compilation and is re-used among code to be compiled. Task (2) results in a fully automated compiler using the gadgets of Task (1).

8.3 Security in Practice

Our goal is to reliably establish baseline protection against simple attacks such as Simple Power Analysis (SPA), Correlation Power Analysis (CPA), and first-order Differential Power Analysis (DPA) [KJJ99a]. Physical side-channel attacks can be classified according to the number of sample points they exploit per measurement of execution and the amount of measured executions in total. In the following, we establish resilience against attacks exploiting a single attacker chosen sample point for large amounts of repeated executions. Such protection forces adversaries to launch more complex attacks that require expertise to exploit multiple sample points. We mandate resilience to be established and evaluated without device-dependent noise or additional countermeasures (e.g., shuffling), which perturb physical measurements but can be removed by pre-processing the measurements.

First-order Boolean masking can provide protection in this setting as sensitive data k is split into shares k_0, k_1 such that $k_0 \oplus k_1 = k$ where one share is a uniform random value. Information on both shares must be recovered by an adversary to recover the masked secret k due to the involved randomness. Attacks exploiting a single measurement point cannot be successful if the two shares remain separated during computation and no measurement sample reveals information on more than one share [ISW03]. The latter constraint is important for resilience in practice as processors are well-known to emit leakage, combining shares during the execution of software [PV17, BGG+14, MOW17, CGD18]. In contrast to existing work, we take additional effort to ensure both constraints are met after compilation.

In [BGG+20] we show a link between side-channel resilience in practice and "Threshold Non-Interference" (NI), a formal notion of side-channel security, in detailed models of observable side-channel information (denoted "leakage models"). First order NI ensures that each observable side-channel information (leakage) depends on at most one share of each masked secret. The formal notion of security implies our intended degree of practical security in case the leakage model contains all information that can be gained by one physical measurement sample. Slightly

different, we exploit the indicated link to prevent human errors and speed up development: we consolidate experienced side-channel behavior in leakage models and use the formal verification tool scVerif [BGG+20] to ensure the absence of vulnerabilities arising from *known* leakage.

We briefly explain how our models are constructed, which consists of two tasks (a) deciding whether modeled leakage is present on a device and (b) systematizing unknown leakage behavior into formal models (the formal model is presented in [BGG+20]).
Validating the presence of modeled leakage can be done by constructing implementations that emit detectable leakage in physical measurements (e.g., using TVLA, Section 8.6.1). The leakage behavior under test must be activated while all other known leakage behavior must not cause detectable leakage. This can be achieved by filling the device state with secret-independent (e.g., constant) data, except for two carefully placed shares, which are combined by the leakage behavior under test (see [PV17] for a detailed discussion). The tool scVerif can be used to verify that known leakage behavior is not causing vulnerabilities in such programs. False positives are possible when unknown leakage behavior is triggered. This can be detected once the model becomes more complete as inconsistencies arise during the validation of another leakage (which is why model construction is an iterative process).

Systematization of unknown leakage behavior is not straightforward: Whenever leakage is detected, informally secure implementations, a hypothetical leakage is modeled according to human understanding. This can be supported by verification as the newly modeled leakage should lead to verification failure in the implementation. Often it is sufficient to revise existing leakage behavior. Subsequently, the validation strategy (a) is employed for the hypothetical leakage to validate its presence.

Our starting point was to validate the presence of published leakage behavior (e.g. [PV17, BGG+14, MOW17, CGD18]), which reduced the likeliness of false positives and quickly led to an almost robust model. We validated the completeness of our models by constructing masked implementations which are provably secure in our model and perform a physical assessment to detect gaps.

8.4 Gadget Design

The gadgets must provide the functionality of instructions supported by a microcontroller while being side-channel resilient in practice. The gadgets are handed over to the compiler in the form of a library, which allows it to construct them for a specific microcontroller and mitigate device-specific leakage behavior (known as “hardening”).

We briefly explain the process of hardening: Device specific leakage behavior allows observing information on more than one share within a single measurement point. Hardening means to prevent such vulnerabilities by breaking the leakage into separate measurement points by either (a) reordering instructions or operands and (b) inserting secret independent instructions (e.g., `NOP`). In addition, the gadgets must ensure that no sensitive data is left as residue, neither in memory nor in registers.

As the compiler is operating in a restricted setting, some technical subtleties arise: (1) The compiler operates on pre-sampled randomness provided in the form of a pointer, which must be correctly maintained by the gadgets to prevent vulnerabilities from randomness re-use. (2) The gadgets operate on dedicated pointers to memory for input and output, and in-place modification in the form of equal input and output pointer is possible, which requires loading shares

before writing outputs. (3) For some leakage behavior, it is required to access secret independent memory, which cannot be provided by the leakage independent compiler passes.

An inherent difficulty is the execution of gadgets in contexts which are determined only after construction is complete (i.e., during compilation) as it corresponds to dynamic composition of masked implementations: Vulnerabilities might arise when a set of shares is reused as input to multiple gadgets as most composition strategies for masked gadgets assume independently shared inputs (e.g. t-SNI [BBD+16]). The easiest approach is to refresh all inputs within each gadget. A more efficient approach is to analyze which shares are reused during compilation and only selectively refresh these shares. Another problem arises when gadgets receive two inputs pointing to the same shares. This case can happen in non-optimized code when protecting `AND R0 R0` the corresponding gadget potentially produces vulnerable leakage. Again, this is easy to circumvent within the compiler or by refreshing inputs internally. Our compiler refreshes all shared values used more than once.

For our case study, the compiler requires gadgets to replace the assembly instructions `EORS` (exclusive or), `ANDS` (logical conjunction), `MOV` (copy), `MVNS` (copy negation), `BICS` ($a\&\neg b$), as well as gadgets to copy and refresh masked values which are adapted from [BGG+20].

8.5 Code Transformation

In this section, we describe the overall process that produces a masked executable from an unmasked C-code. The input program needs to be annotated in order to enable the tainting phase of the code transformation. This annotation consists of one or more in-line assembly statements marking the initial sensitive values in the interface functions of the code (e.g. `main` or `encrypt`). In the first step, an off-the-shelf C-compiler for the target processor is used to generate optimized assembly code from the input C-program. Specifically, the bare-metal ARM-GCC compiler is invoked with the command `arm-none-eabi-gcc {source file} -S -mthumb -mcpu=cortex-m0plus -O3`. The operations described in the following sections are then performed on the assembly code.

8.5.1 Parsing and Tainting

In the parsing step of the transformation, we extract the hierarchical structure and control flow of the input assembly program while keeping all low-level information including opcodes and register allocation used in every instruction. Our implementation relies on several assumptions to implement this: (1) The input program is composed of data and a text segment and includes a prologue and an epilogue consisting of directives. (2) The data segment occurs once and does not intersect with the text segment. (3) The text segment is divided into one or more functions that do not intersect. Each section is marked by a prologue and epilogue of directives. (4) Every function has exactly one return instruction. (5) The number of loop iterations must be known at compile time. (6) Access to memory locations which depends on sensitive data is not supported. While assumptions 1 to 3 are enforced by the compiler 4 to 6 must be guaranteed by the input program. Note that, while assumption five seems strong, the targeted algorithms, such as block ciphers, do generally not need variable loops.

On the highest level, the data and code segments of the program are identified and then processed separately. The data segment is then scanned for data objects and their meta-information is stored. In the next step, basic blocks followed by functions are identified in the text segment.

To this end, the assembly code is parsed line by line, the instruction class is determined (e.g. load/store, arithmetic, branch) and the control flow graph is constructed iteratively. Additionally, conditional blocks and program loops are identified in this process.

The following tainting step is necessary to identify the instructions that operate on sensitive data as well as sensitive data itself. To this end, our compiler creates a tainting context that holds the complete program state including registers and the relevant memory segments. The context is then copied for each encountered basic block to prevent side effects caused by splits in the program flow. Note that the tainting process is also able to determine the size of arrays that contain sensitive data by relying on assumption (5) from above.

8.5.2 Instruction Replacement

In the final step of the code transformation process, the previously tainted instructions are replaced by secure gadgets. To this end, every sensitive value is replaced by a pointer to the first share of its masked realization. The required memory for this process is allocated on the stack or a data segment, depending on context. There is no need to store the size of a value or additional pointers to subsequent shares of the same variable, as the transformation assumes that all variables have a size of one word. Therefore, the other shares of the value can be placed successively in memory as depicted in Fig. 8.2. For every array in the input data segment that

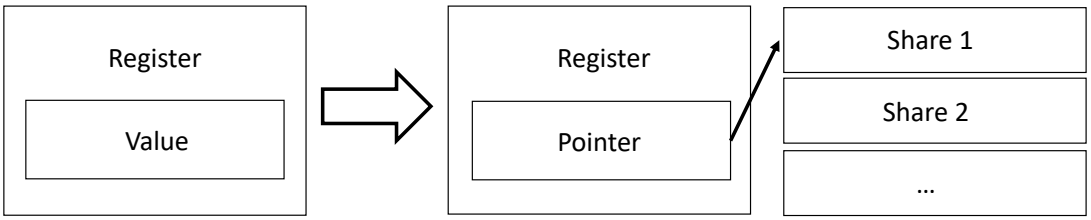


Figure 8.2: Masking of sensitive variable in register.

was tainted a new data object is allocated to store the shared data. Subsequently, the original array is replaced by pointers to the shares. The compiler also allocates a memory segment used to hold an entropy pool and a global pointer to the first entropy word. The gadgets use this pointer at runtime to obtain entropy and increment it after each access. Note that the compiler does not create code that generates the entropy needed by the secure gadgets, giving the choice about the source of randomness (e.g., internal source or external input) to the user. The final global transformation inserts code to mask and unmask the sensitive data at top-level functions.

The transformation of the program code is then applied on function level, basic-block level and finally instruction level.

Function Level The following transformations are applied to the functions: (1) Reservation of additional stack space for temporarily storing shares of sensitive variables. In our proof-of-concept implementation, sufficient space for shared values corresponding to all general purpose registers is allocated. The purpose of this "shadow register" for masked values will be explained in Sect. 8.5.2. In a future optimization, the required stack space could be reduced by analyzing which registers do specifically need this inside each function. (2) A stack base-pointer is created which allows referencing the temporary storage described above. (3) The link register is pushed.

This is necessary because the gadgets are called using linked branches and will therefore overwrite the original link register. (4) Additional push and pop instructions are inserted in order to save registers that are used by the transformation tool. (5) Some minor optimizations are performed , e.g., the removal of successive `mov` instructions with the same source and target.

Basic Block Level The transformations on the basic block level counteract the code expansion produced by the masking compiler. This is necessary as the range of simple conditional branches and unconditional branches on the target platform is only 254 B and 2 kB respectively. The code expansion can prohibit the use of either type of branch instruction. Therefore, the transformation replaces branches with linked long jumps that have a range of 16 MB, except if the branch targets its own basic block and the block is sufficiently small.

Instruction Level In the final transformation step, the actual instruction replacement is performed. As the gadgets expect their parameters and store the result in specific registers, blending code is inserted before and after the gadget call which moves the data to the correct registers. As this blending code plays an important role in overhead generation (both in time and space) a future optimization could combine "glue"-code of consecutive gadget calls. If the vulnerable instruction is not a `mov`, `load` or `store` instruction it is replaced by a call to its corresponding secure gadget at this point. For the remaining instructions, the shadow register file must be used. This is necessary because an unmasked value in the input program corresponds to a pointer in the masked program as explained above. If `load`, `store` or `mov` instruction directly operate on these pointers, semantic equivalence of the input and output program can not be assured. To see this, consider the following sequence of instructions operating directly on unmasked values: mov r1, r0; xor r1, r1, r2.

Here, the value of `r0` should not change. However, if the values are replaced by pointers and the `xor`-operation is replaced by a gadget operating on the referenced data, `r1` would now point to the (masked) value `r1 xor r2`. To prevent this, the move instruction is modified in the following way: The shared data pointed to by `r0` is copied to the shadow register file on the stack corresponding to `r1` and a pointer to the first share is stored in `r1`. Loading and storing data produces similar problems, that are solved with the same approach.

8.5.3 Security of Transformations

The compiler produces secure implementations when provided with secure gadgets. Two cases have to be considered; leakage arising during execution of gadgets and in compiler generated code. The former is true by definition. The latter holds since the compiler generated code operates on pointers to sensitive data but never accesses the data itself without using secure gadgets (including freeing of memory). The argument relies on the fact that no hidden memory accesses are performed based on pointer operations. Such behavior would become visible in the physical evaluation of securely masked gadgets and thus poses no immediate threat.

8.6 Case Study

We chose a bitsliced implementation of the LED-128 block cipher [GPPR11] as our case study target. LED-128 is based on a Substitution-Permutation Network and uses a 128 bit key to

encrypt a 64 bit plaintext to the ciphertext in $12 \cdot 4$ rounds. Each round consists of an addition of round constants, the non-linear `SubCells` layer relying on the 4-bit PRESENT s-boxes, followed by the `ShiftRow` and `MixColumnsSerial` layers. After every four rounds, a round key is added to the state. Our source C-program realizes LED as a bitsliced implementation using the whole 32-bit register width of the target platform.

8.6.1 Security Evaluation

We perform a physical validation of the s-box implementation masked by our compiler, analyzing the effectiveness of the masking against both power and Electro-Magnetic (EM) side-channel attacks.

The evaluation is performed on an industrial Arm Cortex M0+ Microcontroller Unit (MCU), specifically an FRDM-KL82Z prototyping board with removed capacitors. The supply voltage is set to 1.8 V and clock frequency of 96 MHz is used.

For measuring, a Teledyne LeCroy HDO6154 oscilloscope with a sampling rate of 2.5 GSamples/s is used. The power consumption is measured with a Teledyne LeCroy AP033 active differential probe placed in the MCU's power supply line (bandwidth 500 MHz). EM emanation is measured with Langer EMV ICR HH500-75 IC near field probe (coil diameter 500 µm, bandwidth 1 GHz).

A lateral scan over the front side of the MCU is performed to find the optimal position for the EM probe. The resulting heatmap reflects whether the EM signals acquired during the computations show clear, high peaks or whether no structures related to the computation can be observed. Step size for the lateral scan is set to one-tenth of the EM probe's coil diameter to have sufficient lateral resolution. In subsequent , the EM probe is positioned over the location providing the optimal EM signal.

To identify the period of s-box processing in the measured traces, spill actions just before and after the s-box calculations are introduced. These serve as triggers near the start of the s-box calculation by intentionally producing high values in the following t-tests (for test calibration) and to characterize the signal-to-noise ratio. Despite accurate triggers, power and EM traces are synchronized based on reference patterns and determine the highest cross-correlation of optimally shifted traces against this reference pattern.

et al. Schneider provide a detailed discussion of the underlying analysis method, capable of robustly assessing the physical vulnerability of cryptographic devices. Two-tailed t-tests are performed on sets of power consumption and EM emanation traces separately. For this, one million measurements are taken with alternating inputs, i.e., either the input was chosen randomly, or a fixed input was sent to the s-box calculation (known as fixed vs. random t-test). The critical t-value limits are shown by dashed lines in the results in Figure 8.3 and Figure 8.5. T-values above or below these lines indicate significant leakage, i.e., the detectable difference between the two groups respective their distributions and therefore distinguishability of calculations using fixed inputs and calculations using random inputs. The method allows detecting leakage without making assumption on a specific leakage model, yielding a robust indication that the implemented countermeasure efficiently prevents leakage.

The analysis is performed by sending fixed input bytes (respective random input bytes) together with required entropy bytes to the protected s-box implementation on the MCU. No leakage should be visible if the implementation is properly masked.

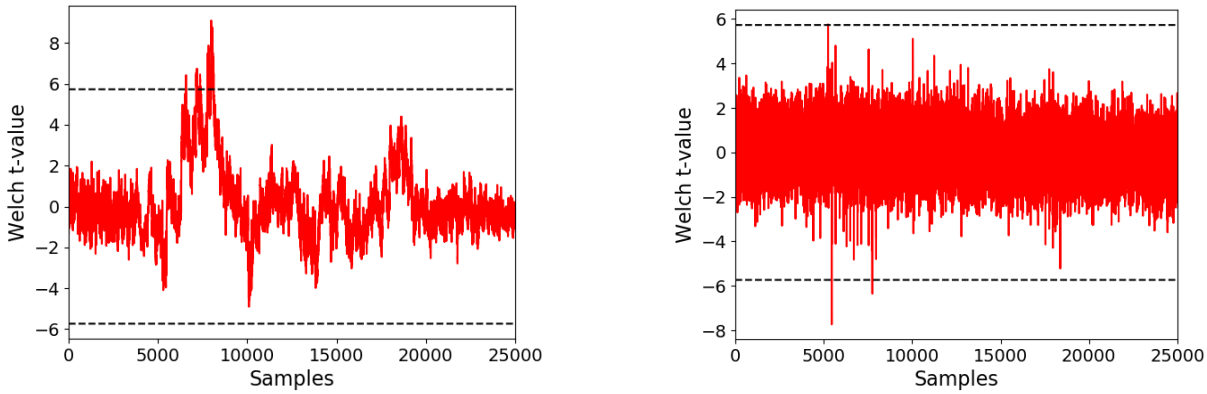


Figure 8.3: t-test results for power and EM traces in unprotected s-box

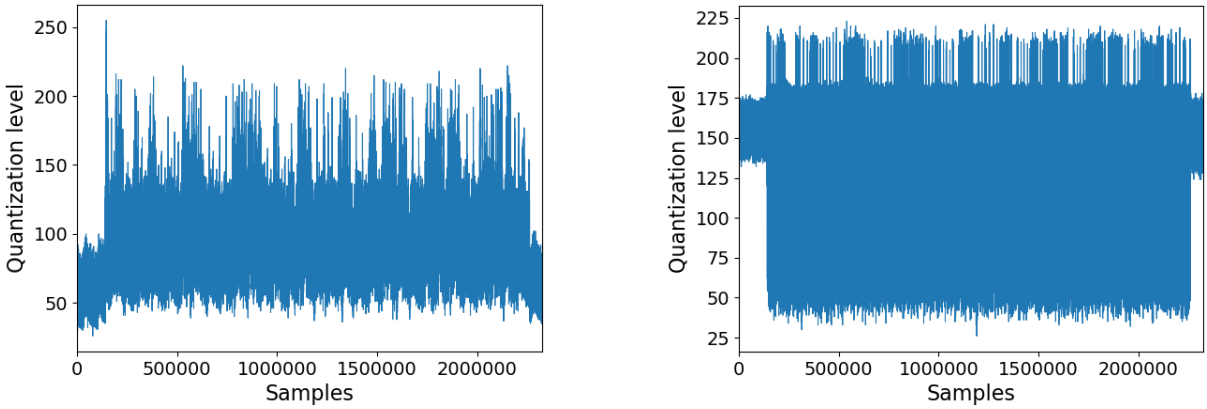


Figure 8.4: Exemplary power (left) and EM trace (right) of the protected s-box

The unprotected implementation shows detectable leakage after just 500 measurements as depicted in the results of the t-test in Figure 8.3.

For the s-box implementation masked by our compiler, one million traces were recorded, and some sample traces are depicted in Figure 8.4. The results of leakage detection visible in Figure 8.5 show that there is only at the beginning and end of the test program an indication of exploitable leakage, due to the artificially introduced spill actions. However, there is no indication of any relevant leakage during the masked s-box computation despite the high number of measurements.

We conclude that without masking, there is extreme data input dependence visible after only 500 traces. After automated protection with our tool, no data dependency is visible in both the power and EM domains, even when using 10^6 traces.

8.6.2 Performance

In this section we provide the relevant overhead produced by the proof-of-concept implementation of our proposed approach to automated masking. When applied to our LED benchmark implementation, our automated masking tool identifies 57.4 %, i.e., 666 of the 1161 instructions in the unmasked input program's assembly code as sensitive. After masking, the number of

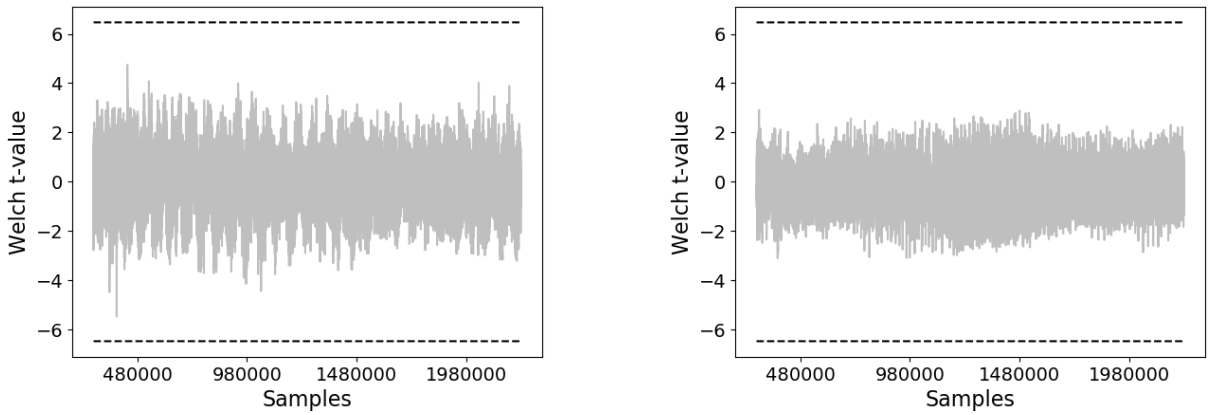


Figure 8.5: t-test results for power and EM traces in the protected s-box

instructions in the code is increased to 8418 (x7.3) and the runtime by x39.3. When our tool is applied to just the s-box, 63 out of 73 (86.3 %) instructions are sensitive. In this case, the number of instructions is increased to 835 (x11.4) and the runtime by 11.7. Note that the timing overhead includes the time needed for masking and unmasking the input data. The required randomness is and 599.6 kB and 324 B respectively.

8.7 Conclusion

We present a dependable compiler providing automated protection against first-order power side-channel attacks. The compiler substitutes vulnerable machine-code with secure gadgets specifically designed to mitigate vulnerabilities arising from device-specific leakage. Further, the security of our compiler cannot be invalidated by optimizing compilation passes which might break otherwise secure masking. The devised transformations are independent of specific side-channel behavior and thus can to be ported to devices with more diverse side-channel behavior. Our gadgets are formally verified in precise models, and the resilience in practice is successfully evaluated in physical measurements.

Since the approach naturally extends to higher-orders of security, this would be an interesting subject for future work.

Part V

Conclusion

Chapter 9

Conclusion

In this chapter, we summarize the results presented in this thesis. Finally, we provide directions for interesting future work.

Contents of this Chapter

9.1 Conclusion **119**

9.2 Related Research Areas and Future Work **120**

9.1 Conclusion

In the first part of this thesis we studied the field of leakage assessment in the context of passive physical side-channel attacks. After identifying problems in traditional test-based TVLA, we developed a new approach for leakage assessment using confidence intervals. This allows evaluators to avoid some of the main downsides of TVLA, such as its inability to provide negative results and its fundamental dependency on measurement noise. By combining the proposed scheme with an efficient side-channel data acquisition system, we provide analysts with an effective and efficient framework for side-channel leakage assessment. In order to allow evaluation of implementations against more advanced attacks that exploit multiple leakage samples, we developed an extension to the confidence-interval-based TVLA method to the multivariate setting. As the computational complexity is very high in this case, we discussed several techniques that allow a more efficient evaluation.

In the second part of this thesis we developed masked realizations for two critical functions that are central to several cryptographic implementations: a comparison for lattice-based KEMs and a modular addition. Due to the immense impact that large-scale quantum computers would have on today's digital infrastructure by rendering all non-PQC asymmetric encryption schemes obsolete, post-quantum secure algorithms are a major direction of cryptographic research. If PQC devices are potentially subjected to side-channel attacks, the implementations should be protected by countermeasures. The efficient masked algorithm that we presented can be an important component of these countermeasures.

Modular additions form an essential part of many cryptographic schemes. In this thesis, we studied Kogge-Stone, Sklansky and Brent-Kung adders in respect to their suitability for boolean masking in hardware, which is especially relevant in applications where arithmetic and boolean operations are combined. We implemented all variants using a TI and multivariately

secure HPC2 gadgets. Our analysis showed that the three architectures have different use cases depending on the constrains regarding area, latency and randomness.

The final part of this thesis focused on holistic side-channel protection measures that can protect different concrete cipher using generic approaches. This simplifies the complicated and expensive task of realizing a dedicated masking scheme for every algorithm running on a device and can thereby speed up development cycles. To this end, we developed a masked ASIP that can intrinsically protect ARX algorithms that are executed by it. In order to achieve this, the proposed architecture relies on an ALU masked with the TI scheme. A strict separation of sensitive and non-sensitive data allows a high performance although the hardware implementation is generic to support all ARX algorithms. Due to this architecture, it is possible to realize secure implementations of new ciphers without the need to consider side-channels. In our final contribution, we proposed an automatic masking approach that can be realized using off-the-shelf microcontrollers to achieve a similar result. Taking into account device specific leakage, a set of secure gadgets and a compiling process were developed that can transform unmasked implementations of bitsliced ciphers into securely masked ones.

9.2 Related Research Areas and Future Work

This section discusses some areas that, while not directly considered in this thesis, are connected to field of side-channel attacks. Additionally, we provide some directions for future work which are related to the contributions presented in this thesis.

The focus of this thesis was the field of passive side-channel attacks on cryptographic implementations. We considered instantaneous power consumption, electromagnetic emanations and – to a lesser degree – timing variations leading to information leakage. Other side channels worth studying include photonic emission [SNK+12a] due to hot-carrier luminescence, acoustic emissions due to vibrating electronic components [GST17] and thermal effects [HS13].

Microarchitectural side-channels can also be exploited and their presence poses a significant threat to computing devices. They generally rely on induced timing variations of code execution depending on sensitive data. In contrast to the types of side-channel attacks discussed in this thesis, microarchitectural side-channel attacks such as Spectre [KGG+18] do not primarily target cryptographic implementations, but general purpose CPUs and programs running on them.

Fault injection attacks form another field of powerful implementation attacks. Here an adversary tries to either disturb the control flow of the device under test or to change the data processed by the device. This can then result in either faulty computations which can then be exploited to reveal sensitive information or in bypassing security checks, e.g, for passwords. Several vectors to achieve this are available, such as voltage spikes at the target's supply lines, electromagnetic pulses in order to induce currents on the device or laser beams that can disturb small areas on the target. Finally, the joint consideration of side-channel attacks and fault attacks is another an important subject of study, both in terms of security assessment as well as in the design of countermeasures. On the attack side, either side-channel information can be used to guide fault injection attacks or fault injections are used to circumvent side-channel attack countermeasures.

Leakage Assessment

While the introduction of confidence intervals to TVLA solves some problems of this evaluation approach, others are still open. For example, in fixed-versus-random measurements, it is not immediately obvious how to choose the fixed value. While there are partial solutions for this problem, e.g., by choosing the fixed value randomly and repeating the evaluation multiple times, the evaluation results can depend on that choice and the number of repetitions.

In the context of multivariate assessment, the computational complexity is still a significant obstacle. Therefore, improvements in this regard are very welcome, especially if the worst-case sensitivity loss of complexity reductions can be bound.

Masking Cryptographic Primitives

Several lattice-based KEMs do not use a prime modulus, but are based on a ring with a power-of-two modulus, were our proposed comparison can not be applied. As the authors of [BDH+21] noted, it is possible to perform a modulus conversion in this case, but the associated overhead would likely nullify the performance gains of our scheme. Therefore, an adaption to non-prime moduli might be a worthwhile research topic. Additionally, the complete realization of multivariately secure PQC schemes, especially KEMs, will help to provide insights on the overhead induced by countermeasures.

In regard to masked hardware architectures for modular addition, other structures, such as Knowles or Han-Carlson adders, could be studied. The application of different masking schemes could also potentially improve the performance. Furthermore, the architectures studied in this thesis have a regular structure which might be exploited to reduce the randomness requirements.

Side-Channel Resistant Integrated Systems

Both approaches presented in the final part of this theses can be extended to provide multivariate security. In the case of SPARX, this could be achieved by realizing the secure ALU using higher-order secure masking schemes. However, care must be taken regarding the load-store architecture that might require adaptions.

The masked compiler could be extended to provide security against multivariate attacks by substituting the proposed gadgets by higher-order secure ones. As this will change the number of shares, the memory allocation of the compiler needs to be adapted. As the proof-of-concept compiler that we developed generates significant overhead in terms of randomness consumption and clock cycles, more efficient designs could increase practical usability.

Part VI

Appendix

Appendix A

Supplementary Material

A.1 Moment Estimation using Histograms.

The required statistical moments can be efficiently computed from histograms as shown in [RGV17] for the univariate case. We implemented a similar approach extended to the multivariate case. However, even in a bivariate scenario computing a two-dimensional histogram over 16 bit combined values would require 65 535 bins *per two-dimensional sample point*, which itself is quadratic over the one-dimensional samples, e.g., roughly 6 TB for 8 bit 1D traces of 5000 samples using 32 bit counters. Note that one histogram *per set* is needed, doubling the memory requirement.

To reduce the number of histogram bins, we normalize the input traces *per point* by the minimum of the first few thousands traces and record the amplitude of the normalized signal. In our experiments this reduced the bins necessary for the histogram computation by multiple factors of two. We keep the normalization trace and thus do not loose any information.

The runtime of this approach was not significantly lower in our experiments. We suspect that the semi-random memory access pattern on the very large histograms is a main cause for the absence of performance gains similar to the univariate case and therefore recommend the traditional algorithm of [SM15] for the calculation of the statistical moments.

A.2 Evaluation Results for DOM-AES.

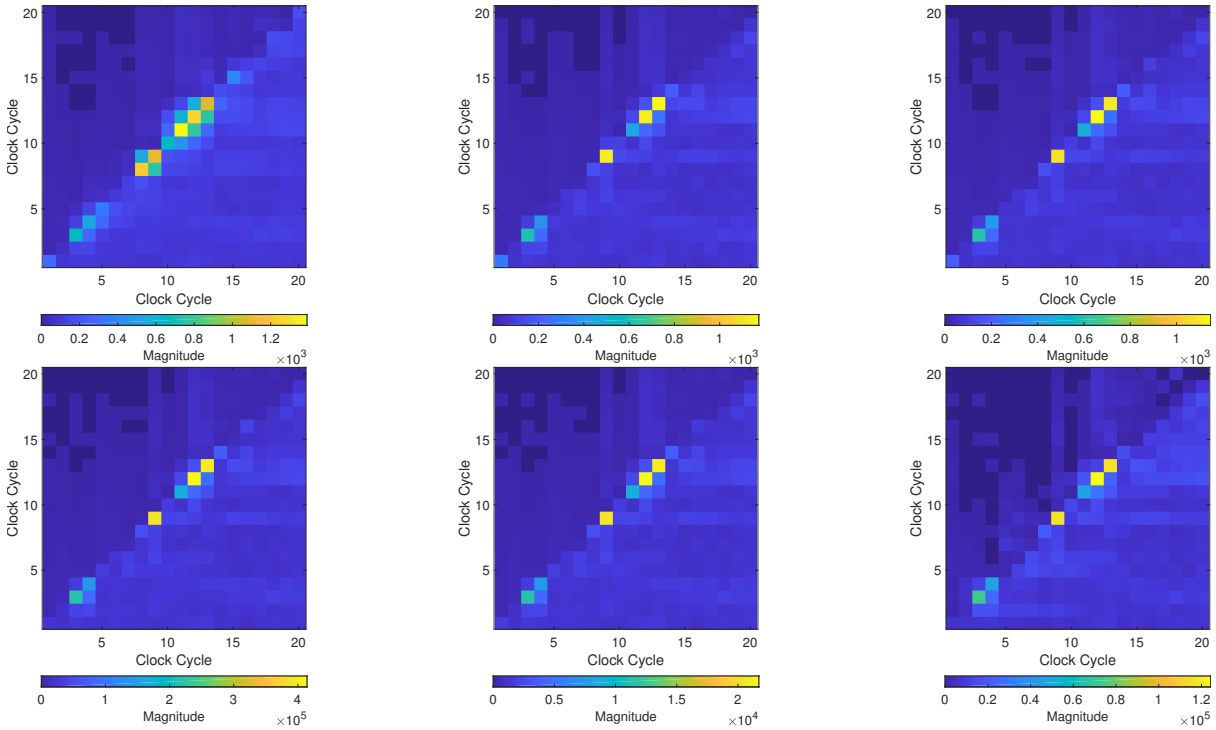


Figure A.1: Per-cycle confidence interval maps for DOM-AES with 16-fold AoT. Best parameters regarding detected tuples are chosen according to Table A.1. From top-left: Full, downsampling, averaging, 1-norm, 2-norm, PCA.

Table A.1: Evaluation results for the masked DOM-AES implementation. Confidence interval bounds are in mV^2.

Algorithm	Dim.	$\gamma_{\min}$	$\gamma_{\max}$	$\gamma_{\min}/\gamma_{\max}$	#Tuples
Full	312.5	3393.51	4065.50	0.83	185
Downsampling	1	710.23	797.14	0.89	123
	4	2219.41	2502.42	0.85	164
	16	2863.33	3412.99	0.84	178
Averaging	1	946.83	1086.57	0.87	80
	4	2314.2	2687.4	0.86	167
	16	2800.8	3293.82	0.85	178
1-Norm	1	$6.18 \cdot 10^6$	$8.35 \cdot 10^6$	0.74	154
	4	$1.38 \cdot 10^7$	$1.59 \cdot 10^7$	0.87	168
	16	$1.01 \cdot 10^6$	$1.19 \cdot 10^6$	0.85	174
2-Norm	1	$4.35 \cdot 10^4$	$5.86 \cdot 10^4$	0.74	128
	4	$1.47 \cdot 10^5$	$1.71 \cdot 10^5$	0.86	168
	16	$5.30 \cdot 10^4$	$6.23 \cdot 10^4$	0.85	174
PCA	1	$3.08 \cdot 10^5$	$3.12 \cdot 10^5$	0.87	78
	4	$3.03 \cdot 10^5$	$3.54 \cdot 10^5$	0.86	131
	16	$3.00 \cdot 10^5$	$3.56 \cdot 10^5$	0.84	127

A.3 Subroutines of A2B Conversion

For more details on the algorithms and a definition of the subroutines, we refer to the original publications [CGV14] and [BBE+18].

Algorithm 6 `Expand` [CGV14]

Input: $(x_i)_{1 \leqslant i \leqslant n} \in \mathbb{F}_2$
Output: $(y_i)_{1 \leqslant i \leqslant 2n} \in \mathbb{F}_2$ such that $\bigoplus_{i=1}^{2n} y_i = \bigoplus_{i=1}^{n} x_i$
1: $(r_i)_{1 \leqslant i \leqslant n} \xleftarrow{\$} \mathbb{F}_2$
2: $(y_i)_{1 \leqslant i \leqslant n} \leftarrow (x_i \oplus r_i)_{1 \leqslant i \leqslant n}$
3: $(y_{2i+1})_{1 \leqslant i \leqslant n} \leftarrow (r_i)_{1 \leqslant i \leqslant n}$ **return** $(y_i)_{1 \leqslant i \leqslant 2n}$

Algorithm 7 `SecAdd` [BBE+18]

Input: $\mathbf{x} = (x_i)_{1 \leqslant i \leqslant n} \in \mathbb{F}_{2^k}$, $\mathbf{y} = (y_i)_{1 \leqslant i \leqslant n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$
Output: $\mathbf{z} = (z_i)_{1 \leqslant i \leqslant n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i z_i = x + y \mod 2^k$

$\mathbf{p} \leftarrow \mathbf{x} \oplus \mathbf{y}$
$\mathbf{g} \leftarrow \texttt{SecAnd}(\mathbf{x}, \mathbf{y})$
for $j = 1$ to $W = \lceil \log_2(k-1) \rceil - 1$ **do**
  pow $\leftarrow 2^{j-1}$
  $\mathbf{a} \leftarrow \mathbf{g} << (\text{pow})$
  $\mathbf{a} \leftarrow \texttt{SecAnd}(\mathbf{a}, \mathbf{p})$
  $\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$
  $\mathbf{a}' \leftarrow \mathbf{p} << (\text{pow})$
  $\mathbf{a}' \leftarrow \texttt{RefreshXOR}(\mathbf{a}', k)$
  $\mathbf{p} \leftarrow \texttt{SecAnd}(\mathbf{p}, \mathbf{a}')$
end for
$\mathbf{a} \leftarrow \mathbf{g} << (2^W)$
$\mathbf{a} \leftarrow \texttt{SecAnd}(\mathbf{a}, \mathbf{p})$
$\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$
$\mathbf{z} \leftarrow \mathbf{x} \oplus \mathbf{y} \oplus (\mathbf{g} << 1)$

Bibliography

[AAB+] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, and Douglas Stebila. NewHope Algorithm Specifications and Supporting Documentation. `https://newhopecrypto.org/data/NewHope_2018_12_02.pdf`.

[ABB+21] Arnold Abromeit, Florian Bache, Leon A. Becker, Marc Gourjon, Tim Güneysu, Sabrina Jorn, Amir Moradi, Maximilian Orlt, and Falk Schellenberg. Automated masking of software implementations on industrial microcontrollers. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*, pages 1006–1011. IEEE, 2021.

[ABK19] Alric Althoff, Jeremy Blackstone, and Ryan Kastner. Holistic power side-channel leakage assessment: Towards a robust multidimensional metric. In *ICCAD*, pages 1–8. ACM, 2019.

[ABMP13] Giovanni Agosta, Alessandro Barenghi, Massimo Maggi, and Gerardo Pelosi. Compiler-based side channel vulnerability analysis and optimized countermeasures application. In *DAC*, pages 81:1–81:6. ACM, 2013.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium*, pages 327–343, 2016.

[AJS16] Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. Newhope on ARM Cortex-M. In *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE*, pages 332–349, 2016.

[APB+04] Riza Aditya, Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Batch verification for equality of discrete logarithms and threshold decryptions. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 494–508. Springer, 2004.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. *IACR Cryptology ePrint Archive*, 2015.

[BBD+16] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser,

Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 116–129. ACM, 2016.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In *Advances in Cryptology - EUROCRYPT*, pages 354–384, 2018.

[BCH+20] N. Belleville, D. Couroussé, K. Heydemann, Q. Meunier, and I. B. El Ouahma. Maskara: Compilation of a masking countermeasure with optimised polynomial interpolation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2020.

[BCO04a] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In Marc Joye and Jean-Jacques Quisquater, editors, *CHES 2004*, volume 3156 of *LNCS*, pages 16–29. Springer, Heidelberg, August 2004.

[BCO04b] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems – CHES'04*, volume 3156 of *LNCS*, pages 16–29. Springer, 2004.

[BDF+17] Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 535–566. Springer, Heidelberg, April / May 2017.

[BDH+21] Shivam Bhasin, Jan-Pieter D'Anvers, Daniel Heinz, Thomas Pöppelmann, and Michiel Van Beirendonck. Attacking and defending masked polynomial comparison for lattice-based cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2021(3):334–359, 2021.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 353–367, 2018.

[BDN+13] Begül Bilgin, Joan Daemen, Ventzislav Nikov, Svetla Nikova, Vincent Rijmen, and Gilles Van Assche. Efficient and first-order DPA resistant implementations of keccak. In *Smart Card Research and Advanced Applications - 12th International Conference, CARDIS 2013, Berlin, Germany, November 27-29, 2013. Revised Selected Papers*, pages 187–199, 2013.

[Ber08] Daniel J. Bernstein. The salsa20 family of stream ciphers. In Matthew J. B. Robshaw and Olivier Billet, editors, *The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, 2008.

[BG22] Florian Bache and Tim Güneysu. Boolean masking for arithmetic additions at arbitrary order in hardware. *Applied Sciences*, 12(5), 2022.

[BGG+14] Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014.

[BGG+20] Gilles Barthe, Marc Gourjon, Benjamin Grégoire, Maximilian Orlt, Clara Paglialonga, and Lars Porth. Masking in fine-grained leakage models: Construction, implementation and verification. *IACR Cryptol. ePrint Arch.*, page 603, 2020.

[BGH+15] Nicolas Bruneau, Sylvain Guilley, Annelie Heuser, Damien Marion, and Olivier Rioul. Less is more - dimensionality reduction from a theoretical perspective. In *CHES*, volume 9293 of *Lecture Notes in Computer Science*, pages 22–41. Springer, 2015.

[BGN+14] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Higher-Order Threshold Implementations. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, number 8874 in Lecture Notes in Computer Science, pages 326–343. Springer Berlin Heidelberg, December 2014.

[BHvW12] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from PCA: first results of using principal component analysis for extensive power analysis. In *CT-RSA*, volume 7178 of *Lecture Notes in Computer Science*, pages 383–397. Springer, 2012.

[BK82] Richard P. Brent and H. T. Kung. A regular layout for parallel adders. *IEEE Trans. Computers*, 31(3):260–264, 1982.

[BKP08] Andrey Bogdanov, Ilya Kizhvatov, and Andrei Pyshkin. Algebraic methods in side-channel collision attacks and practical collision detection. In Dipanwita Roy Chowdhury, Vincent Rijmen, and Abhijit Das, editors, *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 251–265. Springer, Heidelberg, December 2008.

[BNN+15a] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia Tokareva, and Valeriya Vitkup. Threshold implementations of small s-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.

[BNN+15b] Begül Bilgin, Svetla Nikova, Ventzislav Nikov, Vincent Rijmen, Natalia N. Tokareva, and Valeriya Vitkup. Threshold implementations of small s-boxes. *Cryptography and Communications*, 7(1):3–33, 2015.

[BPG18] Florian Bache, Christina Plump, and Tim Güneysu. Confident leakage assessment - A side-channel evaluation framework based on confidence intervals. In Jan Madsen

and Ayse K. Coskun, editors, *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 1117–1122. IEEE, 2018.

[BPO+20] Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. High-speed masking for polynomial comparison in lattice-based kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):483–507, 2020.

[BPW+19] Florian Bache, Christina Plump, Jonas Wloka, Tim Güneysu, and Rolf Drechsler. Evaluation of (power) side-channels in cryptographic implementations. *it Inf. Technol.*, 61(1):15–28, 2019.

[BRN+15] Ali Galip Bayrak, Francesco Regazzoni, David Novo, Philip Brisk, François-Xavier Standaert, and Paolo Ienne. Automatic application of power analysis countermeasures. *IEEE Trans. Computers*, 64(2):329–341, 2015.

[BSMG17] Florian Bache, Tobias Schneider, Amir Moradi, and Tim Giineysu. SPARX - A side-channel protected processor for arx-based cryptography. In David Atienza and Giorgio Di Natale, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 990–995. IEEE, 2017.

[BSS+13] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[BWSG] Florian Bache, Jonas Wloka, Pascal Sasdrich, and Tim Güneysu. Multivariate TVLA - efficient side-channel evaluation using confidence intervals. *IEEE Transactions on Computers*. to appear.

[Cas] Gaëtan Cassiers. "fullverif". `https://github.com/cassiersg/fullverif`. Accessed: 2021-09-30.

[CBG+16] Thomas De Cnudde, Begül Bilgin, Benedikt Gierlichs, Ventzislav Nikov, Svetla Nikova, and Vincent Rijmen. Does coupling affect the security of masked implementations? Cryptology ePrint Archive, Report 2016/1080, 2016. `http://eprint.iacr.org/2016/1080`.

[CBR+15] Thomas De Cnudde, Begül Bilgin, Oscar Reparaz, Ventzislav Nikov, and Svetla Nikova. Higher-order threshold implementation of the AES s-box. In *CARDIS*, volume 9514 of *Lecture Notes in Computer Science*, pages 259–272. Springer, 2015.

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *CHES 2000*, volume 1965 of *LNCS*, pages 252–263. Springer, Heidelberg, August 2000.

[CGD18] Yann Le Corre, Johann Großschädl, and Daniel Dinu. Micro-architectural power simulator for leakage assessment of cryptographic software on ARM cortex-m3 processors. In *COSADE*, volume 10815 of *LNCS*, pages 82–98. Springer, 2018.

[CGLS20] Gaëtan Cassiers, Benjamin Grégoire, Itamar Levi, and François-Xavier Standaert. Hardware private circuits: From trivial composition to full verification. *IACR Cryptol. ePrint Arch.*, 2020:185, 2020.

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between boolean and arithmetic masking of any order. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 188–205. Springer, 2014.

[CITE15] Cong Chen, Mehmet Sinan Inci, Mostafa Taha, and Thomas Eisenbarth. Spectre: A tiny side-channel resistant speck core for fpgas. *IACR Cryptology ePrint Archive*, 2015:691, 2015.

[CJRR99] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 398–412. Springer, Heidelberg, August 1999.

[CPRR13] Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In *FSE*, volume 8424 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 2013.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES 2002*, volume 2523 of *LNCS*, pages 13–28. Springer, Heidelberg, August 2003.

[CS20] Gaëtan Cassiers and François-Xavier Standaert. Trivially and efficiently composing masked gadgets with probe isolating non-interference. *IEEE Trans. Inf. Forensics Secur.*, 15:2542–2555, 2020.

[DSV+15] François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 34–50, 2015.

[DZD+17] A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In *CARDIS*, volume 10728 of *Lecture Notes in Computer Science*, pages 105–122. Springer, 2017.

[EW14] Hassan Eldib and Chao Wang. Synthesis of masking countermeasures against side channel attacks. In *CAV*, volume 8559 of *LNCS*, pages 114–130. Springer, 2014.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.

[GBC+08] Benedikt Gierlichs, Lejla Batina, Christophe Clavier, Thomas Eisenbarth, Aline Gouget, Helena Handschuh, Timo Kasper, Kerstin Lemke-Rust, Stefan Mangard, Amir Moradi, and Elisabeth Oswald. Susceptibility of eSTREAM Candidates towards Side Channel Analysis. In *SASC - The State Of The Art Of Stream Ciphers*, pages 123–150, 2008.

[GBT07] Benedikt Gierlichs, Lejla Batina, and Pim Tuyls. Mutual information analysis – a universal differential side-channel attack. Cryptology ePrint Archive, Report 2007/198, 2007. `http://eprint.iacr.org/2007/198`.

[GJJR11] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011.

[GMK16a] Hannes Gross, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. Cryptology ePrint Archive, Report 2016/486, 2016. `http://eprint.iacr.org/2016/486`.

[GMK16b] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. In Begül Bilgin, Svetla Nikova, and Vincent Rijmen, editors, *Proceedings of the ACM Workshop on Theory of Implementation Security, TIS@CCS 2016 Vienna, Austria, October, 2016*, page 3. ACM, 2016.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES 2001*, volume 2162 of *LNCS*, pages 251–261. Springer, Heidelberg, May 2001.

[GPPR11] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In *CHES*, volume 6917 of *LNCS*, pages 326–341. Springer, 2011.

[Gro15] Hannes Gross. Sharing is caring - on the protection of arithmetic logic units against passive physical attacks. In *RFIDSec*, volume 9440 of *Lecture Notes in Computer Science*, pages 68–84. Springer, 2015.

[Gro16a] Hannes Groß. DOM protected hardware implementation of AES. Available at `https://github.com/hgrosz/aes-dom` as of April 3, 2023, 2016.

[Gro16b] Hannes Gross. Dom protected hardware implementation of aes. `https://github.com/hgrosz/aes-dom`, 2016.

[GST14] Daniel Genkin, Adi Shamir, and Eran Tromer. RSA key extraction via low-bandwidth acoustic cryptanalysis. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 444–461. Springer, Heidelberg, August 2014.

[GST17] Daniel Genkin, Adi Shamir, and Eran Tromer. Acoustic cryptanalysis. *J. Cryptol.*, 30(2):392–443, 2017.

[HOM06] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *ACNS 06*, volume 3989 of *LNCS*, pages 239–252. Springer, Heidelberg, June 2006.

[HS13] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *CARDIS*, volume 8419 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2013.

[iot21] State of iot 2021, Sep 2021.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003*, pages 463–481, 2003.

[KGG+18] Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. *CoRR*, abs/1801.01203, 2018.

[KJJ99a] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[KJJ99b] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in Cryptology – CRYPTO'99*, LNCS, pages 388–397. Springer, 1999.

[KMRV18] Angshuman Karmakar, Jose M. Bermudo Mera, Sujoy Sinha Roy, and Ingrid Verbauwhede. Saber on ARM CCA-secure module lattice-based key encapsulation on ARM. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, (3):243–266, 2018.

[KN10] Dmitry Khovratovich and Ivica Nikolić. Rotational Cryptanalysis of ARX. In *Fast Software Encryption*, number 6147 in Lecture Notes in Computer Science, pages 333–346. Springer, 2010.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM cortex-m4. *IACR Cryptology ePrint Archive*, 2019:844, 2019.

[KRVV19] Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Pushing the speed limit of constant-time discrete gaussian sampling. A case study on the falcon signature scheme. In *Proceedings of the 56th Annual Design Automation Conference, DAC*, pages 88:1–88:6, 2019.

[KS73] Peter M. Kogge and Harold S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Trans. Computers*, 22(8):786–793, 1973.

[LLZ+18] Xianhui Lu, Yamin Liu, Zhenfei Zhang, Dingding Jia, Haiyang Xue, Jingnan He, and Bao Li. LAC: practical ring-LWE based public-key encryption with byte-level modulus. *IACR Cryptology ePrint Archive*, page 1009, 2018.

[MAS15] Bodhisatwa Mazumdar, Sk Subidh Ali, and Ozgur Sinanoglu. Power analysis attacks on ARX: an application to salsa20. In *IOLTS*, pages 40–43. IEEE, 2015.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium - efficient implementation and side-channel evaluation. In *Applied Cryptography and Network Security, ACNS*, pages 344–362, 2019.

[Min] Mini Circuits. ZFL-1000LN+ Datasheet. `https://www.minicircuits.com/pdfs/ZFL-1000LN+.pdf`.

[MM13] Amir Moradi and Oliver Mischke. On the simplicity of converting leakages from multivariate to univariate - (case study of a glitch-resistant masking scheme). In Guido Bertoni and Jean-Sébastien Coron, editors, *CHES 2013*, volume 8086 of *LNCS*, pages 1–20. Springer, Heidelberg, August 2013.

[MOBW13] Luke Mather, Elisabeth Oswald, Joe Bandenburg, and Marcin Wojcik. Does my device leak information? An a priori statistical power analysis of leakage detection tests. Cryptology ePrint Archive, Report 2013/298, 2013. `http://eprint.iacr.org/2013/298`.

[Moo19] Dustin Moody. Round 2 of NIST PQC competition. *Talk at PQCrypto 2019, Chongqing, China, 2019*, 2019.

[MOP07] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[MOPT12] Andrew Moss, Elisabeth Oswald, Dan Page, and Michael Tunstall. Compiler assisted masking. In *CHES*, volume 7428 of *LNCS*, pages 58–75. Springer, 2012.

[MOW17] David McCann, Elisabeth Oswald, and Carolyn Whitnall. Towards practical tools for side channel aware software engineering: 'grey box' modelling for instruction leakages. In *USENIX*, pages 199–216, 2017.

[MPO05] Stefan Mangard, Norbert Pramstaller, and Elisabeth Oswald. Successfully attacking masked AES hardware implementations. In *CHES*, volume 3659 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2005.

[MRSS18] Amir Moradi, Bastian Richter, Tobias Schneider, and François-Xavier Standaert. Leakage detection with the x2-test. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):209–237, 2018.

[MS16] Amir Moradi and Tobias Schneider. Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series. Cryptology ePrint Archive, Report 2016/249, 2016. `http://eprint.iacr.org/2016/249`.

[MW15] Amir Moradi and Alexander Wild. Assessment of hiding the higher-order leakages in hardware - what are the achievements versus overheads? Cryptology ePrint Archive, Report 2015/597, 2015. http://eprint.iacr.org/2015/597.

[NIS16] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. *National Institute of Standards and Technology*, December 2016. See https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[NRR06] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *ICICS*, volume 4307 of *Lecture Notes in Computer Science*, pages 529–545. Springer, 2006.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2-secure and masked Ring-LWE implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, (1):142–174, 2018.

[PMK+11] Axel Poschmann, Amir Moradi, Khoongming Khoo, Chu-Wee Lim, Huaxiong Wang, and San Ling. Side-channel resistant crypto for less than 2, 300 GE. *J. Cryptol.*, 24(2):322–345, 2011.

[PR13] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[PV17] Kostas Papagiannopoulos and Nikita Veshchikov. Mind the gap: Towards secure 1st-order masking in software. In *COSADE*, volume 10348 of *LNCS*, pages 282–297. Springer, 2017.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[RBN+15] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *CRYPTO (1)*, volume 9215 of *Lecture Notes in Computer Science*, pages 764–783. Springer, 2015.

[RdCR+16] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-LWE masking. In *Post-Quantum Cryptography - PQCrypto*, pages 233–244, 2016.

[Rep15] Oscar Reparaz. A note on the security of higher-order threshold implementations. *IACR Cryptol. ePrint Arch.*, 2015:1, 2015.

[RGV17] Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Fast leakage assessment. In *CHES*, volume 10529 of *Lecture Notes in Computer Science*, pages 387–399. Springer, 2017.

[RRdC+16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 683–702, 2015.

[RSBG20] Jan Richter-Brockmann, Pascal Sasdrich, Florian Bache, and Tim Güneysu. Concurrent error detection revisited: hardware protection against fault and side-channel attacks. In Melanie Volkamer and Christian Wressnegger, editors, *ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020*, pages 20:1–20:11. ACM, 2020.

[Rüg02] Bernhard Rüger. *Test- und Schaätztheorie Vol. 2: Statistische Tests*. Lehr- und Handbücher der Statistik. Oldenbourg, Muünchen [u.a.], 2002. XII, 570 S. : graph. Darst.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *CHES*, volume 5154 of *Lecture Notes in Computer Science*, pages 411–425. Springer, 2008.

[SAB+19] P Schwabe, R Avanzi, J Bos, L Ducas, E Kiltz, T Lepoint, V Lyubashevsky, JM Schanck, G Seiler, and D Stehle. Crystals-kyber–algorithm specifications and supporting documentation. *NIST Technical Report*, 2019.

[SAK] Side-channel AttacK User Reference Architecture. `http://satoh.cs.uec.ac.jp/SAKURA/index.html`.

[SBG+18] Markku-Juhani O. Saarinen, Sauvik Bhattacharya, Óscar García-Morchón, Ronald Rietman, Ludo Tolhuizen, and Zhenfei Zhang. Shorter messages and faster post-quantum encryption with Round5 on Cortex M. In *Smart Card Research and Advanced Applications, CARDIS*, pages 95–110, 2018.

[SBO+15] Daehyun Strobel, Florian Bache, David F. Oswald, Falk Schellenberg, and Christof Paar. Scandalee: a side-channel-based disassembler using local electromagnetic emanations. In Wolfgang Nebel and David Atienza, editors, *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, pages 139–144. ACM, 2015.

[Sid67] Zbynek Sidak. Rectangular confidence regions for the means of multivariate normal distributions. *Journal of the American Statistical Association*, 62(318):626–633, 1967.

[Skl60] Jack Sklansky. Conditional-sum addition logic. *IRE Trans. Electron. Comput.*, 9(2):226–231, 1960.

[SKR+13] Khawar Shahzad, Ayesha Khalid, Zoltán Endre Rákossy, Goutam Paul, and Anupam Chattopadhyay. Coarx: a coprocessor for arx-based cryptographic algorithms. In *DAC*, pages 133:1–133:10. ACM, 2013.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 495–513, 2015.

[SMG15a] Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic Addition over Boolean Masking. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security*, number 9092 in Lecture Notes in Computer Science, pages 559–578. Springer International Publishing, June 2015.

[SMG15b] Tobias Schneider, Amir Moradi, and Tim Güneysu. Arithmetic addition over boolean masking - towards first- and second-order resistance in hardware. In *ACNS*, volume 9092 of *Lecture Notes in Computer Science*, pages 559–578. Springer, 2015.

[SMG17] Pascal Sasdrich, Amir Moradi, and Tim Güneysu. Hiding higher-order side-channel leakage - randomizing cryptographic implementations in reconfigurable hardware. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 131–146. Springer, Heidelberg, February 2017.

[SNK+12a] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple Photonic Emission Analysis of AES. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems – CHES'12*, volume 7428 of *LNCS*, pages 41–57. Springer, 2012.

[SNK+12b] Alexander Schlösser, Dmitry Nedospasov, Juliane Krämer, Susanna Orlic, and Jean-Pierre Seifert. Simple photonic emission analysis of AES - photonic side channel analysis for the rest of us. In Emmanuel Prouff and Patrick Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 41–57. Springer, Heidelberg, September 2012.

[SPOG19] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. In *Public-Key Cryptography - PKC*, pages 534–564, 2019.

[SSB+19] Madura A. Shelton, Niels Samwel, Lejla Batina, Francesco Regazzoni, Markus Wagner, and Yuval Yarom. Rosita: Towards automatic elimination of power-analysis leakage in ciphers. *CoRR*, abs/1912.05183, 2019.

[Sta18] François-Xavier Standaert. How (not) to use welch's t-test in side-channel security evaluations. In *CARDIS*, volume 11389 of *Lecture Notes in Computer Science*, pages 65–79. Springer, 2018.

[STE15] Aria Shahverdi, Mostafa Taha, and Thomas Eisenbarth. Silent simon: A threshold implementation under 100 slices. In *HOST*, pages 1–6. IEEE Computer Society, 2015.

[SVO+10] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 112–129, 2010.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In *Theory of Cryptography - TCC*, pages 192–216, 2016.

[TV04a] Kris Tiri and Ingrid Verbauwhede. A dynamic and differential CMOS logic style to resist power and timing attacks on security IC's. Cryptology ePrint Archive, Report 2004/066, 2004. `http://eprint.iacr.org/2004/066`.

[TV04b] Kris Tiri and Ingrid Verbauwhede. A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation. In *DATE*, pages 246–251. IEEE Computer Society, 2004.

[VMKS12] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In *ASIACRYPT*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[WMM19] Felix Wegener, Thorben Moos, and Amir Moradi. DL-LA: deep learning leakage assessment: A modern roadmap for SCA evaluations. *IACR Cryptol. ePrint Arch.*, 2019:505, 2019.

[WO19] Carolyn Whitnall and Elisabeth Oswald. A critical analysis of ISO 17825 ('testing methods for the mitigation of non-invasive attack classes against cryptographic modules'). In *ASIACRYPT (3)*, volume 11923 of *Lecture Notes in Computer Science*, pages 256–284. Springer, 2019.

[WSW19] Jingbo Wang, Chungha Sung, and Chao Wang. Mitigating power side channels during compilation. In *ESEC/SIGSOFT FSE*, pages 590–601. ACM, 2019.

[YH07] J. Yan and H. M. Heys. Hardware Implementation of the Salsa20 and Phelix Stream Ciphers. In *Canadian Conference on Electrical and Computer Engineering*, pages 1125–1128, 2007.

[ZDD+17] Liwei Zhang, A. Adam Ding, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. *IACR Cryptology ePrint Archive*, 2017:287, 2017.

List of Abbreviations

ARX Addition-Rotation-XOR

ASIC Application Specific Integrated Circuit

ASIP Application-Specific Instruction-Set Processor

BKA Brent-Kung Adder

CMOS Complementary Metal Oxide Semiconductor

CPA Correlation Power Analysis

DOM Domain-Oriented Masking

DPA Differential Power Analysis

DUT Device Under Test

EM Electro-Magnetic

FPGA Field Programmable Gate Array

FWER Family-Wise Error Rate

HDL Hardware Description Language

IoT Internet of Things

ISA Instruction Set Architecture

KEM Key Encapsulation Mechanism

KSA Kogge-Stone Adder

LDA Linear Discriminant Analysis

LISA Language for Instruction Set Architecture

LUT Look-Up Table

MCU Microcontroller Unit

NI Non-Interference

NIST National Institute of Standards and Technology

PCA Principal Component Analysis

PINI Probe Isolating Non-Interference

POI Point of Interest

PQC Post-Quantum Cryptography

RNG Random Number Generator

RSA Rivest Shamir and Adleman

SA Sklansky Adder

SABL Sense Amplifier Based Logic

SCA Side-Channel Analysis

SNI Strong Non-Interference

SNR Signal to Noise Ratio

SPA Simple Power Analysis

SPN Substitution-Permutation Network

TI Threshold Implementation

TVLA Test Vector Leakage Assessment

VGA Variable-Gain Amplifier

WDDL Wave Dynamic Differential Logic

List of Figures

3.1 Measurement setup 27
3.2 Measurement sequence 29
3.3 Comparison of a power trace recorded without and with a 25MHz lowpass input filter. 30
3.4 A sample trace of the DOM AES core. 31
3.5 Maximum of absolute t-statistics and detection threshold for moments 1 and 2. . 32
3.6 Confidence intervals depicting lower (green) and upper (blue) bounds for the absolute difference in means. Please note different scales. 33
3.7 First-order confidence intervals at sample point 8715 over the number of measured traces. 34
3.8 Confidence intervals for the absolute difference in variance (2nd-order analysis). . 35
3.9 Influence of noise on a t-test-based TVLA after 4 M measurements. Dotted lines mark the critical value corresponding to a significance of 0.01. 36
3.10 Influence of noise on a confidence-interval-based TVLA after 4 M measurements. 36
3.11 A SNR-interval for first-order leakage using 150 M traces. 37
3.12 The first 1 μs of a system test for fixed-vs-random input with constant masks using 100 k traces. 37
3.13 System test for fixed-vs-fixed input with the same input for both classes after 45 M measurements. 38

4.1 Overlay of four sample traces of the sequential PRESENT implementation without (left) and with 16-fold AoT (right). 48
4.2 Per-cycle confidence interval maps for sequential PRESENT without (left) and with 16-fold (right) AoT. Lower bounds are in upper left half including the diagonal, upper bounds in the lower right half. 48
4.3 Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with downsampling reduction to 1, 4 and 16 (left to right) samples per cycle. 50
4.4 Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) AoT with averaging reduction to 1, 4 and 16 (left to right) samples per cycle. 51
4.5 Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with 1-norm reduction to 1, 4 and 16 (left to right) samples per cycle. 52
4.6 Per-cycle confidence interval maps for sequential PRESENT with 16-fold averaging over traces with 2-norm reduction to 1, 4 and 16 (left to right) samples per cycle. 52

4.7 Per-cycle confidence interval maps for sequential PRESENT without (top) and with 16-fold (bottom) averaging over traces with PCA reduction to 1, 4 and 16 (left to right) samples per cycle. 53

5.1 IND-CCA-secure variant of the `NewHope` KEM. The dashed line highlights the comparison component that is subject of this work. Bold lines indicate masked data. 64
5.2 Sample trace and reference measurement. 79
5.3 First-order SCA analysis of 2- and 3-share implementation. 80
5.4 Second-order leakage for two-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.28$). Points with t-values above the threshold are highlighted red. 80
5.5 Second-order leakage for three-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.36$). Points with t-values above the threshold are highlighted red (none present). 81

6.1 8-bit Kogge-Stone Adder. 86
6.2 8-bit Sklansky Adder. 88
6.3 8-bit Brent-Kung Adder, generation of the MSB carry bit. 89
6.4 Complete 8-bit Brent-Kung Adder. 90

7.1 High-Level Block Diagram of SPARX 98
7.2 A sample power trace, and the result of "fixed versus random" t-test using 10 million traces 103

8.1 High-level structure of our proposed automated masking approach. 107
8.2 Masking of sensitive variable in register. 111
8.3 t-test results for power and EM traces in unprotected s-box 114
8.4 Exemplary power (left) and EM trace (right) of the protected s-box 114
8.5 t-test results for power and EM traces in the protected s-box 115

A.1 Per-cycle confidence interval maps for DOM-AES with 16-fold AoT. Best parameters regarding detected tuples are chosen according to Table A.1. From top-left: Full, downsampling, averaging, 1-norm, 2-norm, PCA. 126

List of Tables

4.1 Evaluation results for the masked sequential PRESENT implementation. Confidence interval bounds are in mV^2. 54
4.2 Measurement performance for the PRESENT and DOM-AES case studies. . . . 55
4.3 Evaluation performance for the PRESENT and DOM-AES case studies in traces per second. 55

5.1 Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for $k = 1024$ including randomness generation. All results are averaged over 100 runs. 76
5.2 Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for different parameter sets including randomness generation. All results are averaged over 100 runs. 77
5.3 Random bit consumption for our ARM implementations of the masked comparison for different parameter sets. All results are averaged over 100 runs. 78

6.1 Number of elementary XOR and AND operations per basic function. 85
6.2 Number of required basic functions for different parallel-prefix adders. 87
6.3 Number of elementary XOR and AND operations for different parallel-prefix adders including the preprocessing stage and final computation of the sum bits. . 87
6.4 Implementation results for different 32-bit adder designs. 92

7.1 Relative Area Cost of SPARX' Modules 104

A.1 Evaluation results for the masked DOM-AES implementation. Confidence interval bounds are in mV^2. 127

About the Author

Author information as of December 2021.

Personal Data

Personal information is omitted in the online version.

Education

Since 10/2017	**PhD-student**, *Ruhr-Universität Bochum, Chair for Embedded Security.*
10/2015 - 09/2017	**PhD-student**, *Universität Bremen, Computer Engineering and IT-Security Group.*
10/2005 - 04/2015	**Dipl.-Ing. IT Security**, *Ruhr-Universität Bochum.* Average Score: Very Good (85%)

Professional Experience

10/2017 - 09/2021	**Research Assistant**, *Ruhr-Universität Bochum.*
10/2015 - 09/2017	**Research Assistant**, *Universität Bremen.*
09/2014 - 03/2015	**Student Trainee**, *ESCRYPT GmbH*, Bochum.
05/2014 - 09/2014	**Intern**, *ESCRYPT GmbH*, München.
04/2009 - 09/2014	**Student Assistant**, *Ruhr-Universität Bochum.*

Publications and Academic Activities

Peer-Reviewed Journal Papers

- Florian Bache, Jonas Wloka, Pascal Sasdrich, and Tim Güneysu. Multivariate TVLA - efficient side-channel evaluation using confidence intervals. *IEEE Transactions on Computers*. to appear
- Florian Bache and Tim Güneysu. Boolean masking for arithmetic additions at arbitrary order in hardware. *Applied Sciences*, 12(5), 2022
- Florian Bache, Christina Plump, Jonas Wloka, Tim Güneysu, and Rolf Drechsler. Evaluation of (power) side-channels in cryptographic implementations. *it Inf. Technol.*, 61(1):15–28, 2019

Peer-Reviewed Conference Proceeding

- Arnold Abromeit, Florian Bache, Leon A. Becker, Marc Gourjon, Tim Güneysu, Sabrina Jorn, Amir Moradi, Maximilian Orlt, and Falk Schellenberg. Automated masking of software implementations on industrial microcontrollers. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021*, pages 1006–1011. IEEE, 2021
- Florian Bache, Clara Paglialonga, Tobias Oder, Tobias Schneider, and Tim Güneysu. High-speed masking for polynomial comparison in lattice-based kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(3):483–507, 2020
- Jan Richter-Brockmann, Pascal Sasdrich, Florian Bache, and Tim Güneysu. Concurrent error detection revisited: hardware protection against fault and side-channel attacks. In Melanie Volkamer and Christian Wressnegger, editors, *ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020*, pages 20:1–20:11. ACM, 2020
- Florian Bache, Christina Plump, and Tim Güneysu. Confident leakage assessment - A side-channel evaluation framework based on confidence intervals. In Jan Madsen and Ayse K. Coskun, editors, *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 1117–1122. IEEE, 2018
- Florian Bache, Tobias Schneider, Amir Moradi, and Tim Giineysu. SPARX - A side-channel protected processor for arx-based cryptography. In David Atienza and Giorgio Di Natale, editors, *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 990–995. IEEE, 2017

- Daehyun Strobel, Florian Bache, David F. Oswald, Falk Schellenberg, and Christof Paar. Scandalee: a side-channel-based disassembler using local electromagnetic emanations. In Wolfgang Nebel and David Atienza, editors, *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, pages 139–144. ACM, 2015

Participation in Selected Conferences and Workshops

- DATE 2021, Virtual Conference
- CHES 2020, Virtual Conference
- CHES 2018, Amsterdam, Netherlands
- DATE 2018, Dresden, Germany
- DATE 2017, Lausanne, Switzerland
- DATE 2015, Grenoble, France